Восточно-Казахстанский технический университет им.Д.Серикбаева

УДК 004:023

на правах рукописи

ФЕДЬКИН ЕВГЕНИЙ МИХАЙЛОВИЧ

Разработка моделей, методов и алгоритмов составления расписаний традиционного и дистанционного обучения в высших учебных заведениях

8D06101 – Информационные системы (по отраслям)

Диссертация на соискание степени
доктора философии (Phd)

Научный консультант
к.ф.-м.н., ассоц. профессор
Денисова Н.Ф.

Зарубежный научный консультант
д.ф.-м.н., профессор
Крак Ю.В.

Республика Казахстан
Усть-Каменогорск, 2023

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей диссертации используются следующие термины и их сокращения:

SQL – Structured Query Language
БД – база данных
ВКТУ – Восточно-Казахстанский технический университет
ВУЗ – высшее учебное заведение
ПО – программное обеспечение
СУБД – система управления базами данных
СНГ – содружество независимых государств
XML – eXtensible Markup Language

ВВЕДЕНИЕ

Актуальность исследования. Задача синтеза расписаний является предметом научных исследований с середины 20 века. Область применения таких задач включает в себя различные сферы деятельности, в том числе и составление расписаний занятий. Для большинства задач нахождение оптимального расписания является трудноразрешимым, так как данные решения должны удовлетворять многочисленным ограничениям, многие из которых могут иметь конфликты между собой.

Решение задач составления различных видов расписаний относится к специальной области прикладной математики – теории расписаний. Данное направление берет свое начало от работы Генри Гантта, который предложил то, что на данный момент называют «диаграммы Гантта». Сам термин «теория расписаний» ввел в оборот Р. Беллман в 1956 году [1].

Одним из главных вопросов «теории расписаний» является классификация и определение сложности задач составления расписания. Наиболее устоявшаяся классификация был предложена Р.Грэхэмом. Имеющиеся на данный момент задачи составления расписания могут быть отнесены к следующим классам:

- NP-полные задачи;
- полиноминальные задачи;
- для некоторых видов задач, связанных с составлением расписания, так и не удалось определить класс задач, к которому они могут быть отнесены.

Для решения задачи составления расписания применяются различные методы и алгоритмы на основе различных математических моделей. Указанные математические модели строятся на применении целевой функции и набора ограничений, которые накладываются на итоговое расписание. Построенные таким образом математические модели в дальнейшем используются для дискретной оптимизации для построения оптимального расписания [1-4].

Развитие информационных технологий привело к их широкому применению в обучении и появлению такого вида обучения как дистанционное обучение. Данный вид обучения в вузах может применяться как для проведения отдельных видов занятий совместно с классическим очным обучением, так и являться основной формой обучения, когда обучение проводится полностью дистанционно. Интенсификация применения дистанционного обучения в образовательном процессе требует интеграции его в учебный процесс что важно учитывать при составлении расписаний занятий.

Развитие информационных систем управления в ВУЗах ставит задачу автоматизации составления расписания занятий на основе универсальных алгоритмов, которые могут учитывать специфику конкретного учебного заведения для различных форм обучения.

Проблеме составления расписаний посвящены многочисленные исследования как в СНГ (Клеванский Н.Н., Маслов М.Г., Галузин К.С., Низамова Г.Ф., Милехина Т.В., Асвад Фирас М. и др.), так и за рубежом (R. Lewis, M. Marte, H. Larget, M.T. Jensen, C. Mihaila и др.). Публикации по данной теме предлагают использовать различные методы и модели: классические методы, метаэвристические, мультиагентные системы, методы решения по прецедентам. Рынок автоматизированных систем также находится в развитии и многие известные ИТ-компании предлагают свои разработки в данной области

Актуальность темы данной диссертационной работы определяется необходимостью создания таких методов, моделей и алгоритмов, которые возможно применять для составления расписаний в высших учебных заведениях на основе специфики образовательного процесса конкретного учебного заведения. Разработке таких методов, моделей и алгоритмов, их теоретическому обоснованию и практическому применению посвящена данная диссертационная работа.

Объектом исследования является процесс составления расписания и его информационное сопровождение для различных видов обучения (очное, заочное, дистанционное, смешанное и др.) в высших учебных заведениях.

Предметом исследования являются модели, методы и алгоритмы, применяемые в процессе составления расписаний для различных видов обучения (очное, заочное, дистанционное, смешанное и др.) в высших учебных заведениях.

Целью исследования является разработка моделей, методов и алгоритмов для составления расписаний в высших учебных заведениях для различных видов обучения (очное, заочное, дистанционное, смешанное и др.) в высших учебных заведениях, а также создание автоматизированной информационной системы для составления расписаний занятий на основе разработанных моделей, методов и алгоритмов.

К задачам данного научного исследования относятся:

а) анализ существующих подходов к составлению расписаний;

б) разработка моделей формирования расписания занятий;

в) разработка методов и алгоритмов составления расписания на основе заданных критериев;

г) разработка автоматизированной информационной системы, которая включает в себя:

- базу данных для хранения данных, необходимых для составления расписания,

- программные модули, реализующие поддержку в работе с базой данных и создающие разработанные модели, методы и алгоритмы для составления расписания;

д) апробация разработанной автоматизированной информационной системы для составления расписаний занятий.

Методы исследования. В работе были использованы различные методы теоретизации расписаний и графов, математического моделирования, системного анализа, сложных вычислений, метода динамического программирования, а также современные методы программирования.

Научная новизна научного исследования заключается в применении модификации генетического алгоритма для составления расписания занятий с использованием функции приспособляемости на базе выполнения ограничений с использованием весовых коэффициентов.

Практическая значимость работы заключается в возможности применения предложенного алгоритма для составления расписания занятий в ВУЗах за счет ускорения составления и обеспечения оптимальности расписания.

Основные научные положения, выносимые на защиту:

а) математическая модель задачи составления расписания занятий на основе целевой функции оценки выполнения ограничений с весовыми коэффициентами, которые накладываются на расписание и разделяемых на «жесткие» и «мягкие» ограничения;

б) алгоритм генерации начальной популяции расписаний для генетического алгоритма на основе кластеризации входных данных по академическим потокам и применении целевой функции для определения оптимального включения академического потока в начальное расписание;

в) алгоритмы генетических операторов (мутации и скрещивания) для формирования новой популяции и селекции особей для формирования нового поколения, а также как условие завершения поиска решения;

г) алгоритм построения функций для реализации оценки выполнения ограничений с возможностью оптимизации вычислений;

д) архитектура информационной системы и программная реализация предложенных методов и алгоритмов составления расписания занятий.

Внедрение результатов работы. Исследование проводилось на базе НАО «Восточно-Казахстанского технического университета имени Д.Серикбаева» (далее ВКТУ). На основе данных образовательного портала была произведена апробация предложенного алгоритма составления расписания. Алгоритм был использован для составления расписания занятий на 3-й триместр 2022-2023 учебного года. Программная реализация составления расписания была внедрена в программный комплекс образовательного портала и будет использоваться в учебном процессе.

Апробация работы. Основные результаты диссертационной работы были доложены и обсуждены на следующих международных конференциях:

- CITech-2018 – Вычислительные и информационные технологии в науке, технике и образовании, г. Усть-Каменогорск, Республика Казахстан.
- SIBIRCON 2019 – International Multi-Conference on Engineering, Computer and Information Sciences, г. Новосибирск, Российская Федерация (индексируется в базе данных Scopus).

- The 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021, г. Краков, Польша (индексируется в базе данных Scopus).

На разработанный в рамках научного исследования программный комплекс составления расписания было получено авторское свидетельство №20431 от 23 сентября 2022 г. «Программа для составления расписания семестровых занятий для ВУЗов», авторы Кумаргажанова С.К., Денисова Н.Ф., Рахметуллина С.Ж., Смаилова С.С., Федькин Е.М. (Приложение Д).

Публикации. По теме научного исследования опубликовано 6 научных работ:

а) 2 научные статьи, индексируемые в базе данных Scopus:

- в журнале Acta Polytechnica Hungarica (83-й процентиль),
- в журнале Eastern-European Journal of Enterprise Technologies (48-й процентиль);

б) 3 научные статьи в журналах, входящих в список рекомендованных ККСОН для публикации научных результатов:

- Вестник Евразийского национального университета имени Л.Н.Гумилева. Серия Математика. Информатика. Механика,
- Вестник академии гражданской авиации,
- Совместный выпуск научных журналов «Вестник Восточно-Казахстанского государственного технического университета им. Д.Серикбаева» и «Вычислительные технологии»;

в) монография «Разработка модели on-line образования для высших учебных заведений РК», авторы С.К. Кумаргажанова, С.С. Смаилова, Н.Ф. Денисова, С.Ж. Рахметуллина, Е.М. Федькин, В.Н. Зуев, Л.Ж. Какишева, Усть-Каменогорск: ВКТУ, 2022. 146 с. ISBN 978-601-208-800-7.

Содержание и объем работы. Диссертация состоит из введения, четырех глав, заключения, списка использованных источников из 110 наименований. Диссертация включает 43 рисунка, 6 таблиц и 9 приложений. Общий объем диссертации включает 234 страниц.

В первом разделе представлен обзор методов и алгоритмов, применяемых для решения задачи составления расписаний занятий. Условно описанные методы могут быть разделены на 2 большие группы: точные и приближенные методы алгоритма. Точные методы и алгоритмы предполагают полный перебор всех возможных комбинации решений и выбор из них оптимального. К точным методам относятся методы ветвей и границ, методы отсечения, методы теории графов и др. Приложенные методы ориентированы на получение расписания близкого к оптимальному за приемлемое время на основе разнообразных эвристик. К данной группе относятся такие методы как метод муравьиной колонии, алгоритм имитации отжига, методы, основанные на ограничениях, нейронные сети и генетические алгоритмы.

Во втором разделе описана разработка математической модели для составления расписания занятий. Данная модель включает в себя входные

данные для расписания: учебные группы и подгруппы, преподаватели, аудитории, дисциплины, виды занятий, дни, временные интервалы и учебные потоки. Выходными данными для предложенной модели является список картежей со следующим набором значений: учебный поток, аудитория, день и временные интервалы. На составляемое расписание накладываются ограничения, которые делятся на «жесткие» (обязательные) и «мягкие» (рекомендации). Для поиска решения задачи была использована модификация генетического алгоритма на основе функции приспособляемости с помощью весовых коэффициентов и оценки выполнения ограничений. В качестве особи, представляющей расписание, была выбрана особь с одной хромосомой, которая имеет набор генов, представляющих отдельные занятия. Каждый ген имеет неизменяемую часть (академический поток), представляющую описание занятия в расписании, и изменяемую часть, представляющую, как описываемое занятие должно быть включено в расписание (день, время, аудитория). Работа предложенного генетического алгоритма строится на генерации начальной популяции с учетом кластеризации входных учебных потоков на их семантической схожести, затем цикл применения генетических операторов мутации и скрещивания, формирования новой популяции и проверки условия завершения поиска решения. Так же в данном разделе предложена схема создания функций для оценки ограничений, позволяющая оптимизировать объем вычислений.

В третьем разделе данной работы приведена схема разработки базы данных для хранения исходных данных для составления расписания. Данная база данных включает в себя таблицы для хранения описанных во втором разделе исходных данных для составления расписания. Физическая реализация базы данных произведена на основе СУБД Microsoft SQL Server 2017. Также в данном разделе приведено описание программной реализации динамической библиотеки, в которой реализован предложенный во втором разделе генетический алгоритм для составления расписания. Данная библиотека включает поддержку входных и выходных данных для расписания, компоненты для реализации проверки ограничений и сохранения итоговых результатов в формате XML. Указанная библиотека является расширяемой, что позволяет использовать её для реализации составления расписания для различных учебных заведений.

В четвертом разделе приведено описание архитектуры образовательного портала ВКТУ им. Д.Серикбаева, описаны его компоненты для расписания занятий и данные, используемые для составления расписаний. На основе проведенного анализа в данном разделе была произведена выборка данных из базы данных образовательного портала в базу данных для составления расписания, о разработке которой говорится в третьем разделе. Для апробации реализованной динамической библиотеки в третьем разделе на основе данных образовательного портала была произведена генерация расписания для 3-го триместра 2022-2023 учебного года. Также был

произведен вычислительный эксперимент для оценки предложенного генетического алгоритма. Указанный вычислительный эксперимент показал следующие результаты: предложенный алгоритм имеет полиноминальную зависимость от количества элементов в итоговом расписании; предложенный алгоритм генерации начальной популяции позволяет получить близкое к оптимальному решение, что позволяет иметь более качественное итоговое решение за счет работы циклической части генетического алгоритма.

1. ОБЗОР И АНАЛИЗ СУЩЕСТВУЮЩИХ ПОДХОДОВ, ПРИМЕНЯЕМЫХ ПРИ СОСТАВЛЕНИИ РАСПИСАНИЙ УЧЕБНЫХ ЗАНЯТИЙ

1.1 Обзор моделей, методов и алгоритмов применяемых при составлении расписаний учебных занятий

Расписание учебных занятий является базовым элементом для организации учебного процесса любого ВУЗа. Задача составления расписания учебных занятий может иметь множество вариантов решения. Вследствие данного факта возникает необходимость в нахождении оптимального решения данной задачи.

Задачи составления различных видов расписаний относятся к специальному разделу математики – теории расписаний. В рамках теории расписаний строится и проводится анализ математических моделей календарного планирования. Математические модели составления расписаний основываются на применении целевой функции и набора различных видов ограничений, которые накладываются на расписание. Построенная таким образом математическая модель в дальнейшем используются для дискретной оптимизации для построения оптимального расписания [1-4].

Расписание учебных занятий часто представляется в виде двумерной таблицы, в которой столбцы могут представлять некоторую учебную группу, студента, преподавателя или аудиторию, а строки представляют день, время и продолжительность проведения учебных занятий. При таком представлении расписания в ячейках таблицы записываются занятия в указанный период времени для учебной группы, студента, преподавателя или аудитории. Из-за такого представления расписания учебных занятий в теории расписаний данный вид расписаний получил название «Time Tabling» [5].

Поскольку задача составления расписания учебных занятий основана на применении математической модели, основанной на использовании заданной целевой функции и набора ограничений, то такие задачи относятся к задачам целочисленного программирования. Задачи целочисленного программирования являются NP-полными задачами, которые для нахождения оптимального решения требуют полного перебора возможных решений и их оценкой критериям оптимальности решения. Следовательно, с увеличением числа входных переменных и их диапазона возможных значений растет число возможных решений и, как следствие, растут временные и вычислительные затраты, связанные с поиском оптимального решения задачи [1, 6-11].

Исходя из вышесказанного, для решения задачи составления оптимального расписания учебных занятий могут применяться различные методы и алгоритмы. Данные методы и алгоритмы производят поиск оптимального расписания на основе значения целевой функции математической модели расписания. Условно данные методы и алгоритмы, в

зависимости от способа поиска решения можно разделить на 2 большие группы [2, 8, 10, 12-30]:

- точные методы – это методы, которые производят полный перебор возможных решений задачи с учетом заданных ограничений и на основе значения целевой функции определяют оптимальное решение,

- приближенные методы – это методы, которые основываются на применении некоторого набора правил поиска, которые позволяют получить оптимальное расписание или приближенное к нему за конечное число шагов.

Рассмотрим данные методы и алгоритмы более подробно.

1.2 Точные методы

Точные методы составления расписаний основываются на применении методов и алгоритмов целочисленного программирования. Работа данных методов и алгоритмов базируется на применении некоторой математической модели с ограничениями на значения переменных, которые используются в данной математической модели [31, 32].

Применение методов и алгоритмов целочисленного программирования для решения задачи составления расписания имеет следующие этапы [12, 33]:

- выделение переменных, которые описывают задачу и значение которых требуется найти для её решения;
- разработка математической модели, которая описывает задачу;
- определение ограничений, которые накладываются на решение задачи;
- определение целевой функции для задачи;
- максимизация или минимизация значения целевой функции.

Использование методов и алгоритмов целочисленного программирования для решения задачи составления расписания сталкивается с определенными трудностями, которые затрудняют их применение для решения задачи. Имеются следующие недостатки методов и алгоритмов целочисленного программирования [5, 34-36]:

- экспоненциальная зависимость времени поиска оптимального решения от размерности задачи;
- отсутствие гарантии на получение оптимального решения задачи;
- сложность оценки влияния различных факторов на поиск оптимального решения задачи;
- сложность оценки ограничений, которые накладываются на решение задачи.

Для решения задачи составления расписания занятий в основном применяются такие методы и алгоритмы целочисленного программирования как:

- методы отсечения,
- методы ветвей и границ,
- методы теории графов.

Рассмотрим, как перечисленные выше методы и алгоритмы целочисленного программирования могут применяться для решения задачи составления расписания занятий.

1.2.1 Методы отсечения. Данная группа методов и алгоритмов целочисленного программирования основываются на переходе от решения задачи целочисленного программирования – «$\Omega^{\text{ц}}F$-задачи» к решению ряда последовательных задач линейного программирования – «$\Omega_k F$-задачи», $k = 0, 1, 2, \ldots$ [12]

Схема работы в данном случае будет иметь следующую последовательность шагов:

- Решение k-ой задачи основывается на решении предыдущей $(k-1)$-ой задачи путем добавления дополнительного условия к условиям, которые задают область допустимых значений $(k-1)$-ой задачи. Данное новое условие называется «правильным отсечением» [12].

- После решения k-ой задачи производится проверка полученного решения условиям решения начальной задачи целочисленного программирования «$\Omega^{\text{ц}}F$-задачи». Если полученное решение k-ой задачи соответствует условия начальной задачи, то мы получаем решение начальной задачи и прекращаем работу итерационного цикла поиска решения. Иначе, повторяем 1-й шаг поиска.

При определении «правильного отсечения» на 1-м шаге алгоритма применяются следующие ограничения [12]:

- отсечение должно быть линейным;

- из области допустимых решений необходимо исключить ту часть, которая содержит возможные оптимальные решения k-ой задачи и которые не соответствуют требованиям исходной задачи, т.е. вводимое ограничение является «отсечением»;

- отсекаемая часть не должна включать ни одной точки, которые принадлежат области допустимых решений исходной задачи, т.е. вводимое ограничение является «правильным».

Для нахождения отсечений для k-ой задачи чаще всего применяются следующие алгоритмы [12]:

- 1-й алгоритм Гомори для решения целочисленной задачи;

- 2-й алгоритм Гомори для решения частичной целочисленной задачи;

- алгоритм Дальтона-Ллевелина для решения дискретной задачи.

Для перечисленных выше алгоритмов поиска «правильного отсечения» применяется формула, которая имеет следующее представление [12]:

$$z = -y_0^k - \sum_{j \in \text{НБ}} \left(-y_j^k\right) x_j, \qquad z \geq 0\,, \tag{1.1}$$

где, x_j – небазисные переменные k-ой задачи,

y_0, y_j – данные элементы определяются по коэффициентам разложения базисной переменной, не удовлетворяющей требованиям исходной задачи по небазисным переменным.

1.2.2 Метод ветвей и границ. Данная группа методов относится к комбинаторным методам. Работа данной группы методов заключается в упорядоченном переборе различных вариантов решения задачи составления расписания и выбора из них тех вариантов, которые имеют определенные перспективные признаки для нахождения оптимального решения задачи.

Работа метода ветвей и границ имеет следующие шаги: множество допустимых решений задачи разбивается на несколько подмножеств, а затем полученные подмножества также разбиваются на более мелкие подмножества. Данное разбиение должно идти до тех пор, пока не будет достигнуто оптимальное решение задачи [12, 37, 38].

Рассмотрим процесс работы метода ветвей и границ. Допустим, имеется некая целевая функция $f(x)$. Для данной целевой функции имеется некое допустимое множество допустимых решений G. Для решения поставленной задачи для целевой функции необходимо найти такое значение $x_0 \in G$, что её значение для данного значения будет иметь значение по следующей формуле (формула 1.2) [5]:

$$f(x_0) = \min_{x \in G} f(x) \tag{1.2}$$

Далее идет процесс решения поставленной задачи путем ветвления. Данный процесс предполагается в разбиении исходного множества G на несколько подмножеств $G_i \subseteq G, i = 1, 2, \dots, r$. При этом объединение полученных подмножеств должно давать исходное множество, т.е. $\bigcup_{i=1}^{r} G_i = G$. Таким образом, исходная задача разбивается на подзадачи, каждая из которых работает со своим подмножеством допустимых решений G_i. Полученные подзадачи в свою очередь так же разбиваются на подзадачи, т.е. процесс поиска решения является рекурсивным. В результате такого рекурсивного поиска формируется дерево поиска оптимального решения.

Для уменьшения роста дерева поиска решений в данной группе методов применяется отсечение тех ветвей дерева, которые, по некоторой оценке, являются неперспективными. Для оценки ветки дерева поиска на предмет её перспективности вычисляются следующие оценки для ветки [5, 37, 38]:

- верхняя оценка (граница) – $UB_i \geq \min_{x \in G_i} f(x)$
- нижняя оценка (граница) – $LB_i \leq \min_{x \in G_i} f(x)$.

Верхняя оценка в данном алгоритме используется для оценки перспективности работы с тои или иной веткой, т.е. данная оценка позволяет оценить содержит ли ветвь поиска оптимальное решение. Иногда верхнюю оценку не вычисляют.

Нижняя оценка в данном алгоритме используется для оценки нужно ли отсекать ветвь в дерево поиска или нет. Для отсечения ветви поиска производится сравнение нижней оценки поиска текущей и родительских веток поиска. В случае если текущая нижняя оценка оказывается больше, чем нижняя оценки родительской ветки, то текущая ветка считается неперспективной и должна быть исключена из дальнейшего поиска оптимального решения, т.е. данная ветвь отсекается.

Условием завершения рекурсивного поиска для данных видов алгоритмов является нахождение вершины дерева поиска, которая содержит только одно возможное решение задачи и для данной вершины нижняя оценка меньше или равно нижней оценки родительской ветки.

Рассмотренный вариант алгоритма для метода ветвей и границ предполагает, что производится минимизация целевой функции. В случае если необходимо произвести максимизацию целевой функции, то роли верхней и нижней оценок взаимозаменяются.

1.2.3 Метод теории графов. Для решения задачи составления расписания также может использоваться метод теории графов.

Граф представляет собой структуру, которая состоит из некоторого числа точек, которые располагаются на некоторой плоскости. Данные точки могут соединяться линиями или стрелками, которые называются ребрами графа. Ребро графа обозначает путь между двумя точками в графе [39].

В теории графов выделяют следующие виды графов [39]:

- Неориентированные графы. В данном виде графов ребра соединяются линиями, которые обозначают двунаправленный путь между вершинами графа.

- Ориентированные графы. В данном виде графов ребра соединяются стрелками, которые обозначают однонаправленный путь между вершинами графа.

Кроме того, графы могут содержать весовые коэффициенты для вершин и ребер. В этом случае граф является взвешенным графом.

Графы обозначаются как (формула 1.3) [39]

$$G = (V, E) \tag{1.3}$$

где, V – множество вершин (точек),
E – множество пар ребер графа.

Для задачи составления расписаний используются неориентированные графы со следующими характеристиками [14, 19, 34, 35, 40, 103]:

- вершины графа представляют планируемые учебные занятия

- ребра графа представляют возможные конфликты между учебными занятиями.

При таком представлении графа задача составления расписания занятий решается как задача раскраски графа в заданное количество цветов.

Исходя из вышесказанного, схема работы алгоритма основанного на применении метода теории графов будет следующая:

- определение множество учебных занятия, для которых требуется составить расписание;
- каждое занятие из определенного выше множества учебных занятий должно быть представлено как вершина неориентированного графа;
- если два занятия не могут быть проведены одновременно, то их необходимо соединить ребром;
- производится решение задачи раскраски полученного не предыдущих шагах неориентированного графа в заданное количество цветов.

Данный способ составления расписаний предоставляет такие преимущества как:

- простота применяемой математической модели,
- совместное использование данного метода с эвристическими метода позволяет получить достаточно неплохие результаты.

Рассмотренные выше точные методы гарантировано позволяют получить, если такое имеется, оптимальное расписание учебных занятий. Однако применение данных методов предполагает использование, в том или ином виде, полного перебора возможных решений. Временные и вычислительные затраты на данный полный перебор растут при увеличении возможного множества вариантов решения задачи. По этой причине для ускорения работы точных методов требуется получение необходимой информации, которая позволяет сократить возможное множество решений.

1.3 Приближенные методы

Приближенные методы и алгоритмы позволяют получить некоторое решение задачи, пусть даже не оптимальное, но приближенному к оптимальному решению, за счёт снижения вычислительных затрат на поиск. Данные методы и алгоритмы применяются вследствие того, что применение точных методов и алгоритмов требуют больших вычислительных ресурсов и в большинстве случаев не могут дать приемлемое решение в большинстве случаев [41].

Приближенные методы и алгоритмы можно условно разделить на следующие группы [12]:

- эвристические методы и алгоритмы;
- вероятностные методы и алгоритмы.

Рассмотрим, как данные методы и алгоритмы используются для решения задачи составления расписаний занятий.

1.3.1 Эвристические методы и алгоритмы. В эвристических методах и алгоритмах применяется технология понижения требований к поиску решения задачи. Данное понижение требований заключается в том, чтобы найти решение задачи близкое к оптимальному за счет понижения использования вычислительных ресурсов и за приемлемое время. Для этого в

эвристических метода и алгоритмах могут использоваться разные разумные предположения без строгих теоретических обоснований. Такие предположения в эвристических методах и алгоритмах могут достаточно эффективно применяться для поиска решения задач с высокой вычислительной сложностью (NP-полные задачи) [7, 11, 42].

NP-полные задачи обладают рядом отличительных свойств, таких как [7, 11, 12]:

- ни одну NP-полной задачу невозможно решить известными полиноминальными методами и алгоритмами

- если для какой-либо NP-полной задачи существует полиноминальный алгоритм решения, то должны существовать полиноминальные алгоритмы и для других NP-полных задач.

Вследствие того, что NP-полные задачи не имеют эффективного алгоритма, то алгоритмы, используемые для решения такого рода задач, имеют в худшем случае экспоненциальное время поиска. Применение таких алгоритмов на практике не частотно, кроме случаев задач с достаточно малым числом вариантов.

Если решаемая задача относится к NP-полным задача, то эффективность её решения может быть достигнуто за счёт применения эвристических методов, схем направленного поиска и исследования частых случаев решаемой задачи.

Эффективность применения эвристических методов и алгоритмов при решении сложных задач состоит в том, что такие методы и алгоритмы позволяют свести поиск решения NP-полной задачи к полиноминальной сложности. Таким образом, NP-полная задача становится P-задачей.

Для получения алгоритма решения NP-полной задачи ряд ученых предложил следующий путь [12]:

- необходимо для решаемой задачи найти такие теоретические свойства, которым должно удовлетворять оптимальное решение задачи;

- на основе полученных выше свойств должен быть разработан полиноминальный алгоритм для решения задачи.

Полученный такий образом полиноминальный алгоритм решения NP-полной задачи должен обеспечивать выполнения ряда требований, таких как:

- если полученное решение какой-либо индивидуальной задачи выполняются определенные условия, то такая задача решается строго с получением строгого оптимального решения;

- результат применения полученного полиноминального алгоритма всегда можно понять была ли решена задача или нет;

- полученный алгоритм должен быть эффективным и статистически значимым, т.е. использование данного алгоритма на произвольных задачах должно в большинстве случае давать точный результат.

Применение эвристических методов и алгоритмов могут давать достаточную эффективность для широкого круга задач, несмотря на то, что применение таких методов может иметь некоторые уязвимости. Среди

эвристических методов и алгоритмов, применение которых может быть эффективным для решения NP-полных задач, можно выделить следующие:

- Метод ветвей и границ без возвратов. Применение данного эвристического метода позволяет сократить область поиска за счёт того, что итоговый результат решения задач не обязательно должен быть точным. В данном варианте алгоритма может быть отброшена то лишняя информация о вершинах поиска с обязательным сохранением только нижней оценки отброшенных вершин. Такой вариант метода позволяет получить нижнюю и верхнюю оценку решения, что позволяют обеспечить точность решения.

- Метод локального поиска. Применение данного алгоритма основывается на применении некоторого изначального набора перестановок, которые используются для поэтапного улучшения изначального решения до тех пор, пока возможно улучшения решения, в противном случае считается, что мы получили локальный оптимум для решения.

- Метод локальных правил. Данный метод предполагает использование методов моделирования, в которых используются специальные правила, которые применяются при ручном составлении расписаний. Применение таких локальных правил позволяют получить решение близкое к оптимальному. Для разрешения конфликтов между локальными правилами применяется система приоритетов, которые определяют, какое именно правило должно быть применено при возникновении противоречий.

Несмотря на то, что эвристические методы и алгоритмы могут быть достаточно эффективны для решения задач поиска оптимального решения, они имеют ряд недостатков, такие как:

- сложность оценки полученного решения на близость его к оптимальному решению;

- эвристические методы и алгоритмы могут давать результаты, которые могут быть далеки от оптимальных.

1.3.2 Вероятностные методы и алгоритмы. Вероятностные методы и алгоритмы являются разновидностью эвристических методов и алгоритмов. Отличительной особенностью данных методов и алгоритмов является кратное моделирование расписаний. Моделирование предполагает работу с набором расписаний, с которыми производятся различные операции поиска и после ряда проигрышей определяется наилучшее решение из множества, которое принимается за решение задачи.

Поиск в вероятностных алгоритма может осуществляться в следующих направлениях [12]:

- ненаправленный случайный поиск;

- направленный случайный поиск без самообучения;

- направленный случайный поиск с самообучением.

Применение вероятностных методов и алгоритмов, как правило, строится на применении разного рода эвристик. Данные эвристики строятся на использовании некоторых интуитивных предположений, которые позволяют получить оптимальное решение. Использование данных эвристик

позволяет ускорить поиск решения, приближенного к некоторому оптимуму. При применении таких эвристик может возникнуть проблема оценки решения, так как сложно определить находимся ли мы в локальном или глобальном максимуме.

Для составления расписаний чаще всего используются следующие вероятностные методы и алгоритмы:

- методы, основанные на ограничениях;
- алгоритм имитации отжига;
- алгоритм муравьиной колонии;
- генетические алгоритмы;
- нейронные сети,

1.3.2.1 Методы, основанные на ограничениях. Применение методов, основанных на ограничениях для решения задачи составления расписаний учебных занятий, сводится к решению задачи удовлетворения ряда ограничений. Процесс формирования расписания в данных методах и алгоритмах выглядит как процесс распределения времени и ресурсов между различными занятиями с учетом выполнения множества ограничений, как правило, на основе некоторого набора правил [43-45, 104-107].

Ограничения в данных методах, как правило, разделяются на две группы:

- «Жесткие» ограничения – это ограничения, которые являются обязательными для составляемого расписания занятий;
- «Мягкие» ограничения – это ограничения, которые являются рекомендательными для составляемого расписания занятий.

В процессе составления расписания в первую очередь должны быть выполнены «жесткие» ограничения. В случае, если возникает противоречие между «жестким» и «мягким» ограничением приоритет выполнения отдается «жесткому» ограничению, а «мягкое» ограничение игнорируется. Данный процесс проверки ограничений повторяется до тех пор, пока есть возможность выполнить как можно больше имеющихся ограничений.

Применение методов, основанных на правилах, позволяет существенно сократить пространство поиска для задачи составления расписаний занятий. Однако, результаты применения данных методов имеет сильную зависимость от выбора начальной версии расписания занятий, что затрудняет применение данных методов.

1.3.2.2 Алгоритм имитации отжига. Данный вид алгоритмов базируется на имитации физического процесса, который имеет место при кристаллизации вещества из жидкости в твердое состояние [34, 46-49]. Использование алгоритма имитации отжига предполагает, что имеется возможность перехода отдельных атомов вещества из одной ячейки в другую в выстроенной кристаллической решетке некоторого вещества. Данный процесс перехода атомов внутри кристаллической решетки вещества происходит при постепенном снижении температуры вещества. Переход атомов между ячейками кристаллической решетки происходит с некоторой

вероятностью. Вероятность данного перехода имеет прямую зависимость от текущей температуры вещества, т.е. чем более низкая температура вещества, тем ниже будет вероятность перехода атомов между ячейками кристаллической решетки.

Работа алгоритма имитации отжига идет по следующей схеме [12]:

а) Создается некое начальное расписание занятий Z_0. Данное расписание становится текущим расписанием, т.е. $Z = Z_0$ и для данного расписания назначается высокая начальная температура $T = T_0$.

б) Изменение компонентов расписания. На данном этапе может быть произведено изменение времени, аудитории и других возможных элементов расписания занятий. По завершении данного этапа у нас будет сформировано новое расписание занятий Z'.

в) Производится вычисление значение целевой функции для текущего расписания Z и нового расписания Z', а затем определяется разница между полученными значениями (формула 1.4):

$$\Delta f = f(Z') - f(Z) \tag{1.4}$$

Если дельта между значениями целевых функций имеет отрицательное или нулевое значение, т.е. $\Delta f \leq 0$, то новое расписание становится текущим расписанием, т.е. $Z = Z'$. В противном случае, т.е. $\Delta f > 0$, вероятность того, что новое расписание станет текущим расписания определяется по формуле 1.5:

$$p = e^{-\frac{\Delta f}{T}} \tag{1.5}$$

г) Далее понижаем текущую температуру процесса T. Понижение текущей температуры приводит, согласно формуле 1.5, приводит к снижению вероятности замены текущего расписания новым расписаниям при каждой итерации алгоритма.

д) Если не выполняется условие завершения работы алгоритма, то переходим на шаг под пунктом б). В качестве условия остановки итераций алгоритма, как правило, является либо достижение определенной температуры вещества или если значение целевой функции перестало изменяться на заданном числе итерационных шагов.

Алгоритмы данного вида показывают высокую эффективность при генерации небольших расписаний занятий.

К недостатками данных алгоритмов относятся:

- применение данных алгоритмов для расписаний с большой размерностью имеют малую эффективность;

- для повышения эффективности поиска решения требуется применение схем Коши или Больцмана, что резко повышает вычислительные затраты для поиска решения задачи.

1.3.2.3 Алгоритм муравьиной колонии. Алгоритм муравьиной колонии базируется на биологической способности муравьев находить кратчайший путь к пище. Для поиска данного пути муравьи выделяют специальный фермент, который называется феромон [50].

Данный алгоритм, так же как алгоритм имитации отжига, является итерационным алгоритмом и предполагает последовательное приближение к оптимальному решению. На каждой итерации в данном алгоритме производится «запуск искусственного муравья». Данный «искусственный муравей» по заданному набору правил производит поиск наилучшего маршрута к «пище» на основе меток феромонов, которые были оставлены предшествующими муравьями. На каждом этапе муравей выполняет последовательность шагов $i = 1,2,3,\dots,n$, и на каждом шаге выбирает требование j для постановки на место i на основе «вероятности перехода» [12, 50].

Для работы алгоритм муравьиной колонии использует следующие параметры:

- η_{ij} – оценка, на основе некой эвристики, которая показывает, как постановка требования j на место i в решении окажется лучше.

- τ_{ij} – «след», который оставляет муравей, корректируемый на каждой итерации алгоритма. Данный параметр определяет, насколько «хорошим» для требования j является позиция i, т.е. данный параметр является накопителем информации о выборе позиции i для требования j.

Работа алгоритма муравьиной колонии идет по следующей схеме [12, 50]:

а) производится вычисление всех оценок η_{ij}

б) производится расчет начального значения «следа» – τ_0

в) для каждого шага $i = 1,2,3,\dots,n$ производится вычисление матрицы вероятностей перехода по следующей формуле (формула 1.6)

$$\rho_{ij} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} * \eta_{ij}^{\beta}}{\sum \tau_{ij}^{\alpha} * \eta_{ij}^{\beta}}, & j \in \Omega \\ 0, & j \notin \Omega \end{cases} \qquad (1.6)$$

где, Ω – множество требований,
α и β – настраиваемые параметры.

г) производится выбор требования j на позицию i на основе следующих параметров:

$$\begin{cases} j = \underset{h \in \Omega}{\operatorname{argmax}} \tau_{ih}^{\alpha} * \eta_{ih}^{\beta}, & q < q_0 \\ j \text{ определяется случайным образом} \\ \text{согласно вероятности } \rho_{ij}, & q \geq q_0 \end{cases} \qquad (1.7)$$

где, $0 \leq q_0 \leq 1$ – параметр алгоритма,
q – случайное число, получаемое на каждом шаге работы.

д) производится определение «локального следа» (формула 1.8)

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\tau_0 \quad (1.8)$$

где, $\rho \in [0,1]$ – коэффициент распада феромона и задается как параметр работы алгоритма.

е) завершающий этап шага итерации – корректировка «глобального следа», которая производится по следующей схеме: если в позиции i выполняется требование j, то значение вычисляется по формуле 1.9, иначе по формуле 1.10.

$$\tau_{ij} = (1-\rho)\tau_{ij} + \frac{\rho}{T^*} \quad (1.9)$$

где T^* – суммарное запаздывание для «лучшего» найденного решения.

$$\tau_{ij} = (1-\rho)\tau_{ij} \quad (1.10)$$

К преимуществам алгоритма муравьиной колонии относятся:
- имеется гарантия сходимости к оптимальному решению;
- имеется возможность его использования в динамических приложениях.

К недостаткам алгоритма муравьиной колонии можно отнести:
- требуется большое число итераций для нахождения решения;
- невозможно определить время сходимости;
- возникают большие трудности для определения значений параметров работы алгоритма, которые, как правило, определяются экспериментальным путем.

1.3.2.3 Нейронные сети. Нейронные сети можно представить как вычислительную систему, выполняющую преобразования различной информации по подобию, как это происходит в мозгу человека [51, 52].

Базовым понятием в нейронных сетях является понятие нейрона, который представляет некую нервную клетку, которая может получать входные сигналы из внешней среды и выдавать сигнал во внешнюю среду, а также передавать и получать сигналы от других нейронов. Поступающие в нейрон сигналы могут возбуждать нейрон, так и могут тормозить его. Суммируя входные сигналы, нейрон принимает решение о пересылке сигнала другим нейронам или во внешнюю среду.

Совокупность искусственных нейронов, соединенных связями передачи сигналов, формируют искусственную нейронную сеть. Для использования таких нейронных сетей требуется их обучение, в результате чего

производится настройка параметров отдельных нейронов, и нейронная сеть приобретает возможность выявлять связи между входными и выходными параметрами сети. Обучение нейронной сети может производиться различными методам, а именно:

- обучение с учителем
- обучение без учителя;
- обучение на примерах;
- смешанное обучения.

Построение нейронной сети может иметь различную архитектуру в зависимости от тех задач, которые данная сеть должна решать. Наиболее часто используются такие архитектурные решения для нейронных сетей как:

- перцептрон,
- сверточные нейросети,
- глубокие нейросети,
- нейросеть Хопфилда,
- самоорганизующаяся карта Кохонена,
- нейросеть Кохонена и др.

Использование нейросетей для решения задачи составления расписаний занятий требует разработки и проведения обучения нейронной сети на основе заданных параметров. Применение нейронных сетей в области решения задач составления расписаний имеет серьезные ограничения связанных с их обучением. Однако, нейронные сети могут эффективно применяться совместно с другими методами и алгоритмами составления расписаний, например, в качестве некоторых эвристик для которых имеются сложности в их формализации [51, 52].

1.4 Генетический алгоритм

Генетический алгоритм представляет собой один из самых эффективных эвристических методов и алгоритмов, который чаще всего используется для решения задачи составления расписаний учебных занятий. Данный алгоритм имеет ряд преимуществ при применении его в качестве алгоритма составления расписания перед другими методами и алгоритмами, а именно:

- алгоритм работает с кодами представляющих набор параметров решаемой задачи, которые является аргументами целевой функции, применяемой в алгоритме;
- алгоритм оперирует всей совокупной областью возможных решений задачи;
- работа алгоритма не требует наличие какой-либо дополнительной информации, что позволяет значительно ускорить процесс поиска;
- для нахождения новых точек поиск алгоритм может использовать различные вероятностные и детерминированные правила.

1.4.1 Основные понятия, применяемые в генетических алгоритмах. Для описания какого-либо генетического алгоритма применяются понятия, которые были заимствованы из генетики.

В генетических алгоритмах заимствованы из генетики и могут использоваться такие понятия как [4, 7, 14, 33, 52, 53]:

- Популяция – это некоторое конечное число особей.

- Особь (организм) – это некий элемент, входящий в состав некой популяции. Особь состоит из набора хромосом (одной или более), в котором кодируются различные параметры задачи, так же называемыми точкам в пространстве поиска.

- Хромосома (цепочки, кодовые последовательности) – представляет собой упорядоченный набор генов.

- Ген (свойство, знак, детектор) – представляет некоторый атомарный элемент внутри хромосомы.

- Генотип (структура) – представляет собой набор хромосом для особи. Как было сказано ранее, генотип может содержать одну или несколько хромосом.

- Фенотип – это набор значений, которые соответствуют определенному генотипу, т.е. фенотип представляет некое множество параметров решаемой задачи.

- Аллель (значение свойства, вариант свойства) – это некое конкретное значение, которое может иметь некоторый ген.

- Локус (позиция) – это место размещения некоторого гена в хромосоме.

- Поколение – это популяция, над которой производится работа генетического алгоритма. Поколение, которое поступает на начало итерационного шага генетического алгоритма, называется «поколением потомков», а поколение, получаемое в конце итерационного шага алгоритма, называется «новым поколением».

1.4.2 Функция приспособляемости (fitness function). Одной из ведущих ролей в работе генетического алгоритма принадлежит функции приспособляемости (fitness function). Название данной функции, так же, как и другие понятия и термины, было заимствовано из генетики.

Функция приспособляемости в генетическом алгоритме используется для оценки меры приспособляемости отдельной особи в популяции, в которую она входит. Полученная с помощью данной функции оценка приспособляемости конкретных особей популяции позволяет произвести отбор особей в «новое поколение» в соответствии с эволюционным принципом «выживает сильнейший».

Функция приспособляемости генетического алгоритма оказывает существенное влияние на работу алгоритма на всех этапах его работы, поэтому данная функция должна иметь достаточно четкое и корректное определение. Работа генетического алгоритма сводится, как правило, к максимизации значения функции приспособляемости. Поэтому функцию приспособляемости часто называют целевой функцией. Если же требуется

провести минимизацию функции приспособляемости, то, как правило, производят специальные преобразования данной функции с целью перевода обратно на максимизацию целевой функции.

1.4.3 Схема работы генетического алгоритма. В классическом варианте работа генетического алгоритма состоит из последовательности шагов [16, 34, 36, 54, 55-61]:

а) Формирование начальной популяции. На данном шаге создается начальная популяция, т.е. заданное число особей с заданным набором хромосом. На данном шаге могут использоваться различные методы и алгоритмы, выбор и применение которых зависит от поставленной задачи и исходных данных. Данный шаг во многом будет определять итоговый результат работы алгоритма, поэтому конечный результат данного этапа должен формировать популяцию, которая наиболее близка к оптимальному решению, иначе конечный результат работы генетического алгоритма может оказаться неудовлетворительным.

б) Применение генетических операторов для создания «новой популяции». На данном шаге могут применяться один или два генетических оператора:

- Оператор скрещивания (crossover). Данный генетический оператор применяется к двум особям из популяции, которые выбираются по некоторому набору правил. У выбранных особей производится обмен отдельных генов по заранее определенным правилам.

- Оператор мутации (mutation). Данный генетический оператор применяется к одной выбранной особи из популяции, которая выбирается по заданным правилам. У выбранной особи производится изменение ряда генов в хромосомах на новые допустимые значения.

в) Селекция особей на основе функции приспособляемости. На данном шаге алгоритма для каждой особи вычисляется значение функции приспособляемости. На основе вычисленных оценок целевой функции производится отсев «нежизнеспособных» особей.

г) Формирование нового поколения. После применения генетических операторов (оператора скрещения и/или оператора мутации) и применения селекции особей формируется «новое поколение».

д) Проверка условия завершения алгоритма. После формирования «нового поколения» для него производится проверка условия завершения поиска решения. Если условие завершения работы итерационный части генетического алгоритма не выполняется, то происходит переход на начало итерационного цикла алгоритма – применение генетических операторов. В противном случае происходит переход на завершающий шаг работы генетического алгоритма.

е) Выбор «наилучшей» особи. Данный шаг является завершающим этапом работы генетического алгоритма. На данном шаге на основе значения целевой функции определяется «наилучшая» особь в последнем поколении. Полученная «наилучшая» особь будет считаться итоговым решением задачи.

Графическая схема работы генетического алгоритма представлена на рисунке 1.1 [62-64].

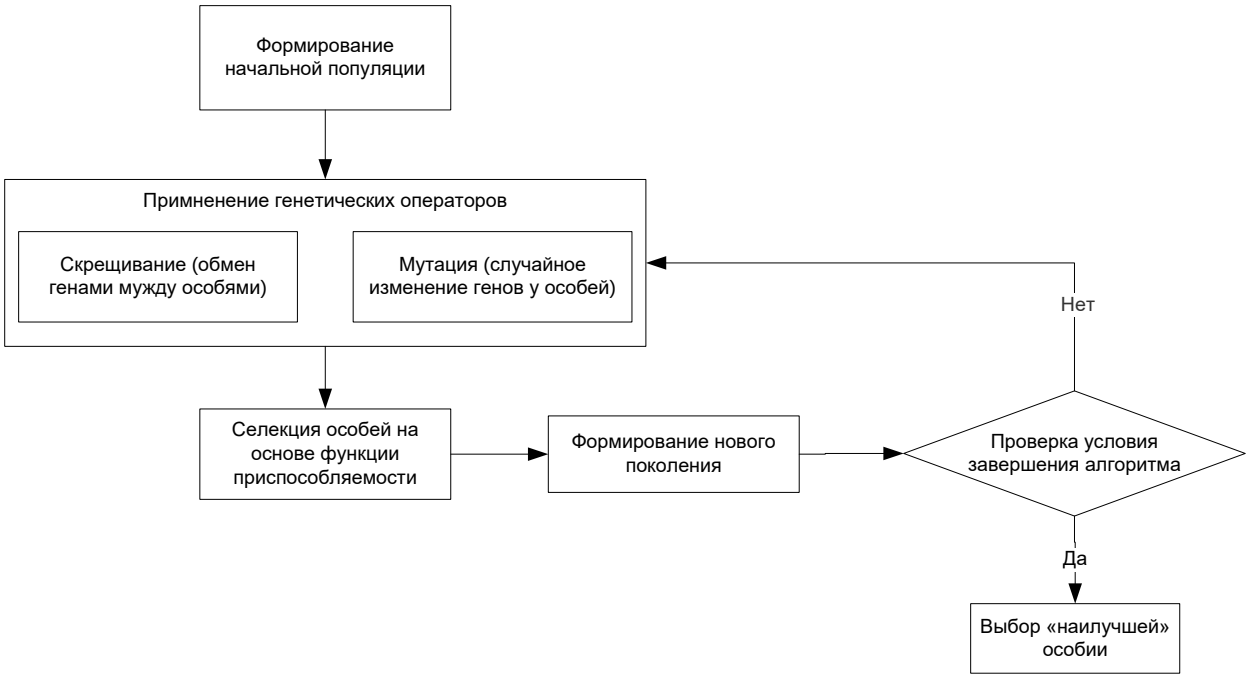


Рисунок 1.1 – Схема работы генетического алгоритма

1.4.4 Применение генетического алгоритма для реализации задачи составления расписаний занятий. Генетический алгоритм довольно часто применяется для решения задачи составления расписания учебных занятий. Имеется достаточно много примеров применения различных модификаций генетического алгоритма для данной цели.

Применение генетического алгоритма для реализации задачи составления расписания учебных занятий в большинстве реализаций предполагает следующие модификации [4, 14, 34, 53]:

- гены в хромосоме представляют различные переменные, которые требуются для составления расписания учебных занятий (преподаватели, учебные группы, дисциплины и другие переменные);
- особь может содержать одну хромосому либо несколько хромосом, которые содержат группы связанных параметров;
- особь представляет собой возможный вариант расписания учебных занятий;
- популяция представляют собой набор возможных вариантов расписаний учебных занятий;
- функция приспособляемости оперирует, как правило, на базе некоторого набора ограничений;
- для формирования новых вариантов расписаний могут использоваться оба генетических оператора;

- задаются правила для проведения селекции различных вариантов расписаний для создания нового поколения;

- определяются правила для проверки условия завершения поиска и выбора «наилучшей» особи для определения оптимального расписания учебных занятий.

Модификация генетического алгоритма для решения задачи составления расписаний учебных занятий во многом зависит от исходных данных задачи и от тех требований и ограничений, которые накладываются на расписание учебных занятий.

1.5 Выводы по первому разделу

Задача составления расписания учебных занятий в высших учебных заведениях является одной из важнейших задач в системе организации полноценного учебного процесса. Задача составления расписания относится к классу NP-полных задач, что определяет сложность и трудоёмкость её решения.

Для решения задачи составления расписания занятий могут использоваться разнообразные методы и алгоритмы. Данные методы и алгоритмы можно условно объединить в две большие группы:

- Точные методы и алгоритмы. Данная группа предполагает полный перебор и оценку возможных вариантов расписания для определения оптимального варианта расписания. Такой полный перебор всех возможных вариантов решения при большой размерности решаемой задачи может привести к значительным временным и вычислительным затратам. Для снижения данных затрат часто используются различные методы и приемы для исключения явно неприемлемых вариантов решения с целью сокращения области поиска и как следствие снижение затрат на решение задачи. К точным методам и алгоритмам относятся: методы ветвей и границ, методы отсечения, методы теории графов и др.

- Приближенные методы и алгоритмы. Данная группа направлена на получение приемлемого расписания учебных занятий близкого к оптимальному расписанию. Поиск решения в данных методах и алгоритмах строится на понижении требований к искомому расписанию и применение разнообразных эвристик. Применение данных методов и алгоритмов позволяет получить приемлемый вариант расписания за приемлемое время. Данная группа включает в себя такие методы и алгоритмы, как метод муравьиной колонии, алгоритм имитации отжига, методы, основанные на ограничениях, нейронные сети и генетические алгоритмы.

По результатам рассмотренных в данном разделе тем были опубликованы тезисы в сборниках 2-х конференций [63, 65], 2 статьи в научных журналах РК [62, 64].

2. Формализация задачи и построение математической модели для составления расписания учебных занятий в высших учебных заведениях

В первой части данного научного исследования мы рассмотрели подходы, которые применяются для решения задачи составления расписаний учебных занятий. Исходя из рассмотренных подходов, для решения задачи составления расписания занятий необходимо решить следующие проблемы [66]:

а) определить исходные данные, которые используются для составления расписаний занятий;

б) определить выходные данные задачи, т.е. представление расписания в результате решения задачи;

в) определить математическую модель, на основе которой будет производиться решение задачи;

г) определить набор ограничений, которые накладываются на расписание занятий;

д) применить какой-либо метод или алгоритм для решения задачи на основе определенной математической модели и ограничений.

Решение указанных выше проблем необходимо рассмотреть те бизнес-процессы, которые имеют место при организации учебного процесса в ВУЗе.

Данное научное исследование проводится на базе Восточно-Казахстанского технического университета им.Д.Серикбаева, поэтому мы рассмотрим, как производится планирование учебного процесса на примере данного ВУЗа.

2.1 Организация планирования учебного процесса в ВУЗе

Учебный процесс в ВУЗах строится на использовании различных учебно-методических рекомендации и документах. В разных ВУЗах указанные рекомендации и документы могут иметь различный состав и отражать различные аспекты учебной деятельности того или иного учебного заведения.

Несмотря на указанное разнообразие учебно-методических документов, которые применяются для организации учебного процесса, имеется ряд документов, которые используются практически во всех учебных заведениях. Данные документы и рекомендаций могут иметь различные наименования, поэтому при рассмотрении таких документов и рекомендаций следует обращать внимание на их содержание, а не на их наименования.

Для составления расписаний учебных занятий используются следующие документы и рекомендации [10, 54, 64, 67-76]:

а) Учебный план по образовательной программе. Данный документ представляет основной документ, который применяется при планировании учебного процесса по какой-либо образовательной программе. Содержание учебного плана может включать два вида компонентов: обязательный

компонент и элективный компонент. Типичный учебный план, как правило, включает в себя следующие элементы:

- Перечень изучаемых компонентов: учебные дисциплины, профессиональные практики, а также другие виды учебных компонентов.
- Объем учебной нагрузки по каждому компоненту учебного плана в часах и кредитах.
- Распределение компонентов по учебным периодам (семестры, триместры, четверти и др.)
- Распределение часов по видам занятий по каждому компоненту учебного плана: лекции, семинары, практические и лабораторные работы, самостоятельная работа обучающегося и прочие виду занятий.
- Формы текущего, промежуточного и итогового контроля по каждому компоненту учебной программы.

б) Индивидуальный учебный план обучающегося. Данный учебный план составляется для каждого обучающегося на базе учебного плана образовательной программы и содержит перечень учебных компонентов, которые должен изучить обучающийся за весь период своего обучения в учебном заведении. Индивидуальный учебный план включает перечень обязательных компонентов и элективную часть, которая включает выбранные обучающимся элективные компоненты из учебного плана образовательной программы.

в) Каталог элективных дисциплин. Данный документ составляется на основе учебного плана образовательной программы и включает в себя перечень элективных дисциплин данной учебной программе. Для каждой дисциплины в данном документе приводится краткое описание, требования и цели изучения. Цель составления каталога элективных дисциплин – помощь обучающимся в выборе элективных дисциплин для включения их в свой индивидуальный план.

г) Рабочий учебный план. Данный документ составляется на учебный год и включает в себя перечень учебных дисциплин, практик и других видов учебной работы по конкретной образовательной программе. Данный документ является основным документом для организации учебного процесса по образовательной программе в течение учебного года.

д) Индивидуальный план преподавателя. Данный документ содержит информацию об учебной нагрузке преподавателя на учебный год. Данный индивидуальный план включает в себя информацию о распределении учебной нагрузке преподавателя в разрезе учебных периодов (учебный год, семестр, триместр, квартал и др.), также в разрезе видов учебной и иных видов работ (лекции, практические занятия, семинары и др.).

е) Рабочий график. Данный документ содержит сведения о дисциплинах, изучаемых в течение некоторого академического периода с указанием учебных групп, академических потоках и преподавателей. В рабочем графике указываются различные требования и пожелания к проведению учебных занятий: требуемые аудитории, время проведения занятий для учебной

группы и/или потока, время проведения занятий для преподавателя и другие требования. Данный документ предоставляется в диспетчерскую службу учебного заведения и является основой для составления расписания учебных занятий на весь академический период.

ж) Расписание учебных занятий. Данный документ содержит сведения об учебных занятиях, которые должны проводиться в течение академического периода (семестр, триместра, квартала и др.). Для каждого занятия в данном документе указываются время проведения, аудитория, академическая групп или поток, преподаватель. Содержание данного документа может быть представлено в различных разрезах – учебных групп, аудиторий или преподавателей. Расписание учебных занятий составляется в высших учебных заведениях, как правило, на календарную неделю. Данное расписание на календарную неделю может действовать весь академический период либо может действовать на определенные недели учебного периода, например, четные и нечетные недели, и др.

з) Расписание экзаменационной сессии. Данный документ содержит сведения о дате и времени проведения экзаменов и иных видов учебных занятий (консультаций, защита курсовых работ и др.), которые проводятся после завершения обучения в академическом периоде. Так же, как и для расписания учебных занятий, для каждого экзамена и иного вида занятий в период сессии указываются аудитория, академическая группа или поток и преподаватель.

Кроме перечисленных выше документов в ВУЗах могут использоваться и применяться и иные виды документов, которые используются для организации и планирования учебного процесса.

Как видно из приведенного списка документов, расписания в высших учебных заведениях делятся на 2 вида: расписание учебных занятий на академический период и расписание экзаменационной сессии. В данном научном исследовании мы будем рассматривать решение задачи для составления обоих видов расписаний.

2.2 Постановка задачи составления расписания учебных занятий

Исходя из рассмотренных в первой главе данного научного исследования подходов к реализации решения задачи составления расписания учебных занятий на первом этапе решения необходимо выполнить следующие задачи:

а) определить исходные данные, которые требуются для решения задачи составления расписания учебных занятий;

б) определить выходные данные, которые должны быть получены как результат решения задачи составления расписания учебных занятий;

в) определить условия, которые накладываются на итоговое решение задачи.

2.2.1 Исходные данные для задачи. В первом разделе данной главы мы рассмотрели какие документы используются для планирования учебного процесса в ВУЗе. Поскольку данное научное исследование проводится на базе ВКТУ им.Д.Серибкаева, то нами были изучены данный набор документов, которые используются при составлении расписания учебных занятий. На основе данного изучения нами были определены следующие переменные: [10, 40, 54, 77-84]:

а) $A \in \{A_1, A_2, \dots, A_a\}$ – множество аудиторий, где будут проводиться учебные занятия.

б) $P \in \{P_1, P_2, \dots, P_p\}$ – множество преподавателей, которые будут проводить учебные занятия.

в) $D \in \{D_1, D_2, \dots, D_d\}$ – множество дисциплин, по которым планируются проведение занятий.

г) $G \in \{G_1, G_2, \dots, G_g\}$ – множество учебных групп, для которых должны проводиться занятия.

д) $S \in \{S_1, S_2, \dots, S_s\}$ – множество подгрупп, на которые может быть разбита учебная группа.

е) $W \in \{W_1, W_2, \dots, W_w\}$ – множество дней (дни недели или календарные дни), в течение которых должны будут проводиться учебные занятия.

ж) $T \in \{T_1, T_2, \dots, T_t\}$ – множество временных интервалов в течение дня, в которые планируется проведение учебных занятий.

з) $K \in (K_1, K_2, \dots, K_k)$ – множество видов занятий (лекции, практики, семинары и др.) по которым планируется проведения занятий.

и) $F \in \{F_1, F_2, \dots, F_f\}$ – множество академических потоков.

Рассмотрим более подробно выделенные нами объекты исходных данных, которые используются для составления расписания занятий.

2.2.1.1 Объект «аудитория». Как известно учебные занятия должны проводиться в аудиториях, которые оснащены требуемым оборудованием и имеют достаточную вместимость. Аудитория может представлять собой как помещение в некотором здании, так и виртуальное место для проведения онлайн занятий.

Для удобства составления расписаний аудиторный фонд, как правило, может быть классифицирован по различным критериям. Наиболее часто аудиторный фонд классифицируется по обеспеченности аудитории, тем или иным оборудованием, например:

- Учебные аудитории – аудитории, которые оснащены стандартными посадочными местам и классными досками (меловые, маркерные, интерактивные доски или проекторы) и предназначенные для проведения обычных занятий (семинарские, практические и иные виды занятий).

- Лекционные аудитории – аудитории большой вместимости для проведения лекционных занятий для больших академических потоков.

- Компьютерные аудитории – аудитории, которые оборудованы компьютерной техникой, на которой должны проводиться какие-либо

учебные работы (графические и проектные работы, изучение работы с каким-либо программным обеспечением и др.).

- Лабораторные аудитории – аудитории, которые оборудованы специальным лабораторным оборудованием и предназначенные для проведения лабораторных занятий на нем.

- Специализированные аудитории – аудитории, которые предназначены для проведения специальных видов занятий.

- Виртуальные аудитории – аудитории, которые предназначены для проведения различных видов онлайн занятий при дистанционном обучении. Вместимость аудитории данного типа представляет собой максимально возможное число пользовательских подключений, которое определяется пропускной способностью серверного и коммуникационного оборудования.

Объект «аудитория» имеет следующий базовый набор атрибутов:

а) Номер аудитории. Данный номер, как правило, состоит из 2-х частей:

- номер корпуса, где расположена аудитория
- номер аудитории внутри корпуса

б) Вместимость аудитории. Вместимость показывает максимальное число обучающихся, которые могут поместиться в аудитории.

в) Тип аудитории, согласно классификации, которую мы рассмотрели ранее.

Помимо описанных выше атрибутов, для аудитории могут использоваться дополнительные атрибуты, если они необходимы для составления расписания занятий.

2.2.1.2 Объект «преподаватель». Проведение учебных занятий осуществляются профессорско-преподавательским составом в соответствии с распределением учебной нагрузки. В нашей модели данный педагогический состав представлен в виде множества объектов «преподаватель».

Объект «преподаватель» имеет следующий набор атрибутов:

а) Идентификатор преподавателя

б) Фамилия, имя и отчество преподавателя

Кроме описанных выше атрибутов, для объекта «преподаватель» могут использоваться дополнительные атрибуты, если они необходимы для составления расписания занятий.

2.2.1.3 Объект «дисциплина». Учебных занятия проводятся по определенным дисциплинам. Перечень дисциплин, по которым проводятся учебные занятия, определяется на основе учебных планов образовательных программ и их распределением по учебным периодам. Поскольку расписание учебных занятий составляется на определенный академический период (семестр, триместр и др.), то перечень учебных дисциплин определяется на основе рабочих учебных планов образовательных программ на данных академический период.

Проведение занятий по определенной дисциплине по определенным видам занятий требует наличие аудиторного фонда с заданными характеристиками и наличием определенного оборудования. Вследствие

этого, для каждой дисциплины и вида занятий необходимо определить зависимости какие аудитории подходят для проведения занятий.

Объект «дисциплина» имеет следующий набор атрибутов:

а) Идентификатор дисциплины

б) Наименование дисциплины

в) Привязка аудитории по типу занятия и приоритет выбора

- привязка конкретной аудитории
- привязка определенного типа аудитории
- привязка преподавателя к аудитории и др.

Помимо описанных выше атрибутов, для дисциплины могут использоваться дополнительные атрибуты, если они необходимы.

2.2.1.4 Объект «учебная группа». Обучающие в высших учебных заведениях объединяются в академические группы по набору таких критериев, как:

- образовательная программа;
- язык обучения;
- форма обучения;
- курс обучения;
- максимальное число обучающихся в учебной группе.

Расписание учебных занятий, как правило, составляется по академическим группам, так как учебная группа является базовой единицей для распределения учебной нагрузки в разрезе преподавателей, учебных дисциплин и учебных занятий.

Объект «учебная группа» имеет следующий набор атрибутов:

а) Идентификатор группы

б) Код группы

в) Образовательная программа

г) Курс обучения

Помимо описанных выше атрибутов, для учебной группы могут использоваться дополнительные атрибуты, если они необходимы.

2.2.1.5 Объект «подгруппа». Проведение учебных занятий по некоторым заданным дисциплинам и видам занятий могут иметь ограничения на максимальное количество обучающихся. Если количество обучающихся в учебной группе или учебном потоке окажется больше данного ограничения, то группа или учебный поток должен быть разбит на несколько подгрупп, так, чтобы количество обучающихся в каждой подгруппе не превышало заданное ограничение на максимальное количество обучающихся в подгруппе.

Проведение занятий в подгруппах могут проводиться:

- параллельно, т.е. занятия во всех подгруппах проводятся в один и тот же день и одно и то же время в разных аудиториях с разными преподавателями

- произвольно, т.е. занятия в подгруппах проводится разные дни и время, а аудитории и преподаватели могут быть как разными, так и одинаковыми.

Подгруппа характеризуется только порядковым номером подгруппы.

2.2.1.6 Объект «день». Проведение учебных занятий проводится в определенные дни. В качестве объекта «день» могут выступать следующие виды дней:

а) Календарные дни. Календарные дни представляют набор календарных дней в некотором диапазоне, в которые должны проводиться учебные занятия. В высших учебных заведениях такой тип объектов «день» используется для составления расписания экзаменационной сессии.

б) Дни недели. Дни недели представляют собой дни недели, по которым проводятся учебные занятия. Расписание в данном случае составляется на весь академический период с разбивкой на учебные недели. Проведение учебных занятий по дням недели могут проводиться с различной частотой:

- еженедельно, когда занятия проводятся каждую неделю
- четная/нечетная неделя, когда занятия проводятся только четной или нечетной недели
- другие виды частот (раз в 2 недели, раз в 3 недели, и др.).

Построение расписание с применением одновременно нескольких частот требует учитывать пересечение объектов «дней», например, если применяется построение расписание с еженедельной частой и частотой четная/нечетная неделя.

2.2.1.7 Объект «вид занятия». Проведение учебных занятий по дисциплине могут иметь различные виды. К таким видам, как правило, относятся следующие виды занятий:

- лекционные занятия;
- практические занятия;
- семинарские занятия;
- лабораторные занятия и др.

Объект «вид занятия» имеет следующие атрибуты:

а) Идентификатор вида занятий;

б) Наименование вида занятий.

В зависимости от вида занятий, определяется, в какой аудитории может проводиться занятие по той или иной дисциплине.

2.2.1.8 Объект «временной интервал». Одно учебное занятие, как правило, длится заданное количество минут, после чего обучающимся предоставляется время для отдыха и/или перехода в другую аудиторию для проведения другого учебного занятия. Указанное время между занятиями может иметь разные периоды. Для составления расписания в учебных заведениях составляется временная сетка, в которой указываются периоды проведения учебных занятий и перерывов между ними в течение дня.

В нашей модели мы будем использовать только временные интервалы для учебных занятий, без временных интервалов перемен между занятиями.

Объект «временной интервал» имеет следующие атрибуты:

а) Идентификатор временного интервала

б) Время начала занятия

в) Время завершения занятия

г) Порядковый номер занятия

Построение временной сетки для занятий, как правило, исключают пересечение времени проведения занятий. Однако могут использоваться временные сетки, в которых время занятий могут пересекаться. В этом случае необходимо учитывать данное обстоятельство при составлении расписания учебных занятий.

2.2.1.9 Объект «учебный поток». Как было сказано выше, расписание учебных занятий, как правило, составляется для академической группы. Однако, для проведения учебных занятий по одной дисциплине у одного преподавателя, академические группы могут объединяться, при условии того, что разрешенное количество обучающихся для данного вида занятия по данной дисциплине не превышает заданное максимальное количество. Такое объединение называется академический поток.

В нашем исследовании мы будем использовать понятие «академический поток» для описания возможного занятия, которое должно быть включено в учебное расписание, т.е. поток может включать как одну академическую группу, так и объединение нескольких групп.

Исходя из выше сказанного, академический поток (F_i) в нашем случае представляет собой следующий кортеж (формула 2.1):

$$F_i = (\{P\}, K, h, \{(G, S, D, k, m)\}) \quad (2.1)$$

Как видно из формулы 2.1, каждый академический поток представляет собой набор из следующих исходных переменных:

а) Множество преподавателей $\{P\}$. Данное множество включает в себя множество преподавателей, которые будут вести занятие. Как правило, данное множество состоит только из одного преподавателя, однако, могут иметься случаи, когда одно занятие могут одновременно проводить два и более преподавателей.

б) Вид занятия K.

в) Количество временных интервалов для занятия h. Проведение учебного занятия может занимать несколько подряд идущих временных интервалов.

г) Множество кортежей, которые представляют из себя набор значений исходных таких переменных как:

- учебная группа G;
- подгруппа S;
- дисциплина D;
- количество обучающихся k;
- битовая маска обучающихся академической группы G, которая задает, какие обучающие из группы включены в данный поток.

Данное определение академического потока позволяет описать возможное учебное занятие, которое должно быть включено в искомое расписание учебных занятий.

2.2.2 Выходные данные задачи. На основе выделенных нами выше исходных переменных для задачи составления расписания учебных занятий решение задачи будет заключаться в нахождении следующего множества $R = \{R_1, R_2, \dots, R_f\}$, где R_i представляет собой элемент расписания. Каждый элемент расписания определяется как кортеж из набора следующих исходных переменных (формула 2.2):

$$R_i = (F_i, A, W, \{T_1, \dots, T_h\}) \quad (2.2)$$

Как видно из приведенного определения выходных данных для задачи составления расписания учебных занятий, мы для каждого академического потока должны определить:

- аудиторию для проведения занятия;
- день проведения занятия;
- временные интервалы для занятия (количество временных интервалов определяется исходя из значения параметра h соответствующего академического потока).

Определив указанные выше значения переменных для каждого R_i мы получим одно из множества возможных решений задачи составления расписания. Поскольку возможных решений задачи составления расписания может быть достаточно много, и многие из них могут быть неприемлемыми или неоптимальными, то нам необходимо далее определить ограничения на возможные решения и критерии оптимальности решения.

2.2.3 Ограничения для задачи составления расписания учебных занятий. Как было сказано в первой главе, на решение задачи составления расписанию могут накладываться различные ограничения, которые могут быть классифицированы различными способами. В данном диссертационном исследовании мы будет использовать разделение ограничений на следующие группы [85, 86]:

- «жесткие» ограничения. Данная группа включает ограничения, которые являются обязательными для итогового решения задачи.
- «мягкие» ограничения. Данная группа включает ограничения, которые являются рекомендательными для итогового решения задачи.

Отнесение того или иного ограничения к «жестким» или «мягким» должно определяться исходя из логики и устанавливаемым требованиям к решению задачи и на основе экспертной оценки [87].

Кроме того, разбиение ограничений может быть и на большее число групп с разной значимость [88], однако в данном исследовании мы будем придерживаться классификации с 2 группам ограничений и регулировать их значимость на основе весовых коэффициентов для каждого отдельного ограничения.

В данном диссертационном исследовании мы будет использовать следующие ограничения для итогового расписания занятий [47, 70, 76, 77, 81, 83-85, 89, 105-106]:

а) Отсутствие накладок для аудитории, т.е. в определенной аудитории в одновременно может проводиться только занятие только для одного академического потока. Для некоторых аудиторий данное ограничение может не действовать, например:

- занятия по физической культуре проводятся на спортивных комплексах в которым могут проводиться занятия разных потоков.

- онлайн занятия, которые проводятся в виртуальном пространстве и др.

В таком случае для аудитории проверяется только её вместимость, т.е. сколько одновременно обучающихся может вместить аудитория.

б) Учет вместимости аудитории, т.е. в аудитории одновременно не может находиться большее число обучающихся, чем вместимость аудитории.

в) Учет приемлемости аудитории для занятия, т.е. учебные занятия должны проводиться только в тех аудиториях, которые имеют необходимое обеспечение для данного вида занятий.

г) Отсутствие накладок для группы/подгруппы/обучающегося, т.е. группа/подгруппа/обучающийся не может одновременно находиться на занятиях в разных потоках.

д) Отсутствие накладок для преподавателя, т.е. преподаватель не может одновременно вести занятия в разных потоках.

е) Минимизация «окон» в расписании группы/обучающегося, т.е. занятия учебный группы/обучающегося в течение дня должны идти компактно, по возможности без разрывов в заданных временных интервалах.

ж) Минимизация «окон» в расписании преподавателя, т. е. занятия, проводимые преподавателем, в течение дня должны идти компактно, по возможности без разрывов в заданных временных интервалах.

з) Ограничение на количество занятий в течение дня у группы/обучающегося. У группы/обучающегося в течение дня должно быть не более 8 часов занятий.

и) Ограничение на количество занятий в течение дня у преподавателя. У преподавателя в течение дня должно быть не более 8 часов занятий.

к) Ограничение на повторение определенного вида занятий у группы/обучающегося в течение дня, т.е. в течении дня не должны повторяться занятия одного вида (лекции, практики и др.) по дисциплине у группы/обучающегося.

л) Возможность исключения определенных дней и/или временных интервалов из расписания группы, например, для проведения занятий на военной кафедре, прохождения стажировки и др.

м) Возможность исключения определенных дней и/или временных интервалов из расписания преподавателя, если преподаватель не может проводить занятия в данные дни и время.

н) Возможность исключения определенных дней и/или временных интервалов из расписания для аудитории, если данная аудитория по каким-либо причинам не может быть задействована для проведения занятий в указанные дни и время.

Требования можно расширять ещё, но перечисленных выше требований вполне достаточно для составления расписания учебных занятий. Также необходимо учитывать, что увеличение числа ограничений может увеличить трудоемкость поиск решения задачи [54, 90].

2.3 Применение генетического алгоритма для поиска решения задачи составления расписания учебных занятий

После определения математической модели расписанию учебных занятий, которое мы рассмотрели в предыдущем разделе, необходимо определить механизм поиска оптимального решения задачи.

В первой главе данного научного исследования мы рассмотрели какие методы и алгоритмы применяются для решения рассматриваемой нами задачи составления расписания учебных занятий. Данные методы и алгоритмы условно разделяются на точные методы и приближенные методы.

Задача составления расписания занятий в высших учебных заведениях имеет большую размерность и к нему могут применяться различные требования, которые часто сложно формализовать. Вследствие данного обстоятельства применение точных методов в нашем случае достаточно затруднительно, поэтому в данном исследовании мы будем использовать один из приближенных методов.

Одним из часто применяемых методов для решения задачи составления расписаний из группы приближенных методов является генетический алгоритм. Данный алгоритм обладает рядом преимуществ при решении задач составления расписаний, которые мы рассмотрели в разделе 1.4.

Исходя из вышесказанного, нами было принято решение использовать в данном научном исследовании генетический алгоритм для реализации поиска решения задачи составления расписания учебных занятий на основе предложенной в разделе 2.2 математической модели расписания.

Как было нами рассмотрено в первой главе данной работы, для использования генетического алгоритма для поиска решения задачи необходимо решить следующие основные задачи:

а) Определить функцию приспособляемости (fitness function).

б) Определить состав генов и хромосом для особи.

в) Определить работу алгоритма на различных стадиях:

- генерация начальной популяции;
- применение генетических операторов мутации и/или скрещивания;
- правило выборки особей и формирования нового поколения;
- правило остановки поиска;
- правило выбора «наилучшей» особи в последнем поколении.

Решение описанных выше задач позволит применить генетический алгоритм для решения поставленной нами задачи составления расписания.

2.3.1 Функция приспособляемости. Функция приспособляемости (fitness function), как было указано в первой главе, является центральным элементом любой реализации генетического алгоритма и оказывает огромное влияние на его работу.

В данном научном исследовании мы определили математическую модель расписания на основе ограничений. Определенные нами ограничения имеют разный приоритет при составлении расписания, поэтому для оценки получаемых решений мы будем использовать отдельные весовые коэффициенты для оценки выполнения каждого ограничения. Данные весовые коэффициенты позволят определять более приемлемые решения задачи по важности ограничений [79-81, 91-94].

Для функции приспособляемости в данном исследовании мы определили следующие параметры:

а) Ограничения, входящие в определяемую функцию, условно делятся на: «жесткие» и «мягкие» ограничения. Разделение ограничений на перечисленные виды задается до начала работы алгоритма на основе предпочтений. Данные предпочтения могут быть выявлены экспериментальным путем или на основе какой-либо экспертной оценки.

б) Выполнение каждого ограничения оценивается дробным числом в диапазоне от 0 (ограничение полностью не выполнено) до 1 (ограничение полностью выполнено).

в) Для каждого ограничения задается весовой коэффициент. Величина весового коэффициента задается положительным произвольным числом. Данная величина определяется изначально задается экспертным путем, а затем при анализе полученных результатов работы алгоритма, может быть скорректирована.

г) Поиск оптимального решения задачи в нашем случае будет достигаться путем максимизации значения функции приспособляемости в следующей последовательности:

- вначале по сумме взвешенных оценок «жестких» ограничений;
- затем по сумме взвешенных оценок «мягких» ограничений.

Исходя из вышеперечисленных требований, функция приспособляемости в данном исследовании будет иметь следующий вид (формула 2.3):

$$f(a) = \left(\sum_{i=1}^{n} k_i h_i(a) \to \max, \sum_{j=1}^{m} l_j \, s_j(a) \to \max \right) \qquad (2.3)$$

где,

a – особь, для которой вычисляется значение целевой функции,

h_i – функция для вычисления оценки выполнения i-го «жесткого» ограничения для особи a,

k_i – весовой коэффициент для i-го «жесткого» ограничения,

s_j – функция для вычисления оценки выполнения j-го «мягкого» ограничения для особи a,

l_i – весовой коэффициент для j-го «мягкого» ограничения,

n – количество «жестких» ограничений,

m – количество «мягких» ограничений.

Предложенная выше функция приспособляемости будет использоваться на всех этапах работы генетического алгоритма.

2.3.2 Определение генома и хромосом особи. Следующей задачей, которую нужно решить для разработки генетического алгоритма – это определение генома и хромосом особи, которая, в нашем случае, будет представлять возможный вариант расписания занятий.

В данном исследовании, исходя из предложенной математической модели, мы будем использовать следующие параметры для особи:

а) особь будет состоять только из одной хромосомы.

б) хромосома будет включать в себя последовательность из n-генов, где n – это число занятий, которые должны быть включены в итоговое расписание.

в) расположение генов в хромосоме у каждой особи должно быть одинаковым, т.е. порядковый номер гена в хромосоме для разных особей соответствует одном и тому же занятию в расписании.

г) каждый ген в хромосоме будет представлять учебное занятие в расписании и будет содержать следующие информационные элементы о нем:

- F (поток) – сведения об учебном потоке для занятия. Данная часть содержит данные некоторого объекта «учебный поток», который мы рассмотрели в разделе 2.2.1.9. Данная часть гена является неизменяемой и передается потомкам особи без каких-либо изменений.

- A (аудитория) – сведения об аудитории для занятия. Данная часть содержит сведения об объекте «аудитория» (раздел 2.2.1.1) из множества аудиторий, который был назначен для проведения занятия. Данный элемент гена является изменяемым.

- W (день) – сведения о дне проведения занятия. Данная часть содержит сведения об объекте «день» (раздел 2.2.1.6) из множества дней, определенных для расписания. Данный элемент гена является изменяемым.

- T (временные интервалы) – сведения о временных интервалах, которые были назначены для занятия. Данная часть содержит список объектов «временной интервал» (раздел 2.2.1.8) из множества временных интервалов. Количество объектов «временной интервал» в данной части гена определяется из количества часов, определенных для объекта «учебный поток». Данный элемент гена является изменяемым.

- E (дефективность гена) – сведения о нарушении ограничений в гене. В данный элемент представляет собой список нарушений, которые были

выявлены при проверке расписания на выполнение ограничений. Формирование списка нарушений должно формироваться в функции приспособляемости (формула 2.3) при вычислении значений функций ограничений для «жестких» (h_i) и «мягких» (s_j) ограничений.

Графическое представление хромосомной и генной структуры особи для нашей модификации генетического алгоритма представлено на рисунке 2.1.

Исходя из определенной нами хромосомной и генной структуры особи для нашей задачи, её решение будет заключаться в подборе допустимых значений для элементов гена A, W и T с учетом ограничений, которые накладываются на расписание.

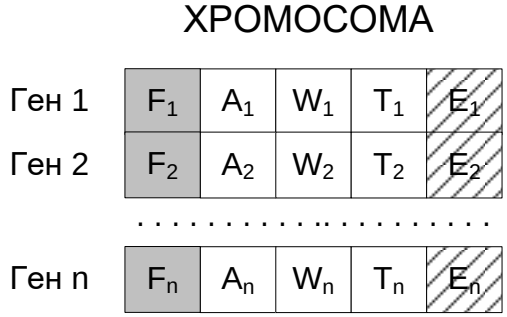


Рисунок 2.1 Хромосомная и генная структура особи

2.3.3 Определение шагов генетического алгоритма. После того, как мы определили функцию приспособляемости, так же хромосомную и генную структуру особи для решения нашей задачи, нам необходимо определить поведение генетического алгоритма на разных этапах его работы. Для этого необходимо решить следующие задачи:

а) определить правила генерации начальной популяции.

б) определить работу генетических операторов скрещивания и мутации.

в) определить правила селекции особей и формирования нового поколения.

г) определить условие завершения работы алгоритма и определения «наилучшей» особи из последнего поколения.

Далее рассмотрим решение поставленных выше задач.

2.3.3.1 Генерация начальной популяции. Первым шагом генетического алгоритма является формирование начальной популяции для последующих шагов работы алгоритма. Данный этап является достаточно важным этапом работы, т.к. от качества начальной популяции во многом будет зависеть качество итогового решения задачи.

В данном научном исследовании нами был предложен следующий алгоритм формирования начальной популяции:

а) На вход подается упорядоченный список учебных потоков, которые должны быть включены в расписание. Упорядочивание учебных потоков может проводиться на основе различных критериев: по количеству

обучающихся в потоке, по преподавателям, по дисциплинам и др. Поскольку учебные потоки часто имеют некоторые связи между собой (по группам, преподавателям и аудиторному фонду), то в данном исследовании мы будем упорядочивать учебные потоки по следующему алгоритму:

- объединение учебных потоков в кластеры нужно производить на основе их семантической близости.
- упорядочивание учебных потоков внутри кластера производится на основе количества обучающихся в потоках в обратном порядке.
- упорядочивание кластеров производится на основе количества обучающихся в учебных потоках данного кластера в обратном порядке.

В данном исследовании семантическая близость учебных потоков будет определяться по следующий формуле (формула 2.4) [95-97, 109]:

$$SI(f_x f_y) = \sum_{i=1}^{n} w_i SI_i \qquad (2.4)$$

где, SI_i – мера i-связи между двумя учебными потоками,
w_i – весовой коэффициент для i-связи

Для определения семантической близости сервисов использованы следующие меры связей и весовые коэффициенты для них (Таблица 2.1):

Таблица 2.1 – Меры связей и весовые коэффициенты

Обозначение	**Описание меры связи**	**Весовой коэффициент**
SI_G	Отношение кардинального множества групп, которые имеются в обоих потоков к кардинальному числу множества групп, которые имеются в обоих потоках	0,8
SI_T	Отношение кардинального множества ППС, которые имеются в обоих потоков к кардинальному числу множества ППС, которые имеются в обоих потоках	0,1
SI_A	Отношение кардинального множества аудиторий, которые имеются в обоих потоков к кардинальному числу множества аудиторий, которые имеются в обоих потоках	0,1

Поскольку учебные потоки, которые подаются на вход, могут иметь разные качества, то количество кластеров, на которые можно распределить учебные потоки, достаточно сложно. Поэтому в данном исследовании мы для кластеризации учебных потоков мы будем использовать алгоритм кластеризации FOREL, который сам определяет число возможных кластеров.

Работа алгоритма кластеризации FOREL будет идти по следующей схеме:

1) Считываются данные по учебным потокам

2) Для каждой пары потоков определяется их сематическая близость на основе формулы 2.4.

3) Из списка потоков выбирается случайный поток.

4) Для выбранного потока из списка потоков выбираются близкие потоки с радиусом поиска R. В данном исследовании мы будем использовать радиус поиска $R = 0,5$. Данные радиус соответствует половине максимального расстояния между учебными потоками.

5) Вычисляем расстояния между выбранными учебными потоками и находим поток, который является центром кластера.

6) Если полученный центр кластера совпадает с выбранным учебным потоком, то, значит, мы нашли кластер. Если кластер найден, то добавляем кластер в список кластеров и исключаем включенные в него учебные потоки из рассмотрения. В противном случае полученный центр кластера становиться выбранным потоком и повторяем действия, начиная с пункта 4.

7) После нахождения очередного кластера, проводим проверку, имеются ли нераспределенные по кластерам учебные потоки. Если нераспределенные потоки имеются, то повторяем действия, начиная с пункта 3, иначе работа алгоритма завершается, и мы получаем список кластеров.

Графическая схема алгоритма кластеризации учебных потоков на основе алгоритма FOREL представлена на рисунке 2.2.

После получения кластеров производится упорядочивание кластеров и учебных потоках в них, как было описано выше. В результате мы получаем упорядоченный набор учебных потоков.

б) Также на вход генетического алгоритма подается объем начальной популяции. Данный объем должен быть достаточным, чтобы на выходе алгоритма было получено приемлемое решение задачи. Объем начальной популяции не должен быть малым, поскольку в данном случае снижается вероятность получения желаемого результата решения задачи, а также он не должен быть слишком большим, так как в данном случае увеличивается объем вычислений, времени и требований к вычислительным ресурсам для получения решения задачи. Рекомендуемый объем начальной популяции зависит от объем входных данных и варьируется в диапазоне от 10 до 20 особей.

в) После получения упорядоченного списка потоков и определения объема начальной популяции производится генерация особей начальной популяции. Генерация каждой особи в начальной популяции производится по следующей схеме:

1) Выбирается поток из головы упорядоченного списка потоков.

2) Выбранный поток включается как ген в хромосому особи.

3) Для созданного гена производится перебор возможных вариантов значений переменных гена: день, временной интервал, аудитория. Для

каждого набора значений переменных гена определяется значение функции приспособляемости (формула 2.3). Из возможных вариантов включения гена в хромосому выбирается значение переменных, которые дают оптимальное значение для целевой функции как описано в разделе 2.3.1.

г) После окончания перебора списка учебных потоков мы получаем сформированную особь, которая представляет собой некоторый вариант расписания учебных занятий.

Схема генерации новой особи представлено на рисунке 2.4.

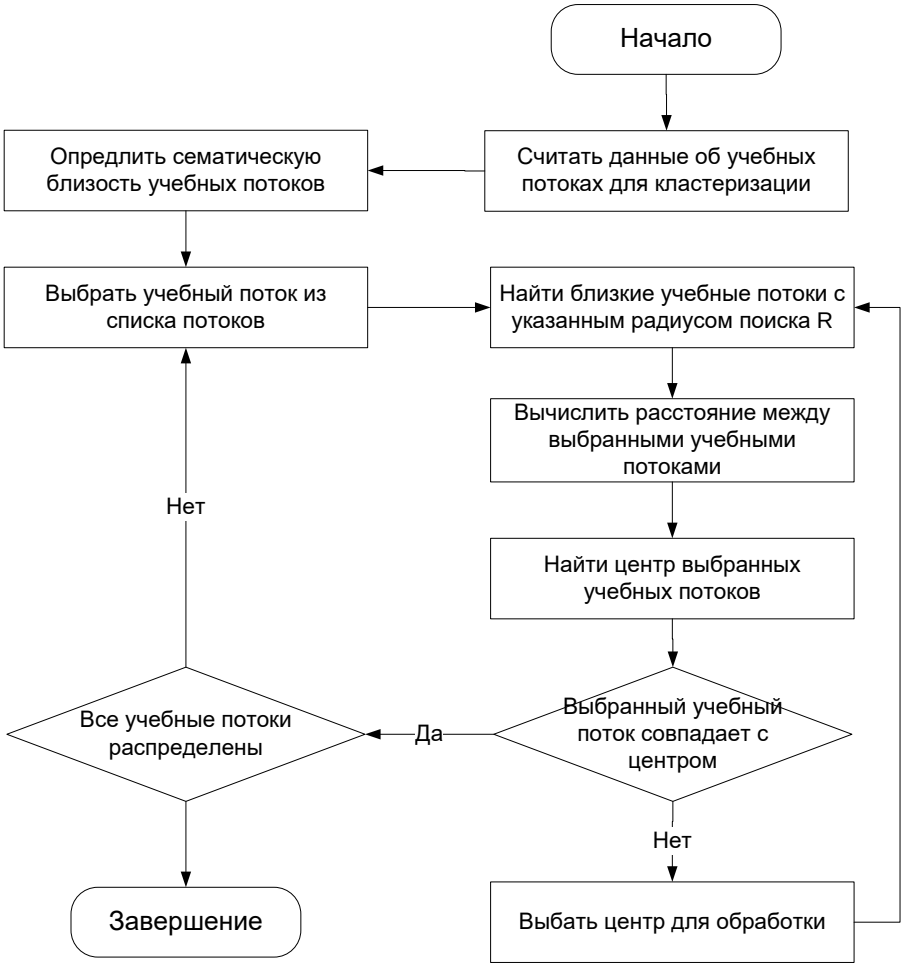


Рисунок 2.2 – Алгоритм FOREL для кластеризации потоков

В итоге мы получаем набор особей начальной популяции, который будет передан на следующий шаг генетического алгоритма.

2.3.3.2 Генетические операторы. В циклической части генетического алгоритма, как было указано в первой главе, могут применяться два вида генетических оператора – скрещивания и мутации. При работе генетического алгоритма может использоваться сразу оба оператора, так и только один из них. Если используются оба оператора, то порядок их применения определяется на этапе проектирования. В данном научном исследовании в нашей модификации генетического алгоритма мы будем использовать оба оператора в следующей последовательности:

- сначала к текущей популяции будет применен оператор мутации;
- затем к полученной популяции будет применен оператор скрещивания.

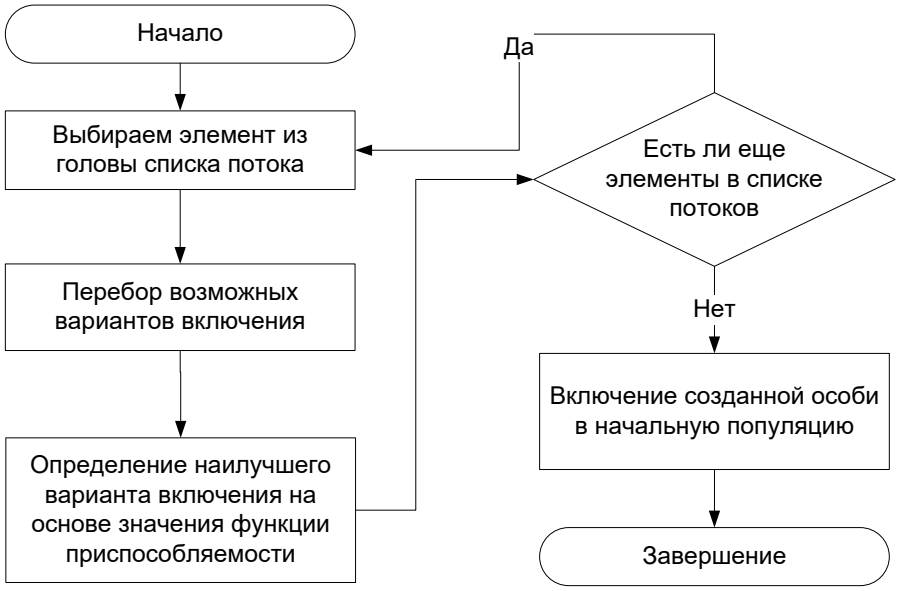


Рисунок 2.3 – Схема формирования новой особи

Рассмотрим, как будут реализованы данные два генетических оператора в реализации нашего генетического алгоритма.

2.3.3.2.1 Оператор мутации. Данный оператор применяется к отдельной особи из популяции. Применение данного генетического оператора в данном исследовании будет выполняться следующим образом:

а) из текущей популяции случайным образом выбирается некоторый набор особей и для каждой из них создается её копия;

б) для каждой созданной особи производиться замена изменяемых частей (день, временной интервал, аудитория) в некоторых генах на другие возможные варианты;

в) полученный набор модифицированных особей включается в популяцию.

Как видно из описания предложенного оператора мутации, его основная работа производится на 2-м этапе, когда производится замена значений в некоторых генах. Реализация выборки генов и замена в них значений могут проводиться различными способами. В данном исследовании мы будем использовать следующие варианты:

а) Модификация некоторого числа генов, которые нарушают некоторое выбранное ограничение, накладываемое на расписание. Данный вариант предполагает следующую последовательность действий:

1) Из набора ограничений, которые накладываются на расписание занятий, случайным образом выбирается одно ограничение.

2) В хромосоме в случайном порядке выбираются гены, которые нарушают выбранное на первом шаге ограничение. Число выбираемых генов

задается до начала работы алгоритма. Рекомендуется выбирать небольшое число генов (от 20 до 100 генов в зависимости от общего числа генов в хромосоме), чтобы не увеличивать время работы алгоритма.

3) Для выбранных генов очищаем значения переменных частей: день, временные интервалы и аудитория.

4) После этого производится последовательный проход по каждому выбранному гену и на каждом проходе выполняются следующие действия:

- для гена производится перебор возможных вариантов значений переменных гена: день, временной интервал, аудитория.

- для каждого набора значений переменных гена определяется значение функции приспособляемости (формула 2.3).

- из возможных вариантов выбирается значение переменных, которые дают оптимальное значение для целевой функции как описано в разделе 2.3.1.

4) Полученная новая модифицированная особь включается в популяцию.

Графическая схема работы данной разновидности оператора мутации приведено на рисунке 2.4.

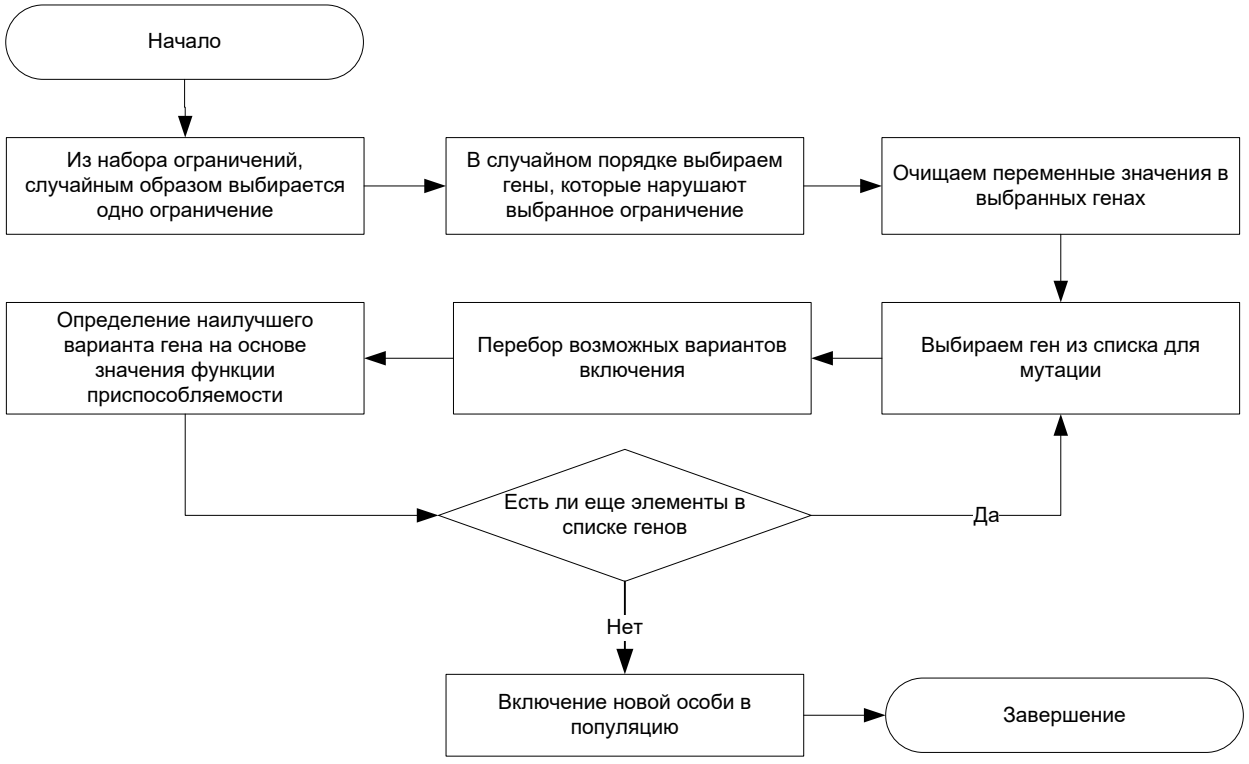


Рисунок 2.4 – Схема работы оператора мутации по модификации генов, которые нарушают некоторое ограничение

б) Попарная модификация некоторого числа генов. Данный вариант оператора модификации предполагает следующую последовательность действий:

1) В хромосоме в случайном порядке выбираются гены. Число выбираемых генов задается до начала работы алгоритма. Рекомендуется выбирать небольшое число генов (от 20 до 100 генов в зависимости от общего числа генов в хромосоме). Поскольку в работе данного варианта модификации применяется попарная модификация генов, то число выбираемых для модификации генов должно быть четным числом.

2) Для выбранных генов очищаем значения переменных частей: день, временные интервалы и аудитория.

3) Далее производится проход по списку генов, выбранных в 1-м шаге. На каждом шаге данного прохода выполняются следующие действия:

- из головы списка выбираются 2 гена,
- для 1-го гена производится перебор возможных вариантов значений переменных гена: день, временной интервал, аудитория,
- для каждого набора значений переменных 1-го гена определяется значение функции приспособляемости (формула 2.3).
- из возможных вариантов значений переменных для 1-го гена выбираются значения, которые дают оптимальное значение для целевой функции как описано в разделе 2.3.1.
- повторяем тот же поиск значений переменных для 2-го гена,
- сохраняем в буфере полученные значения переменных обоих генов и итоговое значение целевой функции,
- заново очищаем значения переменных частей 1-го и 2-го гена,
- повторяем поиск значений переменных частей в обоих генах в обратном порядке: сначала поиск идет для 2-го гена, а затем для 1-го гена,
- сравниваем полученное значение целевой функции с сохраненным ранее в буфере значением целевой функции и если сохраненное значение является более оптимальным, то восстанавливаем сохраненные в буфере предыдущие значения переменных частей обоих генов.

4) Полученная новая модифицированная особь включается в популяцию.

Графическая схема работы данной разновидности оператора мутации приведено на рисунке 2.5.

в) Модификация набора связанных генов по учебной группе, аудитории и преподавателю. Данный вариант работы оператора модификации предполагает следующую последовательность действий:

1) Задается число выбираемых генов до начала работы алгоритма. Рекомендуется выбирать небольшое число генов (не более 10–20% от общего числа генов).

2) Случайным образом выбирается какой-либо ген в хромосоме и добавляется в список генов для модификации.

3) Для выбранного гена выбираем связанные гены, которые добавляются в список для модификации. Связанные гены определяются по наличию одинаковых значений по аудитории, группы или преподавателю. По какому значению будут выбираться связанные гены, определяется случайным образом для каждого выбранного на 2-м шаге гена.

4) Если размер списка выбранных генов для модификации меньше, чем заданное в 1-м шаге число, то повторяем выбор генов, начиная с шага 2.

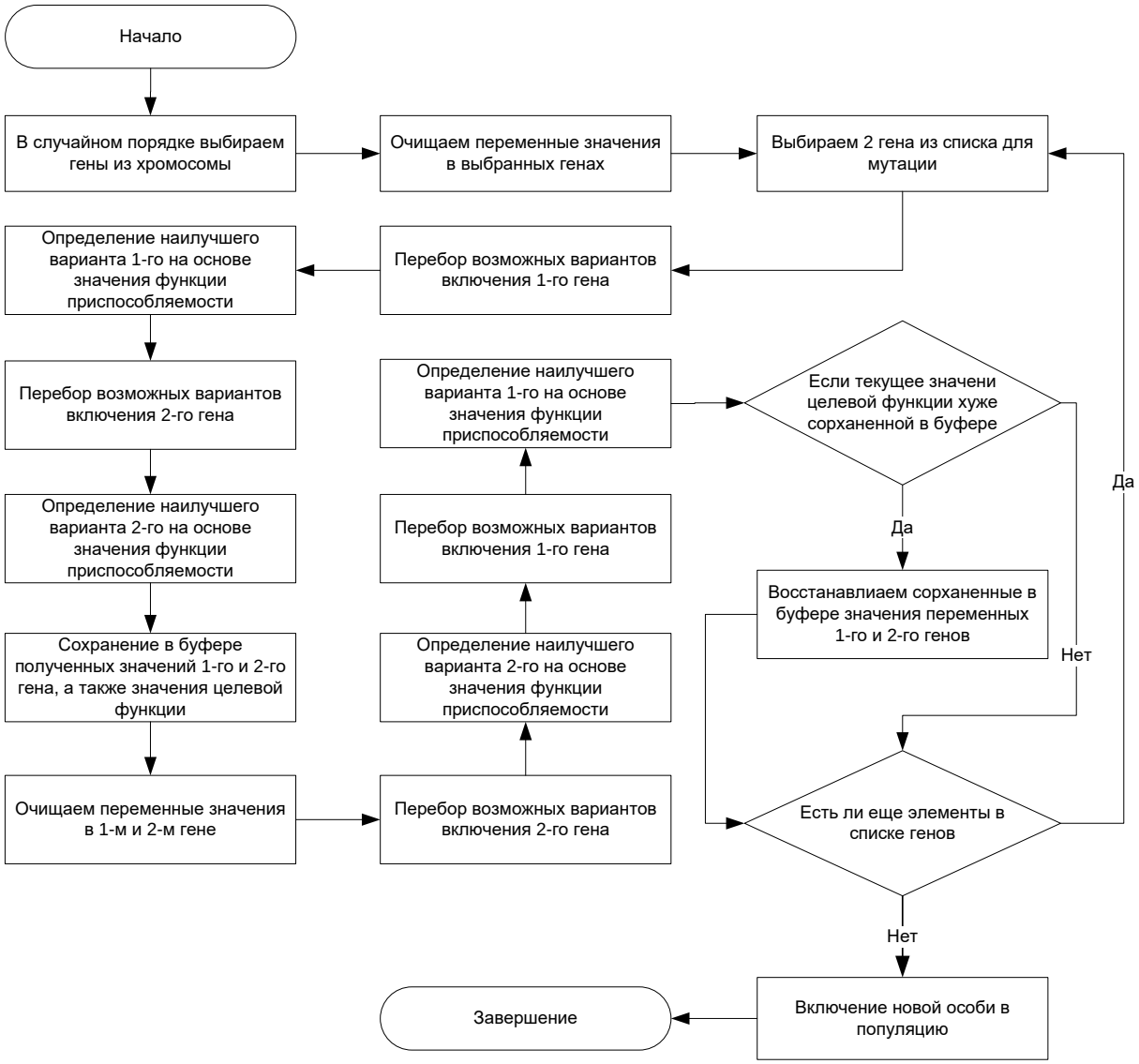


Рисунок 2.5 - Схема работы оператора мутации попарной модификации генов.

5) Для выбранных генов очищаем значения переменных частей: день, временные интервалы и аудитория.

6) После этого производится последовательный проход по каждому выбранному гену и на каждом проходе выполняются следующие действия:

- для гена производится перебор возможных вариантов значений переменных гена: день, временной интервал, аудитория.

- для каждого набора значений переменных гена определяется значение функции приспособляемости (формула 2.3).

- из возможных вариантов выбирается значение переменных, которые дают оптимальное значение для целевой функции как описано в разделе 2.3.1.

7) Полученная новая модифицированная особь включается в популяцию.

Графическая схема работы данной разновидности оператора мутации приведено на рисунке 2.6.

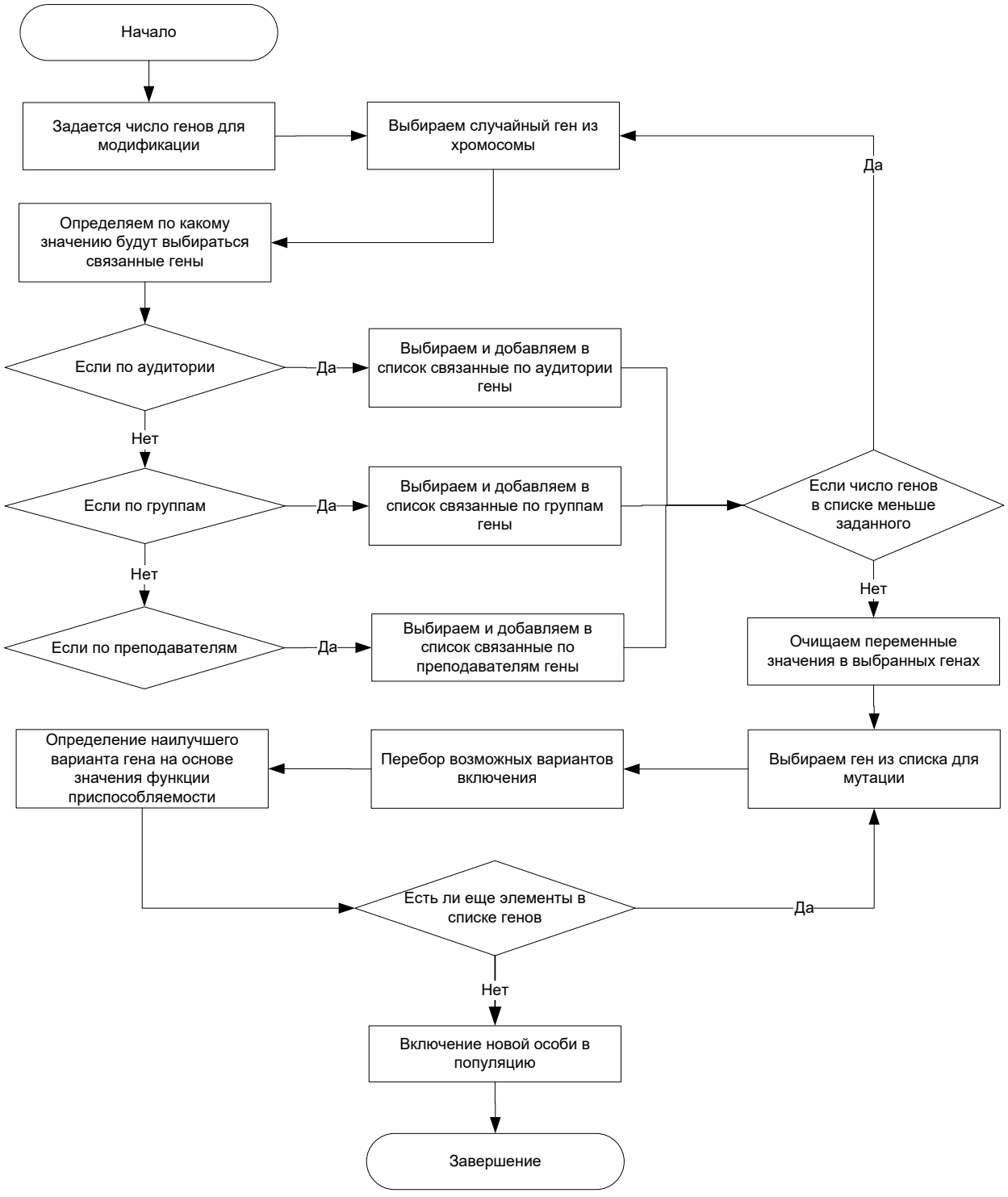


Рисунок 2.6 - Схема работы оператора мутации набора связанных генов по учебной группе, аудитории и преподавателю.

г) Модификация дефективного гена и связанных с ним генов по учебной группе, аудитории и преподавателю. Данный вариант работы оператора мутации предполагает следующую последовательность действий:

1) Из хромосомы случайным образом выбирается какой-либо дефективный ген и добавляется в список генов для модификации.

2) Для выбранного гена выбираем связанные гены, которые добавляются в список для модификации. Связанные гены определяются по наличию одинаковых значений по аудитории, группы или преподавателю.

3) Для выбранных генов очищаем значения переменных частей: день, временные интервалы и аудитория.

4) После этого производится последовательный проход по каждому выбранному гену и на каждом проходе выполняются следующие действия:

- для гена производится перебор возможных вариантов значений переменных гена: день, временной интервал, аудитория.

- для каждого набора значений переменных гена определяется значение функции приспособляемости (формула 2.3).

- из возможных вариантов выбирается значение переменных, которые дают оптимальное значение для целевой функции как описано в разделе 2.3.1.

5) Полученная новая модифицированная особь включается в популяцию.

Графическая схема работы данной разновидности оператора мутации приведено на рисунке 2.7.

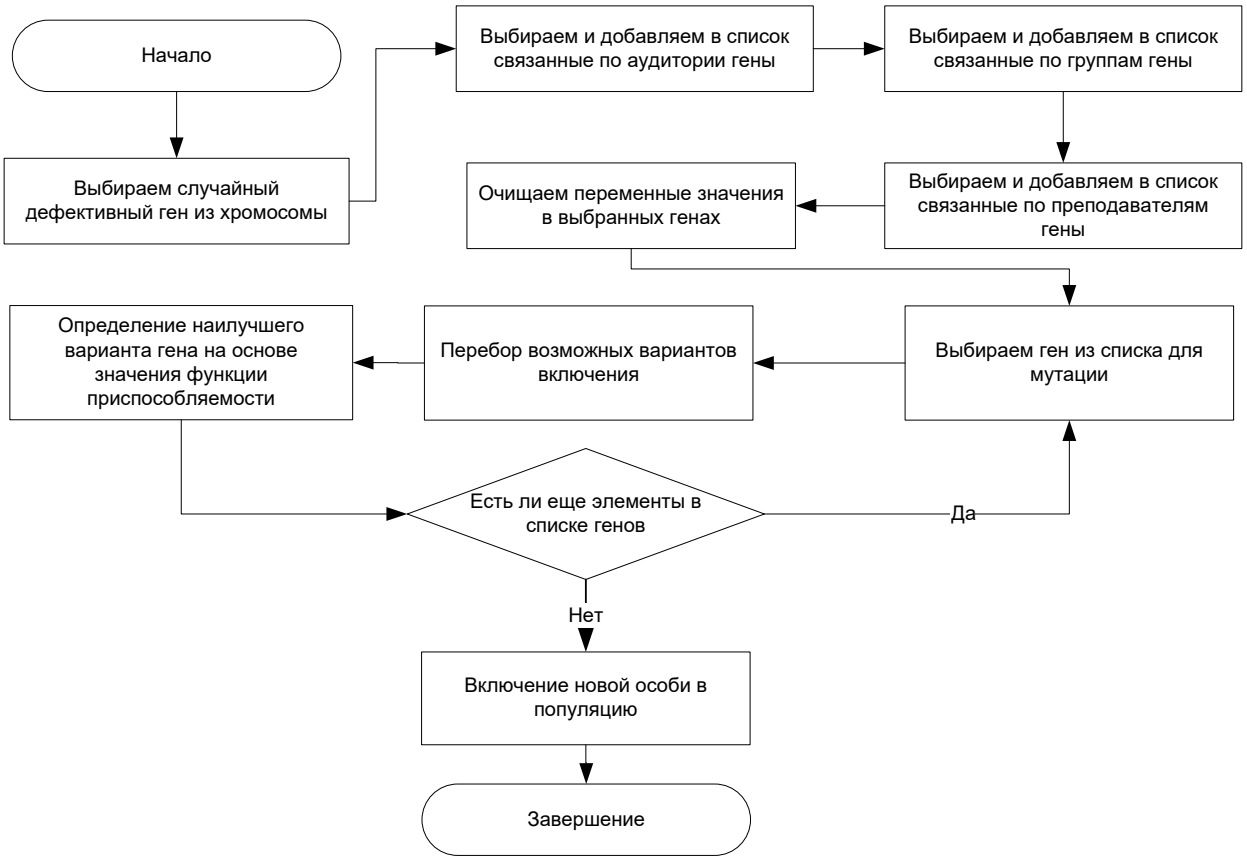


Рисунок 2.7 - Схема работы оператора мутации модификации дефективного гена и связанных с ним генов по учебной группе, аудитории и преподавателю.

Применение описанного выше варианта работы оператора мутации в модификации генетического алгоритма в данном исследовании будем определяться случайным образом для каждой выбранной для мутации особи из популяции.

2.3.3.2.2 Оператор скрещивания. Другим генетическим оператором, который может использоваться при работе генетического алгоритма, является оператор скрещивания. Применение данного оператора в генетическом алгоритме предполагает выполнение следующих действий:

а) из текущей популяции выбираются 2 особи;

б) формируется новая особь с тем же набором хромосом и генов;

в) отдельные значения в хромосомах и генах наследуются от 1-й или 2-й особи;

г) в результате применения оператора скрещивания формируется новая особь, которая добавляется в популяцию.

В данном исследовании мы будем использовать оператор скрещивания, который работает по следующей схеме:

а) из популяции случайным образом выбирается 1-я особь;

б) создается полная копия выбранной особи;

в) из популяции случайным образом выбирается 2-я особь;

г) определяется число генов, у которых могут быть заменены переменные части у созданной особи на значения из генов 2-й особи в пределах от 10% до 20% от общего числа генов;

д) случайным образом из 2-й особи выбирается определенное на предыдущем шаге число генов для переноса их значений в созданную особь;

е) производится проход по выбранным генам 2-й особи и на каждом проходе выполняются следующие действия:

- значения переменной части текущего гена из списка копируются в соответствующий ген в новой особи.

- производится оценка значения функции приспособляемости (формула 2.3) для созданной особи с новыми значениями в гене;

- если значение целевой функции имеет более оптимальное значение, то данные изменения фиксируются, иначе, восстанавливаются исходные значения гена из соответствующего гена 1-й особи;

ж) после завершения прохода по списку генов новая особь включается в текущее поколение.

Графическая схема работы предложенного в данном исследовании генетического оператора скрещивания приведена на рисунке 2.8.

При применении генетических операторов может возникнуть ситуация, когда полученная новая особь будет иметь идентичные значения переменных для всех генов с какой-либо другой особью, которая имеется в популяции. Наличие подобных идентичных особей в популяции может привести к проблемам в работе генетического алгоритма, поэтому при включении новой особи в популяцию необходимо производить проверку на наличие в

популяции идентичной особи, и если идентичная особь имеется в популяции, то новая особь не должна включаться в популяцию.

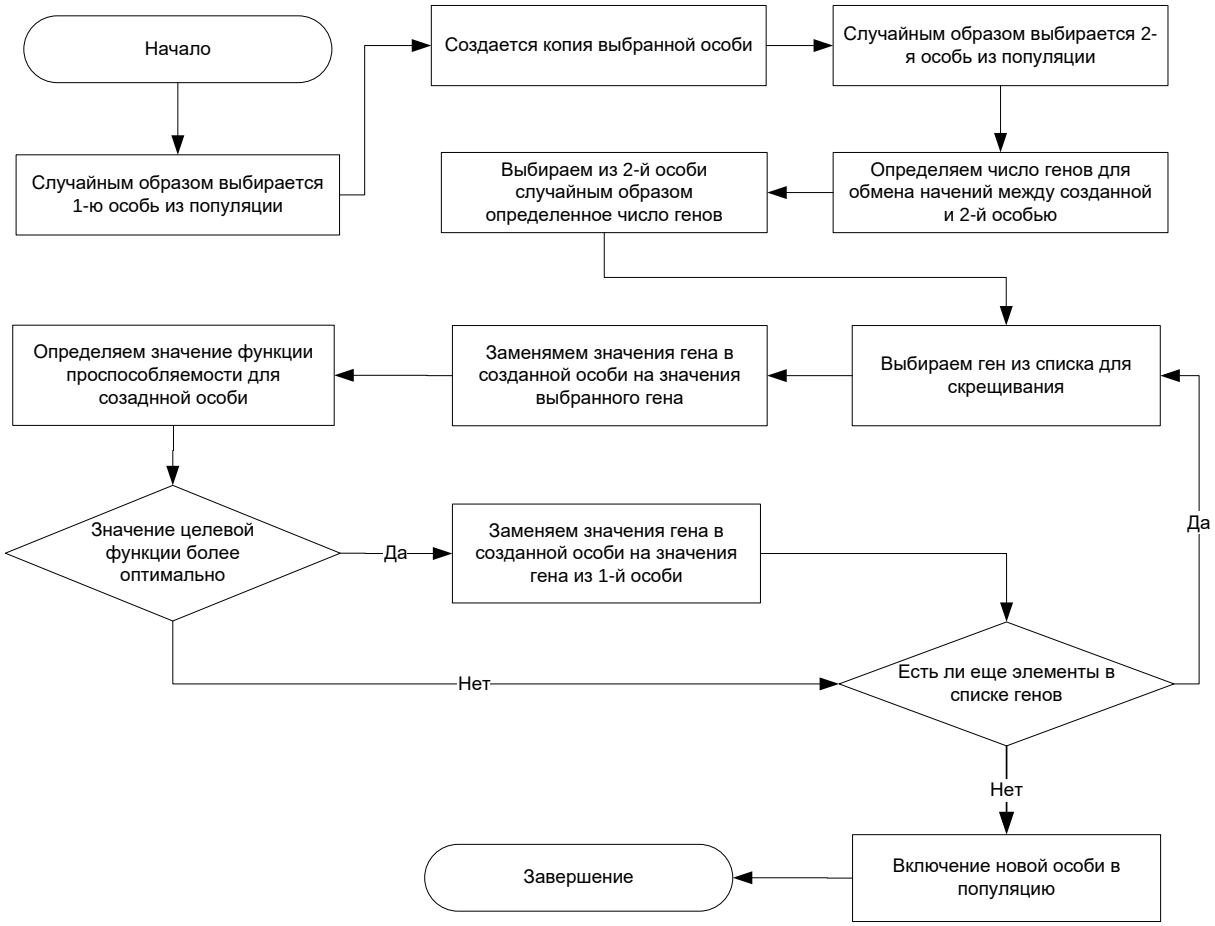


Рисунок 2.8 – Схема работы оператора скрещивания

В результате применения описанных выше генетических операторов мутации и скрещивания будет сформирована новая популяция, которая будет передана на следующий шаг работы генетического алгоритма.

2.3.3.3 Селекция особей и формирование нового поколения. Следующим шагом генетического алгоритма после применения генетических операторов является шаг по селекции особей из полученной популяции. Работа алгоритма на данном шаге основывается на таком биологическом принципе как «выживает сильнейший».

В данном исследовании работа на данном этапе генетического алгоритма будет идти по следующей схеме:

а) Выбираем из полученной после применения генетических операторов новой популяции особь с самым оптимальным значением функции приспособляемости в соответствии с правилом оптимальности, которое было описано в разделе 2.3.1. Если таких особей несколько, то выбирается та, у которой меньшее число дефективных генов. Даная особь включается в новое поколение.

б) Делам проход по списку ограничений, которые накладываются на итоговое решение, и для каждого ограничения:

- определяем особи, у которых значение функции оценки ограничения h или s (формула 2.3) имеет большее значение, чем у выбранной в особи с самым оптимальным значением целевой функции.

- если такие особи имеются, то случайным образом выбираем одну из них и включаем её в новое поколение.

в) Из оставшихся особей случайным образом выбираются $m = n - s - 1$-особей, где n – размер популяции для селекции, а s – число особей, который были выбраны на 2-м шаге отбора. Число n является входным значением генетического алгоритма, как правило, больше или равное числу особей в начальной популяции.

г) Отобранные особи формируют новое поколение популяции, а особи, которые не были включены в новое поколение, исключаются из дальнейшего рассмотрения.

Полученное на данном шаге новое поколение становиться текущей популяцией и передается на следующий шаг генетического алгоритма.

2.3.3.4 Условие завершения работы алгоритма и выбор «наилучшей» особи. Последним шагом в циклической части генетического алгоритма является проверка условия завершения поиска решения. Для этого необходимо определить правило или их набор на основе которых будет проводиться проверка условия завершения работы.

В исследовании для данного шага работы генетического алгоритма будет использоваться следующий упорядоченный набор правил, которые должны применяться последовательно, одна за другим:

а) Значение функции приспособляемости (формула 2.3) достигло максимально возможного значения f_{max} хотя бы для одной особи в популяции. Максимально возможно значение для целевой функции, исходя из её определения, равна сумме значений весовых коэффициентов для «жестких» и «мягких» ограничений (формула 2.5)

$$f_{max} = \left(\sum_{i=1}^{n} k_i , \sum_{j=1}^{m} l_j \right) \tag{2.5}$$

где, k_i – весовой коэффициент для i-го «жесткого» ограничения,
l_i – весовой коэффициент для j-го «мягкого» ограничения,
n – количество «жестких» ограничений,
m – количество «мягких» ограничений.

б) Максимальное значение функции приспособляемости (формула 2.3) в gen-поколениях перестало изменяться, где gen – входной параметр алгоритма. Для реализации данного правила мы для каждой особи должны вести учет числа поколений, в которые она входила.

Если условие завершения работы алгоритма не выполнено, то управление передается на следующий цикл поиска, т.е. на шаг применения генетических операторов.

Если условие завершения работы алгоритма выполняется, то цикл поиска решения завершается и выполняется завершающий этап – выбор «наилучшей» особи из последнего поколения. На данном шаге из поколения выбираются:

а) особи, имеющие самое максимальное значение функции приспособляемости, согласно правилам, определенным в разделе 2.3.1.

б) из выбранных особей выбираются особи, имеющие минимальное число дефектов в генах.

в) из полученных особей случайным образом выбирается одна особь, которая является искомым решением.

Решения, получаемые с использованием генетического алгоритма, могут давать разные результаты на одних и тех же входных данных, поэтому для получения более качественных результатов можно производить поиск решения с использованием генетического алгоритма несколько раз с целью получения более подходящего решения задачи.

На этом рассмотрение предлагаемой модификации генетического алгоритма для решения задачи составления расписания занятий завершено. Перейдем к рассмотрению того, как в данном исследовании будет производиться оценка выполнения ограничений, которые являются частью предложенной функции приспособляемости (формула 2.3).

2.4 Оценка выполнения ограничений

Исходя из определения функции приспособляемости для генетического алгоритма (раздел 2.3.1) для оценки каждого ограничения используется штрафные функции, которые возвращает десятичную дробь в диапазоне от 0 (ограничение полностью не выполнено) до 1 (ограничение полностью выполнено) [77-80].

Для удобства реализации указанных функций для оценки отдельных ограничений в данном исследовании предлагаются общие правила их построения. Рассмотрим данные правила:

а) На вход функции подается расписание для оценки.

б) Исходное расписание разбивается, в зависимости от вида ограничения, на отдельные расписания:

- по учебным группам;
- по преподавателям;
- по аудиториям;
- по учебной группе и дню;
- по преподавателю и дню и др.

в) Далее производится проход по каждому отдельном расписанию, полученному на предыдущем шаге, и для каждого расписания вычисляются следующие значения (формула 2.6):

$$(rd, rt) = \left(\sum_{i=1}^{n} t_i * \varphi(i, a) , \sum_{i=1}^{n} t_i \right) \quad (2.6)$$

где, n – число элементов в выделенном расписании,
a – выделенное для проверки расписание,
t_i – весовой коэффициент для i-го элемента расписания,
φ – функция проверки ограничения.

г) Весовой коэффициент t_i для элемента расписания может определяться различными способами в зависимости от вида ограничения:

- количество обучающихся в потоке для элемента расписания;
- кол-во элементов в расписании и др.

д) Функция проверки ограничения φ принимает на вход номер элемента в расписании для проверки и само расписание, и как результат возвращает одно из 2-х возможных значений:

- 1, если функция определила, что ограничение нарушено
- 0, если ограничение не нарушено.

е) На заключительном этапе производится вычисление значения функции оценки выполнения ограничения по следующей формуле (формула 2.7):

$$R = 1 - \frac{\sum_{i=1}^{n} rd_i}{\sum_{i=1}^{n} rt_i} \quad (2.7)$$

Предложенная схема построения функций оценки выполнения ограничений позволит реализовать ряд оптимизаций, которые уменьшат объем вычислений, что в свою очередь, снизит время поиска решения. К таким оптимизациям относятся:

а) Как было указано, в функциях оценки производится разбивка расписания на отдельные расписания (по учебной группе, аудитории, преподавателю и др.). Одна и та же разбивка расписания может понадобиться в разных ограничениях, поэтому можно реализовать построение данных отдельных расписаний при внесении изменений в генах и на вход функций оценки ограничений уже подавать готовый набор расписаний.

б) Поскольку мы проводим оценку набора расписаний в функциях, то если изменений в проверяемом расписании нет, то значения переменных *rd* и

rt для данного расписанию останутся теми же самыми. Исходя из этого, мы можем сохранять эти значения в некотором буфере и использовать их в расчетах, без повторного вычисления, если изменений в расписании не было.

Описанная выше схема для большинства видов ограничения. Однако, есть ряд ограничений, которые могут быть реализованы в форме ограничения на диапазон допустимых значений и исключены из целевой функции. К таким ограничениям относятся:

а) Учет вместимости и приемлемости аудитории для проведения занятия. Для реализации данных ограничений в форме диапазона возможных значений требуется предварительно определить допустимые аудитории для учебного потока из множества доступных аудиторий и при определении значения переменной гена *A* (аудитория) выбирать значения только из этого диапазона.

б) Исключений отдельных дней и/или временных интервалов из расписания для учебной группы или преподавателя. Для реализации данных ограничений в форме диапазона возможных значений так же требуется предварительно определить допустимые варианты комбинаций дня и временного интервала, которые допустимы для каждого учебного потока и при определении значений переменных генов *W* (день) и *T* (временные интервалы) выбирать одну из возможных комбинаций значений для них.

Кроме этого, для проверки допустимости возможных комбинаций значений переменных гена могут применяться дополнительные специальные проверочные функции, которые будут определять допустимость данного варианта комбинаций значений. В случае недопустимости такой комбинации значений переменных исключать их из рассмотрения. Такого рода функции могут применяться для реализации такого ограничения как исключение определенных дней и временных интервалов для аудитории и др.

Рассмотренные нами варианты проверки выполнения ограничений позволят реализовать поиск решения задачи с учетом накладываемых на итоговое расписание ограничений.

2.5 Выводы по второму разделу

Для решения задачи составления расписаний учебных занятий в данном исследовании была предложена схема на основе модификации генетического алгоритма. В рамках данной части исследования мы выполнили следующее:

а) Рассмотрели, какие исходные документы используются при составлении расписаний учебных занятий: рабочие учебные планы, индивидуальные планы обучающихся, индивидуальные планы преподавателей и др.

б) Разработали математическую модель расписания учебных занятий. Данная модель включает в себя:

- Входные данные для задачи: учебные группы и подгруппы, преподаватели, аудитории, дисциплины, виды занятий, дни, временные

интервалы и учебные потоки. Для указанных входных данных мы определили их характеристики.

- Выходные данные для задачи, которые представляют собой список картежей со следующим набором значений: учебный поток, аудитория, день и временные интервалы, т.е. для каждого учебного потока мы должны подобрать допустимые значения для аудитории, дня и временных интервалов.

- Предложенная модель позволяет реализовать составление расписания очных занятий, онлайн занятий и смешанных занятий. Это достигается за счет введения в модель специальных виртуальных аудиторий, в которых должны проводиться онлайн-занятий и вместимость которых определяется возможным количеством онлайн-подключений к ней на основе серверных и коммуникационных возможностей.

- Ограничения, которые накладываются на итоговое решение задачи. Данные ограничения делятся на 2 большие группы: «жесткие» (обязательные) и «мягкие» (рекомендации) ограничения. Так же мы определили список основных ограничений, которые могут накладываться на расписание: отсутствие «накладок» для учебной группы, преподавателя, аудитории, отсутствие «окон» в расписании учебной группы, преподавателя и др.

в) В качестве поиска решения задачи был выбран генетический алгоритм. Для этого алгоритма в данном исследовании были определены следующие параметры:

1) Особь представляет собой одну хромосому.

2) Каждый ген в хромосоме представляет собой элемент расписания и включает следующие компоненты:

- неизменяемый элемент гена: учебный поток,

- переменные гена: аудитория, день и временные интервалы,

- вычисляемый элемент гена: дефекты гена, т.е. нарушения ограничений, которые накладываются на расписание занятий.

3) В качестве функции приспособляемости была предложена функция оценки ограничений «жестких» и «мягких» ограничений на основе весовых коэффициентов для каждого ограничения. Для данной функции мы определили следующие параметры:

- оценка выполнения ограничения задается в диапазоне от 0 (ограничение полностью не выполнено) до 1 (ограничение полностью выполнено),

- функция вычисляет 2 отдельные оценки: для «жестких» и «мягких» ограничений,

- при поиске решения мы должны максимизировать значение целевой функции в следующем порядке: сначала по оценке «жестких» ограничений, а затем по оценке «мягких» ограничений.

4) Генерация особей начальной популяции строится на последовательном включении новых генов в хромосому на основе

оптимальности значений определенной нами функции приспособляемости. Порядок включения генов определяется на основе их семантической схожести, которая в данном исследовании определяется с использованием алгоритма кластеризации FOREL.

5) Для формирования новых особей в генетическом алгоритме используются следующие генетические операторы:

- Оператор мутации. Данный оператор выбирает случайным образом особи из популяции и производит модификацию генов в ней на основе функции приспособляемости. В данном исследовании предложены 4 варианта работы данного оператора.

- Оператор скрещивания. Данный оператор выбирает случайным образом 2 особи из популяции, создает копию 1-й особи и производит замену значений в её генах на значения из генов 2-й особи на основе значений функции приспособляемости.

6) Селекция особей для нового поколения определяется по 3-м направлениям, которые идут последовательно:

- выбор особи с наилучшим значением целевой функции.
- выбор особей с наилучшими значениями для отдельных ограничений
- случайный выбор особей.

7) В качестве условия завершения работы поиска решения в алгоритме используются условия достижения максимально возможного значения целевой функции или прекращения роста значений целевой функции в течение нескольких поколений.

8) Выбор «наилучшей» особи, как решения задачи, строится на выборе из последнего поколения особи с максимальным значением функции приспособляемости.

г) Для реализации проверки ограничений в функции приспособляемости предложена схема построения функций проверки, которая предполагает разбивку проверяемого расписания на отдельные расписания и вычисления выполнения ограничения на них. Предложенная схема построения функций проверки позволяет снизить объем вычислений, т.к. позволяет производить пересчет только изменений в расписании, а не рассматривать все расписание каждый раз заново.

д) Для реализации проверки ряда ограничений было предложено использовать другие виды реализации:

- реализация на основе диапазона допустимых значений,
- реализация на основе предварительной проверки на допустимость значений.

По результатам рассмотренных в данном разделе тем были опубликованы тезисы в сборниках 2-х конференций [63, 65], 2 статьи в научных журналах РК [62, 64], а также раздел в монографии [95].

3. Архитектура и программная реализация генетического алгоритма для составления расписания учебных занятий

На рынке программных продуктов существует ряд программных продуктов, которые предоставляют различные возможности по автоматизации составления расписаний занятий: 1С, БИТ.МУЗ, AVTOR и др. Данные программные продукты имеют реализацию на разных программных платформах с возможностью интеграции с другими информационными система [77, 78, 98-100, 108].

Исходя из выше сказанного, при программной реализации генетического алгоритма, предложенного в нашем исследовании, мы должны предусмотреть следующие элементы:

- получение данных с других систем.
- настройка алгоритма под потребности учебного заведения.
- передача готовых данных в другие системы.

Описанные выше требования к программной реализации определили следующие её компоненты:

а) база данных для хранения исходных данных для составления расписания,

б) клиентское приложение, которое будет использоваться для работы с исходными данными,

в) динамическая библиотека, которая будет реализовывать генерацию расписания на основе предложенного во второй главе генетический алгоритм,

г) приложение для генерации расписания, которое будет получать данные из разработанной БД и, используя динамическую библиотеку, производить составление расписание по заданным параметрам.

Рассмотрим реализацию данных компонентов программной реализации.

3.1 Проектирование базы данных для программы составления расписания занятий

Проектирование любой БД производится поэтапно и включает такие этапы как:

- построение инфологической модели БД;
- построение логической модели БД;
- нормализация отношений в БД;
- реализация предложенной логической модели БД в виде физической БД на основе определенной СУБД.

Проведем построение БД для нашего исследования по перечисленным выше этапам проектирования.

3.1.1 Построение инфологической модели БД. На данном этапе построения БД производится определение тех информационных объектов, которые будут включены в рассматриваемую БД. Кроме того, для каждого

определяемого информационного объекта определяется его атрибутивный состав.

В рамках данного исследования нами были выявлены необходимые информационные объекты и их атрибуты, которые приведены в таблице 3.1.

Таблица 3.1 – Перечень информационных объектов и атрибутов

Объект	**Атрибут**
Менеджер расписания	Идентификатор менеджера расписания
	Название менеджера расписания
Дисциплина	Идентификатор дисциплины расписания
	Идентификатор менеджера расписания
	Идентификатор дисциплины
	Название дисциплины
Преподаватель	Идентификатор преподавателя расписания
	Идентификатор менеджера расписания
	Идентификатор преподавателя
	ФИО преподавателя
Атрибут преподавателя	Идентификатор атрибута преподавателя
	Идентификатор преподавателя расписания
	Название атрибута
	Значение атрибута
Время занятия	Идентификатор времени занятия расписания
	Идентификатор менеджера расписания
	Идентификатор времени занятия
	Время начала занятия
	Время завершения занятия
	Номер занятия
День	Идентификатор дня занятия
	Идентификатор менеджера расписания
	Название
	Битовая маска занятости
Учебная группа	Идентификатор группы расписания
	Идентификатор менеджера расписания
	Идентификатор группы
	Шифр группы
Атрибуты учебной группы	Идентификатор атрибута учебной группы
	Идентификатор группы расписания
	Название атрибута
	Значение атрибута
Вид учебной аудитории	Идентификатор типа учебной аудитории
	Название типа учебной аудитории
Аудитория	Идентификатор аудитории для расписания
	Идентификатор менеджера расписания

Объект	**Атрибут**
	Идентификатор аудитории
	Количество мест в аудитории
	Идентификатор типа учебной аудитории
	Название аудитории
Атрибуты аудитории	Идентификатор атрибута аудитории для расписания
	Идентификатор аудитории для расписания
	Название атрибута
	Значение атрибута
Вид занятия	Идентификатор вида занятия
	Название вида занятия
Аудитории для дисциплины	Идентификатор аудитории для дисциплины расписания
	Идентификатор вида занятия
	Идентификатор дисциплины
	Идентификатор аудитории
	Идентификатор преподавателя
	Идентификатор вида учебной аудитории
	Приоритет выбора аудитории
Учебный поток	Идентификатор учебного потока для расписания
	Идентификатор менеджера расписания
	Число учебных занятий для потока
	Количество повторений для потока
	Идентификатор вида занятия
Атрибуты учебного потока	Идентификатор атрибута учебного потока
	Идентификатор учебного потока для расписания
	Название атрибута
	Значение атрибута
Учебные группы потока	Идентификатор группы учебного потока
	Идентификатор учебного потока для расписания
	Идентификатор группы
	Количество обучающихся
	Идентификатор дисциплины
	Номер учебного периода
	Номер подгруппы
	Маска включения обучающихся группы в поток
Преподаватели потока	Идентификатор преподавателя потока

Объект	Атрибут
	Идентификатор учебного потока для расписания
	Идентификатор преподавателя
	Номер подгруппы

На заключительном шаге построения инфологической модели БД строится специальная диаграмма – ER-диаграмма. Данная диаграмма показывает взаимосвязи между определенными в модели информационными объектами. На данной диаграмме представлены следующие виды элементов:

- информационные объекты – изображаются в виде прямоугольников с названием соответствующих информационных объектов.

- действия связи между информационными объектами – изображаются в виде ромбов с название соответствующих действий.

- связи между объектами и действиями – изображаются в виде соединительных линий между объектами и действиями.

Для нашей БД ER-диаграмма будет иметь следующий вид (рисунок 3.1):

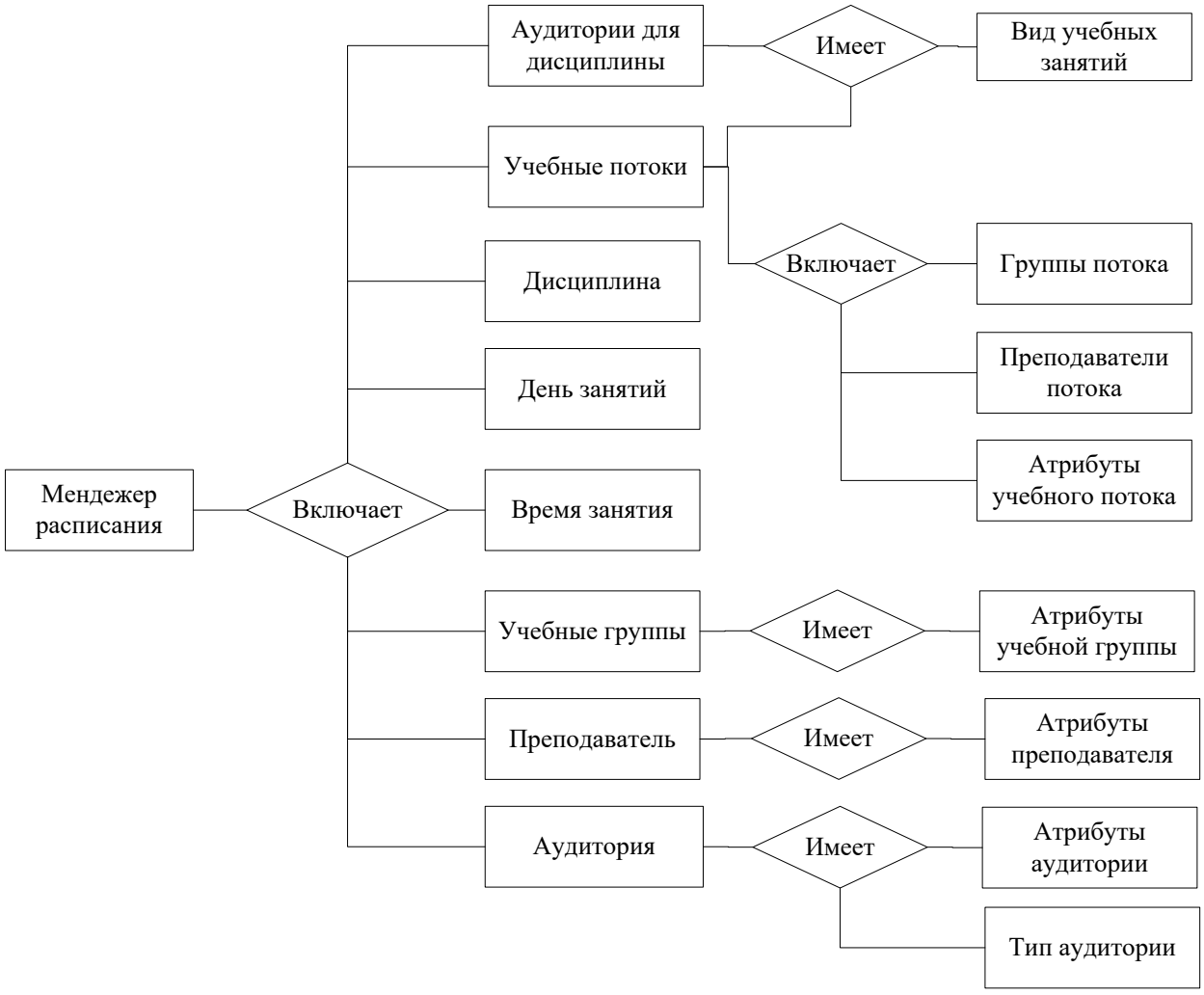


Рисунок 3.1 – ER-диаграмма БД

Построенная нами инфологическая модель БД будет являться основой для следующего этапа проектирования – построение логической модели БД.

3.1.2 Построение логической модели БД. На данном этапе проектирования на основе построенной инфологической модели производится построение логической модели, в рамках которой производятся следующие действия:

- определяются необходимые отношения и их атрибуты;
- для каждого отношения определяются первичный ключ;
- для связи между отношениями определяются внешние ключи;
- проводится нормализация отношений до 3-й нормальной формы;
- производится построение логической схемы данных.

Исходя из вышеизложенного, в рамках данного исследования для построения логической модели нами были выполнены все изложенные действия. Результат данного этапа проектирования приведен в таблице 3.2, в которой представлены следующие элементы:

- наименование отношения;
- для каждого отношения представлен перечень его атрибутов и типы данных этих атрибутов;
- для каждого отношения определен первичный ключ (ПК)
- если атрибут отношения связан с другими отношениями, то данный атрибут помечен как внешний ключ (ВК) с указанием отношения, на которое он ссылается.

Таблица 3.2 – Логическая модель данных

Отношения	**Атрибут**	**Тип**	**Ключ**	**Ссылка на родительское отношение**
Manager	ManagerID	Целое	ПК	
	Name	Строка		
Discipline	TimetableDisciplineID	Целое	ПК	
	ManagerID	Целое	ВК	Manager
	DisciplineID	Целое		
	Name	Строка		
Teachers	TimetableTeacherID	Целое	ПК	
	ManagerID	Целое	ВК	Manager
	TeacherID	Целое		
	Name	Строка		
TeacherAttributes	TeacherAttributeID	Целое	ПК	
	TimetableTeacherID	Целое	ВК	Teachers
	Name	Строка		
	Value	Строка		
LectureTime	TimetableLectureTimeID	Целое	ПК	
	ManagerID	Целое	ВК	Manager

Отношения	Атрибут	Тип	Ключ	Ссылка на родительское отношение
	LectureTimeID	Целое		
	StartTime	Время		
	FinishTime	Время		
	NumberLecture	Целое		
Days	TimetableDayID	Целое	ПК	
	ManagerID	Целое	ВК	Manager
	Name	Строка		
	Mask	Строка		
Groups	TimetableGroupID	Целое	ПК	
	ManagerID	Целое	ВК	Manager
	GroupID	Целое		
	Code	Строка		
GroupAttributes	GroupAttributeID	Целое	ПК	
	TimetableGroupID	Целое	ВК	Groups
	Name	Строка		
	Value	Строка		
RoomTypes	RoomTypeID	Целое	ПК	
	Name	Строка		
Rooms	TimetableRoomID	Целое	ПК	
	ManagerID	Целое	ВК	Manager
	RoomID	Целое		
	Places	Целое		
	RoomTypeID	Целое	ВК	RoomTypes
	Name	Строка		
RoomAttributes	RoomAttributeID	Целое	ПК	
	TimetableRoomID	Целое	ВК	Rooms
	Name	Строка		
	Value	Строка		
TypeLecture	TypeLecture	Целое	ПК	
	Name	Строка		
DisciplineRooms	DisciplineRoomID	Целое	ПК	
	TypeLecture	Целое	ВК	TypeLecture
	ManagerID	Целое	ВК	Manager
	DisciplineID	Целое		
	RoomID	Целое		
	TeacherID	Целое		
	RoomTypeID	Целое	ВК	RoomTypes
	PrioritySelect	Целое		
Flows	FlowID	Целое	ПК	
	ManagerID	Целое	ВК	Manager

Отношения	Атрибут	Тип	Ключ	Ссылка на родительское отношение
	HoursCount	Целое		
	CountElements	Целое		
	TypeLecture	Целое	ВК	TypeLecture
FlowAttributes	FlowAttributeID	Целое	ПК	
	FlowID	Целое	ВК	Flows
	Name	Строка		
	Value	Строка		
FlowGroups	FlowGroupID	Целое	ПК	
	FlowID	Целое	ВК	Flows
	GroupID	Целое		
	CountStudent	Целое		
	DisciplineID	Целое		
	TermNumber	Целое		
	SubGroup	Целое		
	SubGroupMask	Строка		
FlowTeachers	FlowTeacherID	Целое	ПК	
	FlowID	Целое	ВК	Flows
	TeacherID	Целое		
	SubGroup	Целое		

В заключение проектирования логической модели БД для лучшего представления связей между отношениями строится логическая схема БД, которая для нашего исследования приведена на рисунке 3.2.

3.1.3 Реализация физической модели БД. После построения логической модели БД производится её реализация на физическую модель в определенной системе управления базами данных. Для данного исследования в качестве СУБД была выбрана СУБД Microsoft SQL Server 2017.

Перенос логической модели на физическую модель производится по следующей схеме:

- каждое отношение логической модели отображается в физическую таблицу с заданным набором полей, которые соответствуют атрибутам отношения;
- для каждой таблицы при её создании указывается первичный ключ для полей, которые соответствуют атрибутам первичного ключа отношения в логической модели БД;
- для каждого внешнего ключа в логической модели в физической БД создается соответствующий внешний ключ в дочерней таблице, который ссылается на поле в родительской таблице.

В современных СУБД для создания перечисленных выше объектов физической БД используется язык Data Definition Language (DDL), который является подмножеством скриптового языка SQL для определения метаданных БД. В приложении А приведен скрипт на языке SQL для создания физической БД для нашего исследования.

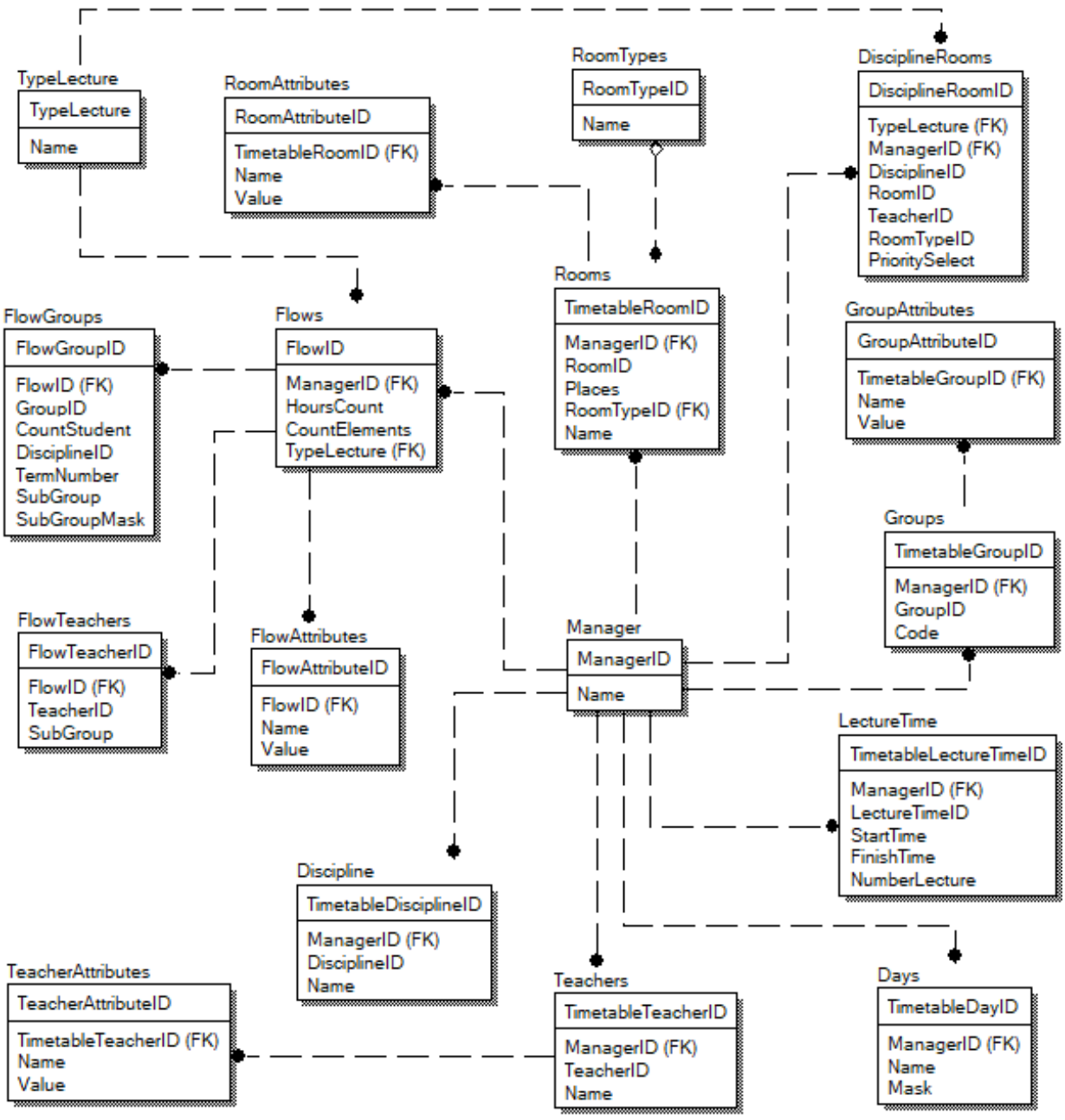


Рисунок 3.2 – Логическая схема БД

На этом этап проектирования БД для нашего исследования завершен, и мы можем перейти к следующему этапу проектирования – создание клиентского приложения для работы с созданной БД.

3.2 Программная реализация клиентского приложения для работы с БД для составления расписания занятий

Для работы с данным в БД, модель которой представлена в разделе 3.1, было разработано клиентское приложение в среде Embracodero Delphi 2009. Данное приложение представляет собой автономное windows-приложение. Главное окно данного приложения представлено на рисунке 3.3.

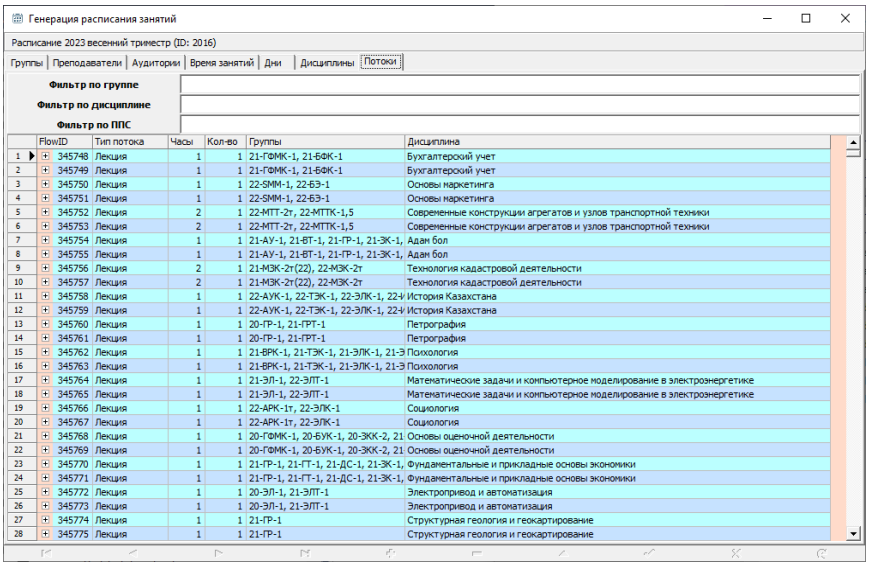


Рисунок 3.3 – Главное окно приложения

Для создания и/или выбора расписания для редактирования в верхней части окно имеется кнопку «Выбор расписания…» нажав которую откроется диалоговое окно «Менеджера расписаний», в котором имеется возможность редактирования и/или выбора расписания (рисунок 3.4).

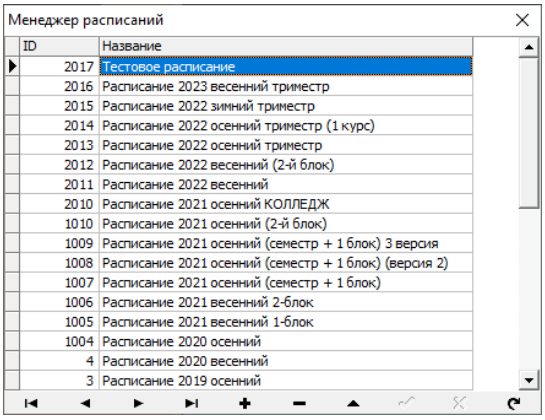


Рисунок 3.4 – Окно «Менеджер расписаний»

Для выбора расписания для редактирования необходимо в окне «Менеджера расписания» выбрать соответствующую строку и закрыть

диалоговое окно. После этого в главном окне приложения на соответствующих вкладках появятся данные по выбранному расписанию:

А) На вкладке «Группы» представлен список групп, для которых требуется составить расписание. Для каждой группы нужно указать следующий набор данных (рисунок 3.5):

- Идентификатор группы.
- Шифр группы.
- Дополнительные атрибуты группы, если они имеются. Каждый атрибут задается парой значений: имя атрибута и его значение.

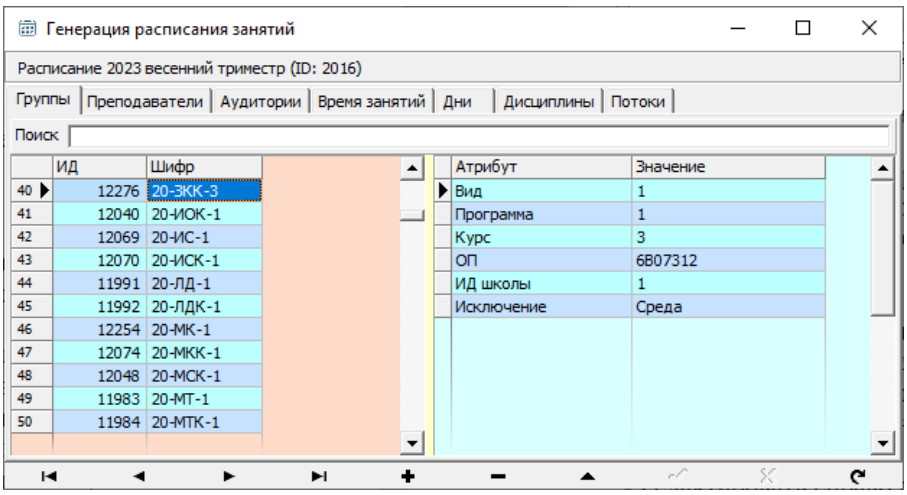


Рисунок 3.5 – Вкладка «Группы»

Б) На вкладке «Преподаватели» представлен список преподавателей, которые будут вести занятия. Для каждого преподавателя нужно указать следующий набор данных (рисунок 3.6):

- Идентификатор преподавателя.
- ФИО преподавателя.
- Дополнительные атрибуты преподавателя, если они имеются. Каждый атрибут задается парой значений: имя атрибута и его значение.

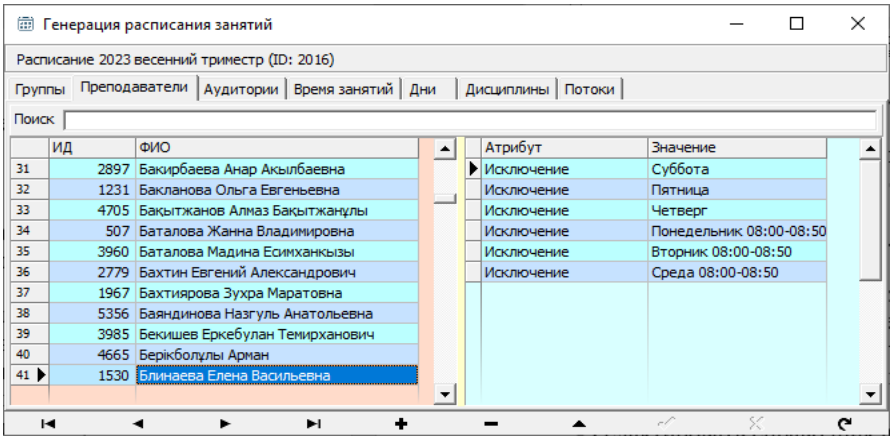


Рисунок 3.6 – Вкладка «Преподаватели»

В) На вкладке «Аудитории» представлен список аудиторий, которые будут использоваться для составления расписания занятий. Для каждой аудитории нужно указать следующий набор данных (рисунок 3.7):

- Идентификатор аудитории.
- Номер аудитории.
- Количество посадочных мест в аудитории.
- Тип аудитории.
- Дополнительные атрибуты аудитории, если они имеются. Каждый атрибут задается парой значений: имя атрибута и его значение.

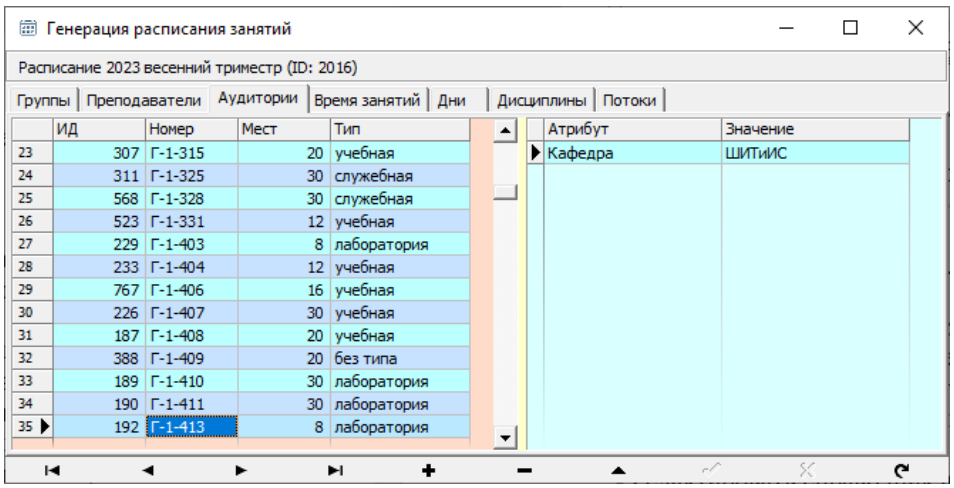


Рисунок 3.7 – Вкладка «Аудитории»

Г) На вкладке «Время занятий» представлен список временных интервалов для проведения занятий по расписанию. Для каждого временного интервала нужно указать следующий набор данных (рисунок 3.8):

- Идентификатор временного интервала.
- Порядковый номер временного интервала в сетки расписания.
- Время начала занятия.
- Время завершения занятия.

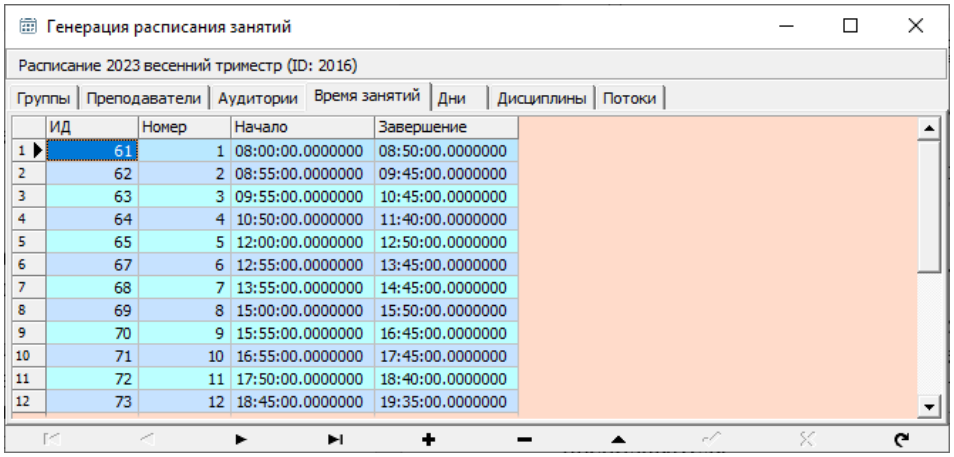
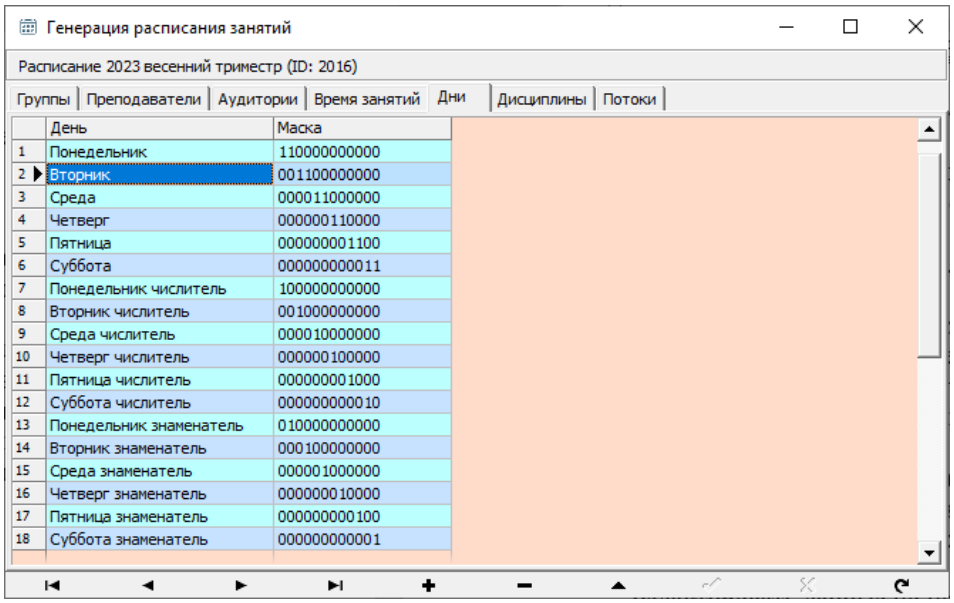


Рисунок 3.8 – Вкладка «Время занятий»

Д) На вкладке «Дни» представлен перечень дней для проведения занятий для расписания. Для каждого дня необходимо задать следующий набор значений (рисунок 3.9):

- Наименование дня – день недели, календарный день и др.
- Маска дня. Маска представляет собой строку, которая представляет маску занятости данного дня в сетке расписания. Задается маска в виде упорядоченной последовательности цифр: 1 (занято) и 0 (не занято).



	День	Маска
1	Понедельник	110000000000
2	Вторник	001100000000
3	Среда	000011000000
4	Четверг	000000110000
5	Пятница	000000001100
6	Суббота	000000000011
7	Понедельник числитель	100000000000
8	Вторник числитель	001000000000
9	Среда числитель	000010000000
10	Четверг числитель	000000100000
11	Пятница числитель	000000001000
12	Суббота числитель	000000000010
13	Понедельник знаменатель	010000000000
14	Вторник знаменатель	000100000000
15	Среда знаменатель	000001000000
16	Четверг знаменатель	000000010000
17	Пятница знаменатель	000000000100
18	Суббота знаменатель	000000000001

Рисунок 3.9 – Вкладка «Дни»

Е) На вкладке «Дисциплины» представлен перечень дисциплин, по которым должны проводиться занятия по расписанию. Для каждой дисциплины необходимо задать следующий набор данных и значений (рисунок 3.10):

- Идентификатор дисциплины.
- Наименование дисциплины.
- Набор аудиторий для дисциплины. Аудитории для проведения занятий по дисциплине задаются в виде следующего набора данных:

1) Тип занятия по дисциплине – лекция, практика, лабораторная и др.

2) Вид аудитории – в данном поле указывается вид аудитории, в которой возможно проведение выбранного типа занятий. Данное поле заполняется, только если для проведения выбранного типа занятия можно использовать данный тип аудитории, иначе поле оставляется пустым.

3) Аудитория – в данном поле указывается конкретная аудитория, в которой возможно проведения выбранного типа занятий. Данное поле заполняется, если для проведения выбранного типа занятия можно использовать конкретную аудитории, иначе поле оставляется пустым.

4) Преподаватель – в данном поле указывается ФИО преподавателя, для которого возможно использование данной аудитории или типа аудитории для

проведения выбранного типа занятия. Если аудитория или тип аудитории предназначен для любых преподавателей, то данное поле оставляется пустым.

5) Приоритет – определяет приоритет выбора аудитории из списка доступных аудиторий при составлении расписания.

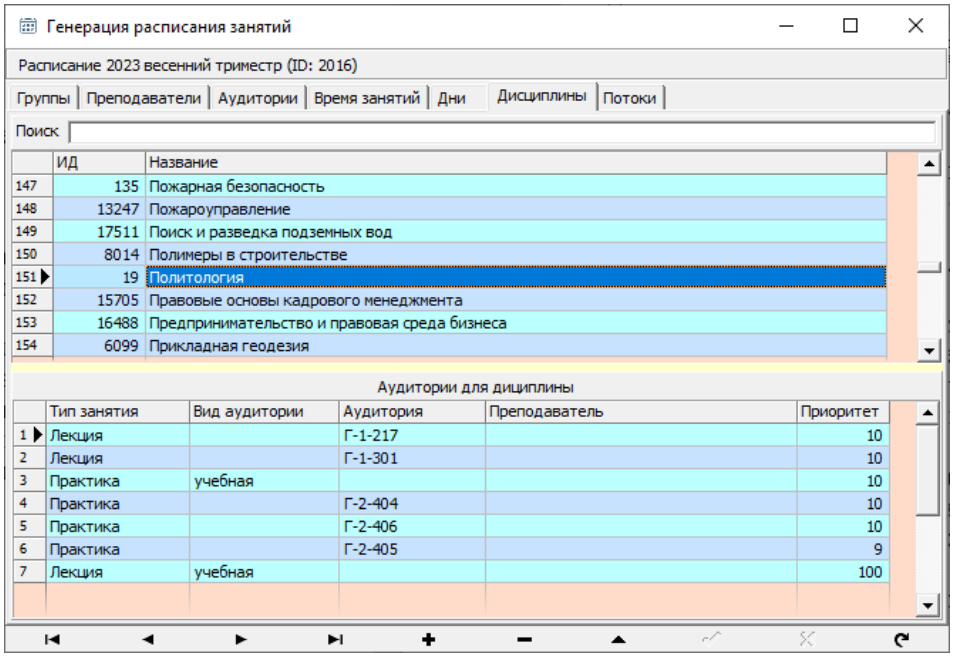


Рисунок 3.10 – Вкладка «Дисциплины»

Ж) На вкладке «Потоки» представлены перечень академических потоков, которые должны быть включены в расписание занятий. Для каждого академического потока необходимо задать следующий набор данных и значений (рисунок 3.11):

- Идентификатор потока
- Тип потока – лекция, практика, лабораторная или др.
- Количество часов занятий для потока.
- Количество повторений данного потока в расписании.
- Дополнительные атрибуты академического потока, если они имеются. Каждый атрибут задается парой значений: имя атрибута и его значение.
- Набор учебных групп академического потока. Для каждой группы в академическом потоке указывается следующий набор значений:

1) Шифр учебной группы.

2) Название дисциплины, по которой должно проводиться занятие.

3) Количество обучающихся данной учебной группы, которые должны изучать данную дисциплину в академическом потоке.

4) Номер семестра по учебному плану.

5) Номер подгруппы.

6) Маска подгруппы. Маска подгруппы представляет собой маску занятости обучающихся группы в данном учебном потоке. Задается маска в

виде упорядоченной последовательности цифр: 1 (обучающийся занимается в данной подгруппе) и 0 (обучающийся не занимается в данной подгруппе).

- Набор преподавателей, которые ведут занятия для данного академического потока. Для каждого преподавателя в академическом потоке указывается следующий набор значений:

1) ФИО преподавателя.

2) Номер подгруппы.

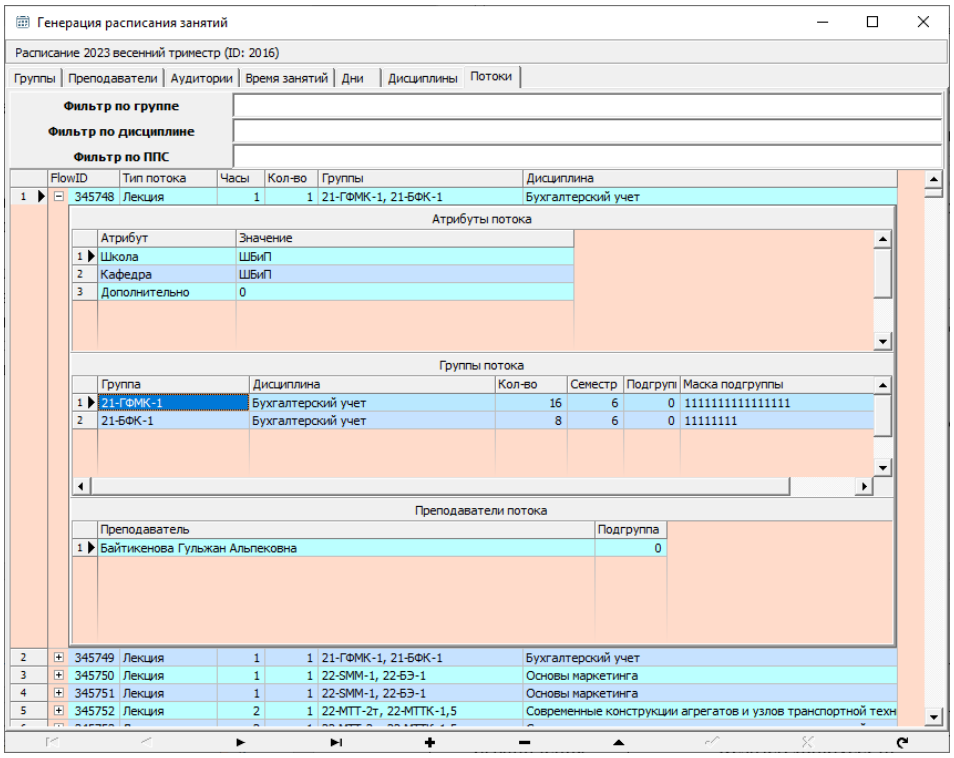


Рисунок 3.11 – Вкладке «Потоки»

Для удобства работы на вкладках «Группы», «Преподаватели», «Дисциплины» и «Потоки» имеются поля ввода, которые позволяют проводить фильтрацию таблиц на указанных вкладках.

С помощью описанного выше приложения имеется возможность создавать и редактировать данные в БД, которые в дальнейшем будут использоваться для составления расписания учебных занятий.

Исходный код описанного выше клиентского приложения для работы с БД представлен в приложении Б.

3.3 Программная реализация динамической библиотеки для составления расписания занятий на основе генетического алгоритма

Следующим этапом данного исследования является программная реализация предложенного в главе 2 модификации генетического алгоритма для составления расписания занятий. Реализацию указанной динамической библиотеки мы будем производить на базе программного каркаса Microsoft .Net Framework 4.7, а в качестве языка программированию будем

использовать язык программирования C#. Для проведения реализации библиотеки будет использоваться среда программирования Microsoft Visual Studio 2017.

Создание приложений в программном каркасе .Net Framework предполагает проектирование и реализацию различных классов, структур, интерфейсов и других элементов, которые объединяются в пространства имен, которые имеют иерархическую структуру. Для нашей библиотеки в качестве корневого пространства имен выбрали имя *ektu.Timetable*.

Для удобства реализации и использования данной динамической библиотеки мы раздели реализуемые в ней компоненты на несколько частей:

а) Справочники – в данной части находятся компоненты, которые представляют собой справочные элементы, требуемые для составления расписаний занятий.

б) Академические потоки – в данной части расположены компоненты, которые содержат сведения об академических потоках, которые должны быть включены в расписание занятий.

в) Элементы расписания – в данную часть включены классы представляющий отдельные элементы расписания и вариант некоторого расписания, классы реализующие генерацию начального расписания, классы реализующие предложенные в второй главе варианты работы генетических операторов скрещивания и мутации, а также вспомогательные классы разбивки расписания на отдельные расписания, которые будут использоваться при проверке ограничений как было предложено в 4-м разделе 2-й главы.

г) Ограничения – в данной части содержаться классы, которые реализуют проверку ограничений, которые накладываются на расписание.

д) Генератор расписания – в данную часть включен класс, который реализует работу предложенного генетического алгоритма для составления расписания занятий.

Рассмотрим более подробно реализацию каждой части описанной выше динамической библиотеки.

3.3.1 Справочники. Данная часть динамической библиотеки содержит компоненты, которые представляют справочные данные для составления расписания и размещена в пространстве имен *ektu.Timetable.Dict*.

В состав данной части библиотеки входят:

а) Абстрактный класс Item. Данный класс содержит свойства для хранения, чтения и записи данных по объекту, которые используются в классах потомках. Данные в объекте класса хранятся в виде пар значений «имя свойства» - «значение свойства». Такая модель хранения данных позволяет хранить различные атрибуты объектов разных типов.

б) Класс Room. Данный класс является потоком класса Item и представляет учебную аудиторию в расписании и имеет такие специфические свойства для аудитории такие как:

- RoomID – идентификатор аудитории;

- Places – количество мест в аудитории.

в) Класс Teacher. Данный класс является потомком класса Item и представляет преподавателя в расписании. Данный класс имеет одно специфическое свойство TeacherID – идентификатор преподавателя.

г) Класс Discipline. Данный класс является потомком класса Item и представляет дисциплину в расписании. Данный класс имеет такие специфические свойства как:

- DisciplineID – идентификатор дисциплины
- Rooms – список аудиторий по видам занятий для данной дисциплины.

д) Класс Group. Данный класс является потомком класса Item и представляет академическую группу в расписании. Данный класс имеет одно специфическое свойство GroupID – идентификатор группы.

е) Класс LectureTime. Данный класс является потомком класса Item и представляет временной интервал в расписании. Данный класс имеется следующие специфические свойства:

- LectureTimeID – идентификатор временного интервала,
- StartTime – время начала занятия,
- FinishTime – время завершения занятия,
- NumberLecture – порядковый номер временного интервала в сетке расписания,
- Next – следующий по порядку временный интервал временного интервала,
- Cross – список временных интервалов, которые пересекаются с текущим временным интервалом,

Схема описанных выше классов приведено на рисунке 3.12.

ж) Абстрактный класс DayItem. Данный класс представляет собой день в расписании и реализует интерфейс IComparable для сравнения объектов данного класса и их упорядочивания. Данный класс имеет следующие специфические свойства:

- ID – идентификатор дня,
- Cross – список дней, которые пересекаются с текущим днем.

з) Класс DateDayItem. Данный класс является потомком класса DayItem и представляет календарный день в расписании. Данный класс имеет следующие специфические свойства:

- поле date – календарный день,
- поле id – идентификатор календарного дня,
- конструктор DateDayItem – данный конструктор принимает на вход дату и на её основе создает новый объект класса,
- статистический метод DateTimeID – данный метод принимает на вход дату и на её основе создает уникальный целочисленный идентификатор для данной календарной даты,
- переопределенный метод ToString – данный метод создает текстовое представление для данного объекта класса DateDayItem.

- свойство ID – переопределяет свойство базового класса и возвращает уникальный идентификатор дня, полученный через метод DateTimeID.

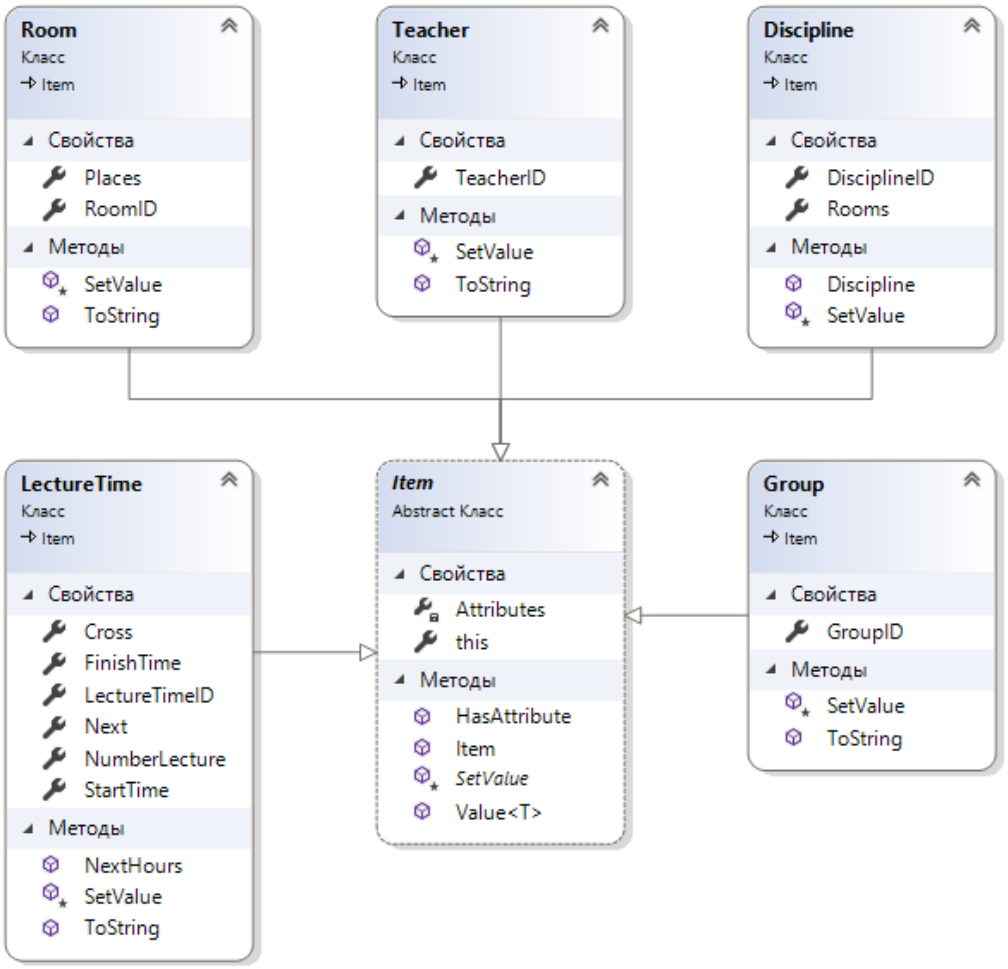


Рисунок 3.12 – Классы справочники

и) Класс WeekDayItem. Данный класс является потомком класса DayItem и представляет собой день недели в расписании. Данный класс имеет следующие специфические свойства:

- статические свойства, которые хранят объекты данного класса, представляющие дни недели: Moday (понедельник), Tuesday (вторник), Wednesday (среда), Thursday (четверг), Friday (пятница), Saturday (суббота) и Sunday (воскресение).

- свойство WeekDayID – идентификатор для дня недели

- свойство ID – переопределяет свойство базового класса и возвращает уникальный идентификатор WeekDayID дня недели.

- переопределенный метод ToString – данный метод создает текстовое представление для данного объекта класса WeekDayItem.

к) Класс WeekDay2Item. Данный класс является потомком класса DayItem и представляет собой день недели в расписании по схеме

«числитель/знаменатель», т.е. занятия проводимы каждую неделю или раз в 2 недели. Данный класс содержит следующие специфические свойства:

- перечисление WeekType, которое представляет тип недели для проведения занятий: Both (каждая неделя), Week1 (числительная неделя) и Week2 (знаменательная неделя).
- свойство DayID – идентификатор дня недели.
- свойство WeekNumber – данное свойство имеет тип WeekType и представляет тип недели,
- свойство ID – переопределяет свойство базового класса и возвращает уникальный идентификатор дня недели.
- переопределенный метод ToString – данный метод создает текстовое представление для данного объекта класса WeekDay2Item.

Схема описанных выше классов, представляющих дни в расписании, приведено на рисунке 3.13.

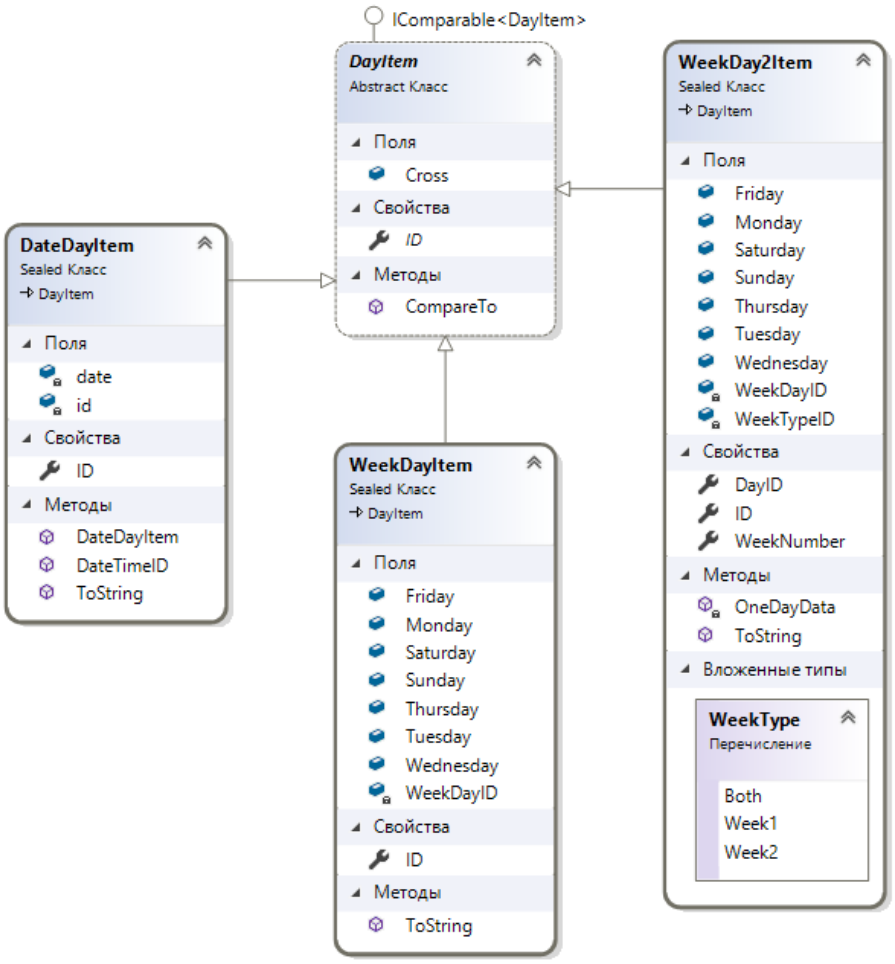


Рисунок 3.13 – Классы дней для расписания

3.3.2 Академические потоки. В данной части динамической библиотеки содержатся компоненты для описания академических потоков, из которых формируется расписание занятий. Компоненты данной части библиотеки входят в пространство имен ektu.Timetable.Flow.

В состав компонентов данной части библиотеки входят:

а) Абстрактный класс TypeFlowBase – представляет абстрактный класс для различных видов занятий в расписании. Данный класс имеет следующие специфические свойства:

- ID – идентификатор вида занятия,
- Name – название вида занятия.

б) Класс TypeFlowLecture – данный класс является потомком класса TypeFlowBase и представляет класс для лекционного занятия в расписании. В данном классе имеется закрытый конструктор и специальное статическое свойство Item, которое хранит объект данного класса. Данная схема позволяет реализовать шаблон проектирования Singleton (шаблон наличия единственный объект класса).

в) Класс TypeFlowPractice – данный класс является потомком класса TypeFlowBase и представляет класс для практического занятия в расписании. Данный класс построен по той же схеме, что и класс TypeFlowLecture.

г) Класс TypeFlowLaboratory – данный класса является потомком класса TypeFlowBase и представляет класс для лабораторного занятия в расписании. Данный класс построен по той же схеме, что и класс TypeFlowLecture.

д) Класс RoomSelectInfo – данный класс представляет описание для выбора аудитории для дисциплины и используется в классе представляющим дисциплину (ektu.Timetable.Dict.Discipline) и имеет следующие специфические свойства:

- Room – объект, представляющий аудиторию,
- Priority – приоритет выбор аудитории для занятия.

е) Класс SelectDataWeekTime – данный класс представляет описания для выбора дня и временного интервала для занятия и имеет следующие специфические свойства:

- Day – объект, представляющий день занятия,
- Time – объект, представляющий временной интервал для занятия,
- Priority – приоритет выбора дня и временного интервала для занятия.

ж) Класс FlowItem – данный класс представляет данные по академической группе в потоке и имеет следующие специфические свойства:

- CountStudent – количество обучающихся группы включенных в академический поток,
- Discipline – объект дисциплины, по которой у академической группы проводятся занятие в потоке,
- Group – объект, представляющий академическую группу,
- SubGroup – номер подгруппы,
- SubGroupMask – битовая маска, которая показывает занятость обучающихся группы в академическом потоке.

з) Класс FlowData – данный класс представляет объект академического потока в расписании и имеет следующие специфические свойства:

- CountHours – число временных интервалов для академического потока,
- CountStudent – число обучающихся в академическом потоке,

- DayTime – список объектов класса SelectDataWeekTime для выбора дня и временного интервала для составления расписания,

- FirstDiscipline – объект класса Discipline, который описывает дисциплину для академического потока,

- FlowId – идентификатор академического потока,

- Items – список объектов данные по академическим группам, включенным в данный академический поток,

- Linked – список объектов связанных академических потоков, т.е. потоков параметры включения которых в расписание завит от параметров включения данного потока (например, если занятие по связанным потокам должны следовать один за другим или если занятия должны проводится в разные дни и т.п.),

- Par – список объектов академических потоков, занятия по которым должны проводиться параллельно, т.е. в один и тот же день и один и тот же временной интервал (например, занятие по подгруппам и т.п.),

- Rooms – список объектов класса RoomSelectInfo для выбора аудитории для проведения занятия для данного академического потока,

- Teachers – список объектов преподавателей, которые должны вести занятия в данном академическом потоке,

- TypeFlow – объект вида занятия для данного академического потока.

и) Класс ForelClassificator – данный класс реализует модель кластеризации академических потоков на основе их сематической близости с использованием алгоритма FOREL. Данный класс имеет следующие специфические компоненты:

- Конструктор ForeClassificator. Данный конструктор принимает на вход следующий набор параметров:

1) список академических потоков

2) весовой коэффициент для меры связи академических групп между двумя академическими потоками (SI_G, таблица 2.1),

3) весовой коэффициент для меры связи преподавателей между двумя академическими потоками (SI_T, таблица 2.1),

4) весовой коэффициент для меры связи аудиторий между двумя академическими потоками (SI_A, таблица 2.1).

На основе значений данных параметров конструктора формируется матрица смежности для заданных академических потоков для проведения кластеризации.

- закрытый метод FindCenter – данный метод реализует поиск центра для кластера по алгоритму, предложенному в разделе 2.3.3.1.

- закрытый метод FindCluster – данный метод реализует поиск кластера по предложенному в разделе 2.3.3.1 алгоритму кластеризации.

- открытый метод GetClusters – данный метод реализует поиск всех кластеров по предложенному в разделе 2.3.3.1 алгоритму кластеризации на основе заданного радиуса поиска, который передается в метод в качестве его параметра.

Схема компонентов данной части представлена на рисунке 3.14.

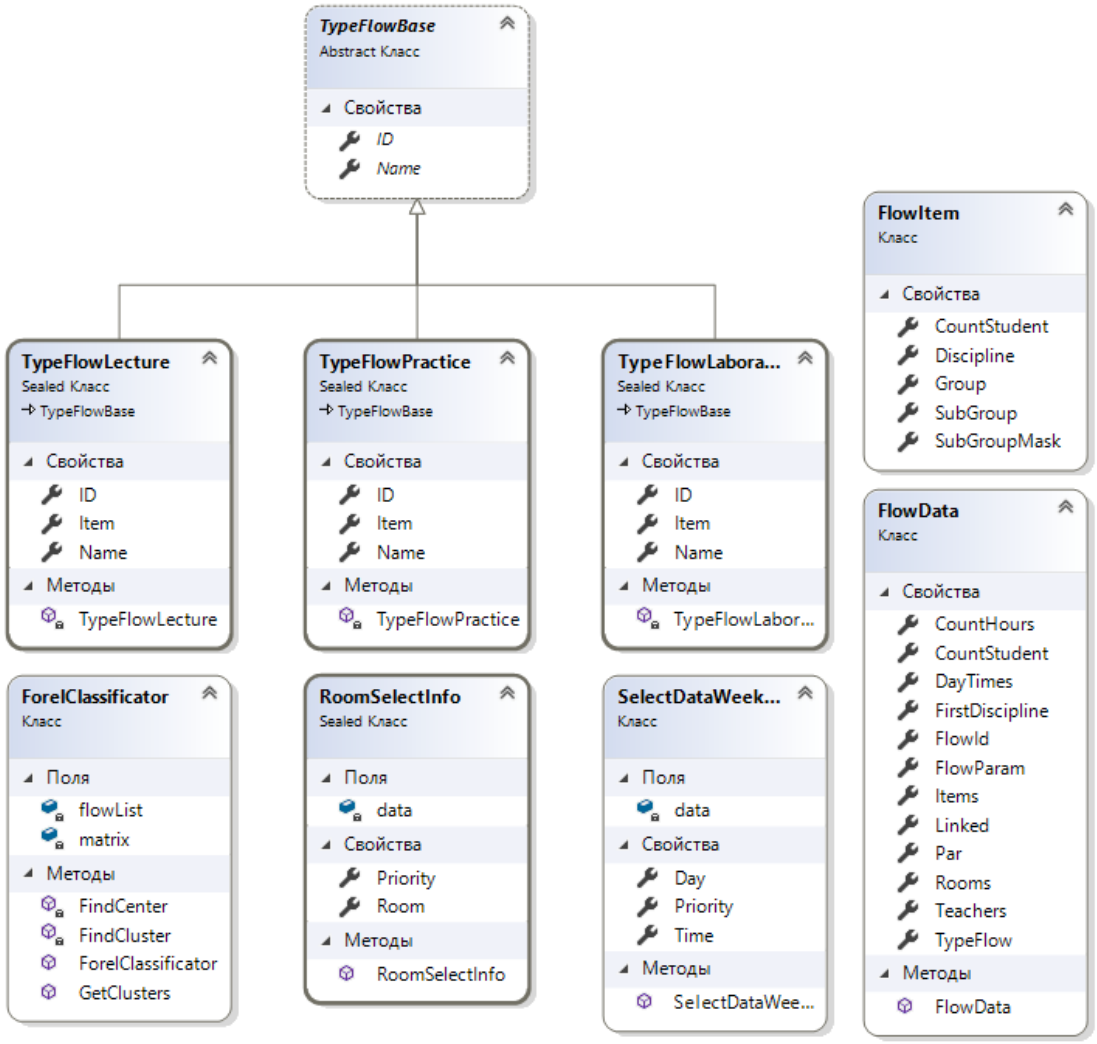


Рисунок 3.14 – Схема компонентов академических потоков

3.3.3 Элементы расписания. В данной части динамической библиотеки содержатся компоненты, представляющие расписание занятий и вспомогательные компоненты для составления расписания. Компоненты данной части располагаются в пространстве имен ektu.Timetable.Timetable.

Компоненты в данной части библиотеки условно можно разделить на следующие группы:

- Компоненты расписания – данные компоненты используются для хранения некоторого варианта расписания и его компонентов,
- Компоненты формирования расписания – данные компоненты реализуют механизмы генетического алгоритма, предложенного в 2-м разделе данного исследования по генерации начального расписания, применения генетических операторов (мутации и скрещивания) для изменения расписания, а также компоненты для разбивки расписания на

блоки, которые используются в системе оценки выполнения ограничений как было предложено в разделе 2.4 данного исследования.

Рассмотрим данные компоненты более подробно.

3.3.3.1 Компоненты расписания. К компонентам расписания в нашей динамической библиотеки относятся следующие её элементы:

а) Перечисление TypeGenTimetalbe – данное перечисление задает для расписания, в результате какой операции оно было получено: начальное расписание (Initial), расписание, полученное в результате применения оператора мутации (Mutating) или расписание, полученное в результате применения оператора скрещивания (Crossing).

б) Класс TimetableItem – данный класс представляет занятие в расписании и имеет следующие специфические компоненты:

- свойство Day – день занятия,
- свойство Room – аудитория для занятия,
- свойство Time – временной интервал для занятия,
- свойство Flow – объект академического потока, по которому проводится занятие
- свойство Owner – объект расписания, в которое входит данное занятие,
- свойство NotModified – логическое значение, которое определяет, может ли изменяться данный элемент в операции мутации или скрещивания,
- свойство IsDefected – логическое значение, которое показывает, есть ли нарушения ограничений в данном элементе расписания,
- свойство DefectedCount – показывает количество нарушений ограничений для данного элемента расписания,
- свойство HardDefectEval – показывает сумму коэффициентов по нарушениям «жестких» ограничений для элемента расписания,
- свойство SortDefectEval – показывает сумму коэффициентов по нарушениям «мягких» ограничений для элемента расписания,
- свойство GetLinked – список связанных элементов расписания с данным элементом расписания,
- свойство GetPar – список параллельных элементов расписания с данным элементом расписания,
- метод Assing – данный метод используется для одновременного задания значений для свойств Day, Time и Room,
- метод SetDefected – данный метод используется для отметки или снятия метки нарушения ограничения в данном элементе расписания,
- событие ClearItem – данное событие возникает при изменении переменных элемента расписания (Day, Time, Room) и используется для уведомления объекта расписания для очистки кэша отдельных расписаний (по группе, по преподавателю, по аудитории),
- событие ChangeItem – данное событие при изменении переменных элемента расписания (Day, Time, Room) и используется для уведомления объекта расписания для включения элемента в кэш отельных расписаний.

в) Класс TimetableData – данный класс представляет некоторое расписание занятий и включает следующие специфические компоненты:

- индексатор класса – данный компонент позволяет получить элемент расписания по его номеру или по академическому потоку,
- свойство Count – количество элементов в расписании,
- свойство Generation – количество поколений, в которых существует расписание,
- свойство Type – тип получения расписания: начальное, мутация или скрещивание,
- свойство HardRestrictValue – значение целевой функции для «жестких» ограничений,
- свойство SortRestrictValue – значение целевой функции для «мягких» ограничений,
- свойство RestrictValue – сумма значений целевой функции для «жестких» и «мягких» ограничений,
- свойство ScheduleByGroups – список расписаний по отдельным академическим группам построенное из элементов текущего расписания,
- свойство ScheduleByTeacher – список расписаний по отдельным преподавателям построенное из элементов текущего расписания,
- свойство ScheduleByRoom – список расписаний по отдельным аудиториям построенное из элементов текущего расписания,
- метод AddItem – данный метод добавляет элемент в текущее расписание,
- методы CheckRestrinct и UpdateRestrictValues – данные методы проводят оценку выполнения ограничений, накладываемых на расписание,
- метод Clone – данный метод создает полную копию текущего расписания,
- метод IsEqual – данный метод сравнивает текущее расписание с другим расписанием на идентичность,
- методы Defected и NoDefected – данные методы возвращают элементы расписания, которые имеют или не имеют нарушения ограничений,
- метод GetRestrictItemValue – данный метод возвращает значение выполнения указанного ограничения,
- метод GetXml – данный метод возвращает текущее расписание в формате XML,
- метод IsBusyRoom – данный метод проверяет, занята ли переданная аудитория в заданное время и день,
- метод IsBusyTeacher – данный метод проверяет, занят ли преподаватель в заданное время и день.

д) Класс TimetableItemProxy – вспомогательный класс, используемый для хранения отдельных расписаний (по группе, по преподавателю или по аудитории) на основе основного расписания. Данный класс введен для хранения данных при наличии пересечения временных интервалов в расписании. Указанный класс имеет следующие специфические свойства:

- data – объект элемента расписания,

- IsBlank – логическое свойство, которое задает, что элемент расписания имеет отношение к другому временному интервалу, который пересекается с другими временным интервалом.

е) Класс TimetableSubject – данный класс представляет отдельное расписание (по группе, по преподавателю или по аудитории) построенное из общего расписания. Указанный класс имеет следующие специфические компоненты:

- свойство AllData – данное свойство возвращает список элементов отдельного расписания в разрезе дне и временных интервалов,

- свойство AsList – данное свойство возвращает список элементов отдельного расписания,

- метод AddItem – данный метод добавляет элемент расписания в список элементов отдельного расписания,

- метод ClearItem – данный метод удаляет элемент расписания из списка элементов отдельного расписания,

- метод ByDayTime – данный метод возвращает список элементов отдельного расписания по заданному дню и временному интервалу,

- метод IsDayTime – данный метод возвращает логическое значение о наличии занятия в указанный день и временной интервал в расписании,

- методы GetCacheRestrict и SetCacheRestrict – данные методы получают и задают вычисленное значения для кэша вычисления выполнения ограничения (формула 2.6) относящееся ко всему расписанию,

- методы GetCacheWeekDay и SetCacheWeekDay – данные методы получают и задают вычисленное значения для кэша вычисления выполнения ограничения (формула 2.6) относящееся к определенному дню в расписании.

Схема описанных компонентов представлена на рисунке 3.15.

3.3.3.2 Компоненты формирования расписания. Компоненты данной части используется для реализации составления расписания на разных этапах работы предложенного генетического алгоритма. В состав данной части входят следующие компоненты:

а) Интерфейс ITimetableDataInitial – данный интерфейс предоставляет метод Generate для создания начального расписания,

б) Интерфейс ITimetableDataCrosser – данный интерфейс предоставляет метод Execute для реализации оператора скрещивания,

в) Интерфейс ITimetableDataMutator – данный интерфейс предоставляет метод Execute для реализации оператор мутации,

г) Класс TimetableDataInitial – данный класс реализует интерфейс ITimetableDataInitial генерации начального расписания на основе алгоритма, предложенного во 2-й главе,

д) Класс TimetableDataCrosser – данный класс реализует интерфейс ITimetableDataCrosser оператора скрещивания на основе схемы алгоритма, предложенного во 2-й главы для оператора скрещивания,

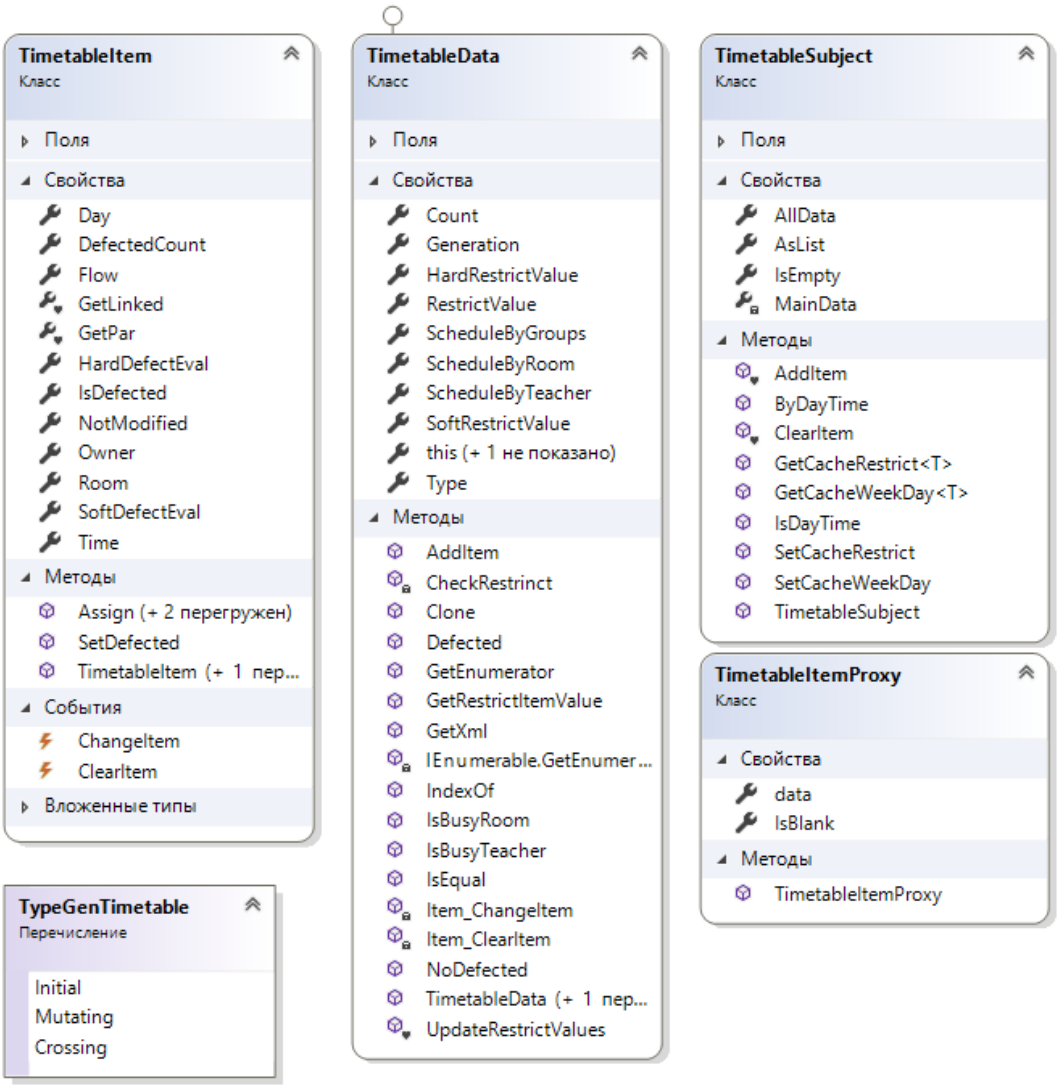


Рисунок 3.15 – Компоненты расписания

е) Класс TimetableDataMutator – данный класс реализует интерфейс ITimetableDataMutator оператора мутации на основе схем работы данного генетического оператора предложенного в 2-м разделе данного исследования. Указанный класс реализует 4 метода, которые реализуют предложенные схемы работы оператора мутации:

- RandomMutation – модификация некоторого числа генов, которые нарушают некоторое выбранное ограничение, накладываемое на расписание,
- RandomGroupMutation – попарная модификация некоторого числа генов,
- DefectMutation – модификация набора связанных генов по учебной группе, аудитории и преподавателю,
- MutationSelect – модификация дефективного гена и связанных с ним генов по учебной группе, аудитории и преподавателю.

Вызов одно из перечисленных методов осуществляется в реализации метода Execute случайным образом.

ж) Класс Excluder – данный класс используется для исключения комбинаций значений переменных в расписании по заданным условиями. В данном классе реализован единственный виртуальный метод Exclude возвращающий логическое значение об исключении заданных значений переменных расписания из рассмотрения. Реализация данного метода в классе всегда возвращает логическое значение «ложь», т.е. не исключать никакие возможные значения переменных из рассмотрения. Для реализации исключения значений переменных пользователи библиотеки должны переопределить указанный метод.

з) Статический класс TimetableOptimalFinder – данный класс реализует методы, используемые для формирования начальной популяции и оператора мутации:

- FindNonBusyRoom – данный метод возвращает свободную аудиторию для элемента расписания, если она имеется, иначе возвращается пустое значение,
- FindAnyRoom – данный метод возвращает любую подходящую аудиторию для элемента расписания, если она имеется, иначе возвращается пустое значение,
- ClearData – данный метод устанавливает пустые значения переменным для элемента расписания,
- SaveItemValues – данный метод сохраняет значения для элемента расписания, которые можно использовать для восстановления исходных значений элемента расписания,
- FindOptimal – данный метод производит поиск значений для переменных элемента расписания на основе оптимальности значения целевой функции (формула 2.3),
- FindOptimal2 – данный метод производит поиск значений для переменных 2-х элементов расписания на основе оптимальности значения целевой функции (формула 2.3).

Схема описанных компонентов представлена на рисунке 3.16.

3.3.4 Ограничения. В данной части динамической библиотеки представлены компоненты, которые реализуют проверку ограничений, накладываемых на расписание занятий.

В состав данной части входят следующие элементы:

а) Интерфейс IRestrict – данный интерфейс объявляет свойства и методы, которые должны быть реализованы в классах, которые будут реализовывать проверку ограничений, накладываемых на расписание. В данном интерфейсе объявлены следующие элементы:

- Свойство Name – имя ограничения,
- Свойство Factor – весовой коэффициент для ограничения,
- Метод Check – данный метод при реализации данного интерфейса в классе должен реализовать проверку расписания на выполнение ограничения и на выходе должен вернуть число с плавающей запятой. Возвращаемое

значение должно быть в диапазоне от 0 до 1 согласно требованиям к функции приспособляемости для нашего генетического алгоритма (раздел 2.3.1).

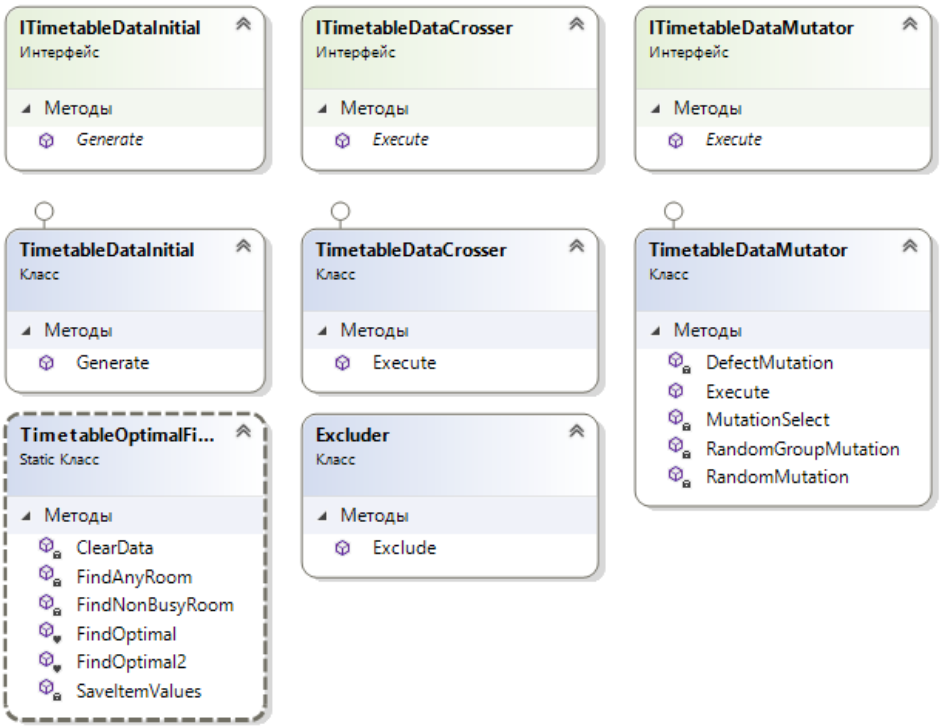


Рисунок 3.16 – Компоненты формирования расписания.

б) Класс RestrictRoom – данный класс реализует проверку ограничения на отсутствие накладок для аудитории. Реализация данной проверки осуществляет по схеме построения функции проверки (раздел 2.4):

- из исходного расписания берется расписание аудиторий,
- производится проход по всем расписаниям аудиторий и для каждого расписания аудитории проверяется, было ли изменение в данном расписании,
- если есть изменения в расписании аудитории или нет результатов предыдущей проверки расписания, то производится проверка накладок для аудитории, иначе используется результаты предыдущего вычисления,
- проверка накладок аудитории проводится по 2 разным видам в зависимости от вида аудитории:

1) метод calcOneLesson – данный метод осуществляет проверку того, что в аудитории может проводиться только одно занятие,

2) метод calPlaceLesson – данный метод осуществляет проверку того, что количество обучающихся в аудитории в данный временной интервал не превышает вместимость аудитории,

- на заключительном этапе производится вычисление значения выполнения ограничения.

в) Класс RestrictGroup – данный класс реализует проверку ограничения на отсутствие накладок для академической группы. Реализация данной

проверки осуществляется по схеме построения функции проверки (раздел 2.4):

- из исходного расписания берется расписание групп,
- проводится проход по всем расписаниям групп и для каждого расписания группы проверяется, было ли изменение в данном расписании,
- если есть изменения в расписании группы или нет результатов предыдущей проверки расписания, то производится проверка накладок для группы, иначе используются результаты предыдущего вычисления,
- проверка накладок по группе производится с использованием метода calc путем проверки количества занятий группы по дням и временным интервалам и если количество занятий более 1, то имеет место нарушение ограничения,
- на заключительном этапе производится вычисление значения выполнения ограничения.

г) Класс RestrictEmployee – данный класс реализует проверку ограничения на отсутствие накладок для преподавателя. Реализация данной проверки осуществляется по схеме построения функции проверки (раздел 2.4):

- из исходного расписания берется расписание преподавателей,
- производится проход по всем расписаниям преподавателей и для каждого расписания преподавателя проверяется, было ли изменение в данном расписании,
- если есть изменения в расписании преподавателя или нет результатов предыдущей проверки расписания, то производится проверка накладок для преподавателя, иначе используются результаты предыдущего вычисления,
- проверка накладок по преподавателю производится с использованием метода calc путем проверки количества занятий преподавателя по дням и временным интервалам и если количество занятий более 1, то имеет место нарушение ограничения,
- на заключительном этапе производится вычисление значения выполнения ограничения.

д) Класс RestrictGroupFill – данный класс реализует проверку ограничения на отсутствие разрывов в расписании группы в течение дня. Реализация данной проверки осуществляется по схеме построения функции проверки (раздел 2.4):

- из исходного расписания берется расписание учебных групп,
- производится проход по всем расписаниям групп и для каждого расписания группы проверяется, было ли изменение в данной расписании,
- если есть изменения в расписании группы или нет результатов предыдущей проверки расписания, то производится проверка наличия разрывов в течение дня у группы, иначе используется результаты предыдущего вычисления,
- проверка разрывов в расписании группы производится по следующей схеме:

1) расписание группы разбивается на дни

2) если есть изменения в расписании дня группы или нет результатов предыдущей проверки дня, то производится проверка наличия разрывов в расписании, иначе используется результаты предыдущего вычисления,

3) на заключительном этапе производится подсчет значений по всем дням расписания группы,

- на заключительном этапе производится вычисление значения выполнения ограничения.

е) Класс RestrictEmployeeFill – данный класс реализует проверку ограничения на отсутствие разрывов в расписании преподавателя в течение дня. Реализация данной проверки осуществляется по схеме построения функции проверки (раздел 2.4):

- из исходного расписания берется расписание преподавателей,

- производится проход по всем расписаниям преподавателей и для каждого расписания преподавателя проверяется было ли изменение в данном расписании,

- если есть изменения в расписании преподавателя или нет результатов предыдущей проверки расписания, то производится проверка наличия разрывов в течение дня у преподавателя, иначе используется результаты предыдущего вычисления,

- проверка разрывов в расписании преподавателя производится по следующей схеме:

1) расписание преподавателя разбивается на дни

2) если есть изменения в расписании дня преподавателя или нет результатов предыдущей проверки дня, то производится проверка наличия разрывов в расписании, иначе используется результаты предыдущего вычисления,

3) на заключительном этапе производится подсчет значений по всем дням расписания преподавателя,

- на заключительном этапе производится вычисление значения выполнения ограничения.

ж) Класс RestrictGroupHours – данный класс реализует проверку ограничения на количество занятий в течение дня у группы. Разрешенное количество занятий для группы задается в конструкторе класса, в который должны быть переданы минимальное и максимальное количество занятий у группы в течение дня. Реализация данной проверки осуществляется по схеме построения функции проверки (раздел 2.4):

- из исходного расписания берется расписание учебных групп,

- производится проход по всем расписаниям групп и для каждого расписания группы проверяется, было ли изменение в данной расписании,

- если есть изменения в расписании группы или нет результатов предыдущей проверки расписания, то производится проверка на разрешенное количество занятий в течение дня у группы, иначе используется результаты предыдущего вычисления,

- проверка разрешенного количества занятий в течение дня в расписании группы производится по следующей схеме:

1) расписание группы разбивается на дни

2) если есть изменения в расписании дня группы или нет результатов предыдущей проверки дня, то производится подсчет количества занятий в течение дня, и проверка не нарушает полученное количество установленного числа занятий, иначе используется результаты предыдущего вычисления,

3) на заключительном этапе производится подсчет значений по всем дням расписания группы,

- на заключительном этапе производится вычисление значения выполнения ограничения.

з) Класс RestrictEmployeeHours – данный класс реализует проверку ограничения на количество занятий в течение дня у преподавателя. Разрешенное количество занятий для группы задается в конструкторе класса, в который должны быть переданы минимальное и максимальное количество занятий у преподавателя в течение дня. Реализация данной проверки осуществляется по схеме построения функции проверки (раздел 2.4):

- из исходного расписания берется расписание преподавателей,

- производится проход по всем расписаниям преподавателей и для каждого расписания преподавателя проверяется было ли изменение в данном расписании,

- если есть изменения в расписании преподавателя или нет результатов предыдущей проверки расписания, то производится проверка на разрешенное количество занятий в течение дня у преподавателя, иначе используется результаты предыдущего вычисления,

- проверка разрешенного количества занятий в течение дня в расписании преподавателя производится по следующей схеме:

1) расписание преподавателя разбивается на дни

2) если есть изменения в расписании дня преподавателя или нет результатов предыдущей проверки дня, то производится подсчет количества занятий в течение дня, и проверка не нарушает полученное количество установленного числа занятий, иначе используется результаты предыдущего вычисления,

3) на заключительном этапе производится подсчет значений по всем дням расписания преподавателя,

- на заключительном этапе производится вычисление значения выполнения ограничения.

и) Класс RestrictOneDayDisciplines – данный класс реализует проверку ограничения на повторение занятие по определенной дисциплине и определенному виду занятий в течение дня у группы, т.е. если занятия по определенной дисциплине и определенному виду занятий должны проводиться в разные дни. Реализация данной проверки осуществляется по схеме построения функции проверки (раздел 2.4):

- из исходного расписания берется расписание учебных групп,

- производится проход по всем расписаниям групп и для каждого расписания группы проверяется, было ли изменение в данной расписании,

- если есть изменения в расписании группы или нет результатов предыдущей проверки расписания, то производится проверка на повторение занятия по определенным дисциплинам и определенным видам занятий в течение дня у группы, иначе используется результаты предыдущего вычисления,

- проверка разрешенного количества занятий в течение дня в расписании группы производится по следующей схеме:

1) расписание группы разбивается на дни

2) если есть изменения в расписании дня группы или нет результатов предыдущей проверки дня, то производится проверка, не повторяются ли занятия по определенной дисциплине и определенному виду занятий в течение дня у группы, иначе используется результаты предыдущего вычисления,

3) на заключительном этапе производится подсчет значений по всем дням расписания группы,

- на заключительном этапе производится вычисление значения выполнения ограничения.

к) Класс RestrictDayDiscipline – данный класс реализует проверку ограничения на количество дисциплин, по которым могут проводиться занятия в течение дня у группы:

- лекционные занятия – не более 3-х дисциплин в течении,

- практические и лабораторные занятия – не более 3-х дисциплин в течение дня.

Реализация данной проверки осуществляется по схеме построения функции проверки (раздел 2.4):

- из исходного расписания берется расписание учебных групп,

- производится проход по всем расписаниям групп и для каждого расписания группы проверяется, было ли изменение в данной расписании,

- если есть изменения в расписании группы или нет результатов предыдущей проверки расписания, то производится проверка разрешенное количество дисциплин в течение дня у группы, иначе используется результаты предыдущего вычисления,

- проверка разрешенного количества дисциплин в течение дня в расписании группы производится по следующей схеме:

1) расписание группы разбивается на дни

2) если есть изменения в расписании дня группы или нет результатов предыдущей проверки дня, то производится проверка не превышает ли количество дисциплин разрешенное число в течение дня у группы, иначе используется результаты предыдущего вычисления,

3) на заключительном этапе производится подсчет значений по всем дням расписания группы,

- на заключительном этапе производится вычисление значения выполнения ограничения.

Кроме реализации проверки ограничений описанные выше классы имеются специальные свойства (Filter, FilterItem, FilterFlow) для исключения отдельных элементов расписания из проверки по тем или иным причинам.

Схема описанных компонентов представлена на рисунке 3.17.

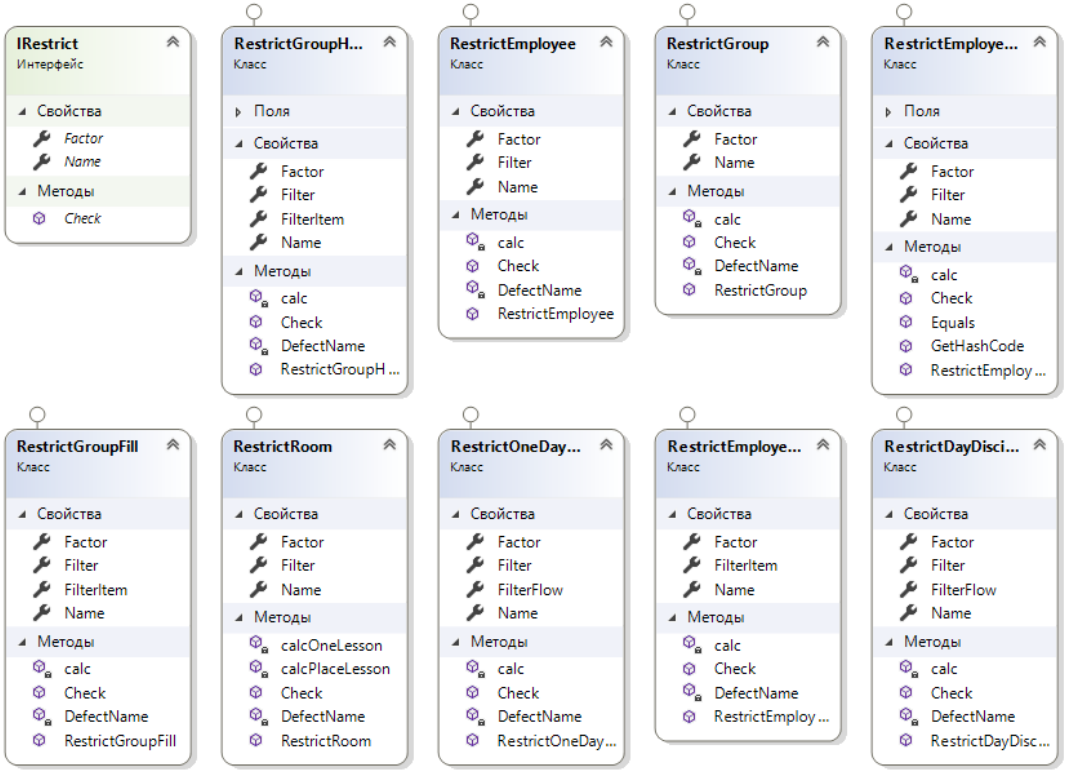


Рисунок 3.17 – Компоненты проверки ограничений для расписания

3.3.5 Генератор расписания. Последним элементом нашей динамической библиотеки является класс генератора расписания Generator, который реализует работу предложенного в данном исследовании генетического алгоритма составления расписания занятий.

Данный класс включает в себя следующие компоненты:

а) Свойства объекта класса:

- CountTimetable – размер начальной популяции, т.е. данное свойство задает сколько расписаний будет сгенерировано на этапе генерации начальной популяции.

- CountTimetableCalc – размер популяции для отбора на этапе формирования новой популяции.

- Crosses – указатель на класс, реализующий оператор скрещивания. По умолчанию данное свойство хранит объект класса TimetableDataCrosser (раздел 3.3.3). Пользователи библиотеки могут предоставить собственную версию класса, который будет проводить операцию скрещивания.

- CrossingSelectPercent – задает вероятность выбора особей для применения оператора скрещивания.

- Initiator – указатель на класс, реализующий операцию генерации начального расписания. По умолчанию данное свойство хранит объект класса TimetableDataInitial (раздел 3.3.3). Пользователи библиотеки могут предоставить собственную версию класса, который будет проводить операцию генерации начального расписания.

- Mutator - указатель на класс, реализующий оператор мутации. По умолчанию данное свойство хранит объект класса TimetableDataMutator (раздел 3.3.3). Пользователи библиотеки могут предоставить собственную версию класса, который будет проводить операцию мутации.

- MutationSelectPercent – задает вероятность выбора особей для применения оператора мутации.

- FitnessValue – значение целевой функции, при котором нужно прекратить поиск решения.

- Excluder – класс для исключения комбинаций значений переменных в расписании по заданным условиями. По умолчанию данное свойство хранит объект класса Excluder (раздел 3.3.3). Пользователи библиотеки должны предоставить собственную реализацию класса Excluder, если они хотят реализовать собственную схему исключения комбинаций значений переменных для расписания.

б) Абстрактные свойства объекта класса – данные свойства необходимо реализовать пользователям класса для предоставления сведений необходимых для составления расписания:

- Disciplines – список дисциплин для расписания,
- Flows – список академических потоков для расписания,
- Groups – список академических групп для расписания,
- Rooms – список аудитория для расписания,
- Teachers – список преподавателей для расписания,
- Times – список временных интервалов для расписания,
- WeekDays – список дней для расписания,
- Hard – список «жестких» ограничений, накладываемых на расписание,
- Soft – список «мягких» ограничений, накладываемых на расписание.

в) Методы объект класса:

- MutateHardExec – данный метод выполняет выборку и проведение мутации расписаний,

- CrossingExec – данный метод выполняет выборку и проведения скрещивания расписаний,

- SortTimetables – данные метод производит сортировку популяции на значения целевой функции расписаний,

- GenerateTimetable – данный метод реализует работу предложенного в данном исследования генетического алгоритма.

г) События объект класса

- InitialInfo – данное событие возникает при завершении генерации начальной популяции,

- StartStep – данное событие возникает при начале проведения операции мутации и скрещивания

- FinishExecute – данное событие возникает при завершении операций мутации и скрещивания,

- FinishAlive – данное событие возникает при завершении работы генетического алгоритма,

- InfoAction – данное событие уведомляет о возникновении какого-либо события при работе генетического алгоритма.

Схема класса генератора расписания приведено на рисунке 3.18.

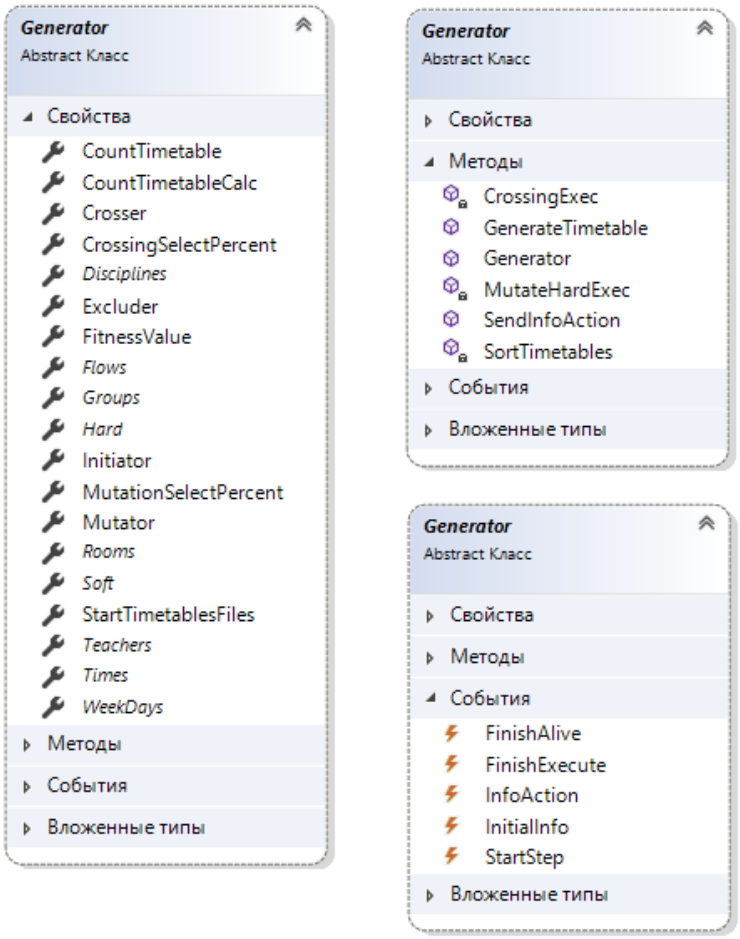


Рисунок 3.18 – Класс генератора расписания

Исходный код рассмотренной нами динамической библиотеки приведен в приложении В.

3.4 Применение разработанных компонентов для генерации расписания

Разработанные в рамках данного исследования база данных, приложение для работы с базой данных и динамическая библиотека, реализующая работу генетического алгоритма, могут быть использованы для реализации приложения для генерации расписания в рамках конкретного высшего учебного заведения.

Реализация генерации при использовании перечисленных выше компонентов может быть выполнена по следующий схеме:

а) Необходимо загрузить данные из информационной системы учебного заведения в разработанную в пункте 3.1 базу данных,

б) С использованием разработанного в разделе 3.2 приложения работы с базой данной внести дополнительные или скорректировать данные необходимые для составления расписания,

в) Реализовать абстрактный класс Generator из раздела 3.3.3, реализовав как минимум следующие элементы:

- абстрактные свойства, задающие справочные данные для составления расписания: Disciplines, Flows, Groups, Rooms, Teachers, Times и WeekDays,
- абстрактные свойства содержание перечень «жестких» и «мягких» ограничений для расписания: Hard и Soft
- задать параметры работы алгоритма: размер начальной популяции, размер отбора особей и желаемое значение целевой функции,
- задать обработчики отслеживания событий, возникающих при работе генетического оператора.

г) Класс расписания в динамической библиотеке имеет возможность сохранения расписания в XML-формате. Используя данный формат представления расписания, имеется возможность импортировать его информационную систему учебного заведения.

Данная схема является общей и может быть изменена и уточнена под нужды конкретного учебного заведения. В приложении Г приведен шаблон консольного приложения на языке программирования C#, который реализует предложенную выше схему генерации расписания на основе разработанной в данном разделе динамической библиотеки.

Поскольку данное исследование проводится на базе ВКТУ им.Д.Серикбаева, то мы будем использовать предложенную схему для реализации генерации расписания на базе образовательного портала данного учебного заведения. Данную реализацию мы рассмотрим в следующей главе.

3.5 Заключение по третьему разделу

Для реализации предложенного в рамках данного исследования генетического алгоритма составления расписания занятий нами были определены и реализованы следующие компоненты:

а) Спроектирована и реализована база данных для хранения исходных данных, требуемых для составления расписания:

- построена инфологическая модель БД: определены объектный и атрибутивный состав БД, построена ER-диаграмма базы данных показывающая связи между объектами БД,

- на основе инфологической модели построена логическая модель БД: определены отношения и их атрибуты, определены первичные ключи отношений и внешние ключи, произведена нормализация отношений в БД,

- на основе логической модели БД она была реализована в физическую БД на базе СУБД Microsoft SQL Server 2017.

б) Для работы с разработанной БД было спроектировано и разработано автономное клиентское приложение, которое позволяет производить редактирование данных в БД.

в) Разработана динамическая библиотека на базе программного фреймворка Microsoft .Net 4.7, в котором реализована работа предложенного в данном исследовании генетического алгоритма составления расписания занятий. В указанной выше динамической библиотеке были реализованы следующие компоненты:

- Справочные компоненты – компоненты, представляющие справочные сведения, необходимые для составления расписания занятий,

- Компоненты академических потоков – компоненты, представляющие сведения об академических потоках, которые должны быть включены в расписание занятий,

- Компоненты расписания – компоненты, представляющие отдельные элементы расписания и расписание в общем, а также компоненты, реализующие операции генерации начальной популяции, операции мутации и скрещивания предложенного нами генетического алгоритма.

- Компоненты ограничений – компоненты, работающие по схеме, предложенной в разделе 2.4, и реализующие проверку выполнения ограничений, накладываемых на расписание.

- Генератор расписания – данный компонент библиотеки является центральным звеном, который реализует схему работы предложенного в данном исследовании генетического алгоритма по формированию расписания занятий.

Разработанные в данной части исследования компоненты являются расширяемыми, что позволяют произвести генерацию расписания, учитывающую особенности конкретного учебного заведения. Для настройки работы разработанных компонентов в данной части исследования была предложена схема настройки работы компонентов под конкретное учебное заведение.

По результатам рассмотренных в данном разделе тем были опубликованы тезисы в сборниках 2-х конференций [63, 65], 3 статьи в научных журналах РК [62, 64, 108], а также получено свидетельство на разработанный программный продукт [67] (Приложение Д).

4. Апробация разработанного генетического алгоритма

Предложенный в данном исследовании генетический алгоритм для составления расписания и его программная реализация могут быть использованы совместно с информационными системами учебных заведений.

Апробация программной реализации алгоритма составления расписания данного исследования была осуществлена на образовательном портале ВКТУ им. Д.Серикбаева (www.do.ektu.kz). Указанный образовательный портал объединяет в своем составе различные элементы, связанные с поддержкой учебного процесса, в том числе и расписание занятий.

4.1 Архитектура образовательного портала ВКТУ им.Д.Серикбаева

Архитектура образовательного портала ВКТУ им.Д.Серикбаева построена с использованием технологии клиент-сервер и представлена на рисунке 4.1. Для хранения данных используется система управления базами данных Microsoft SQL Server 2017. Доступ к информации осуществляется с помощью Web-сайта (http://www.do.ektu.kz) под управлением Web-сервера Internet Information Server 10.0 на базе операционной системы Windows 2019 Server [101, 102].

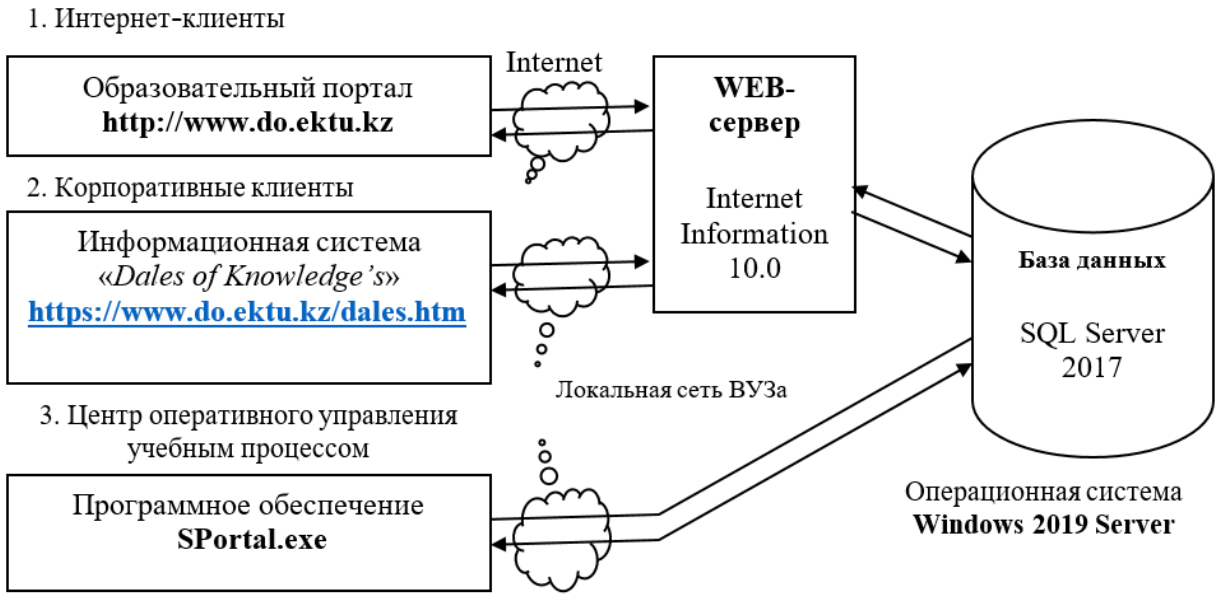


Рисунок 4.1 – Архитектура образовательного портала

Использование архитектуры тонкого клиента, при котором все компоненты размещены на сервере, позволяет минимизировать трафик, как со стороны клиента, так и со стороны сервера. Данная схема позволяет разделить командные функции управления и функции обеспечения и формирования выводов. Благодаря гибкости предложенной схемы можно

вносить изменения в одну компоненту без коррекции или с минимальной коррекцией другой. Кроме того, упрощается синтез системы управления учебным процессом, так как каждый блок можно проектировать относительно независимо, соблюдая лишь спецификации интерфейса между блоками.

Использование веб-интерфейса для доступа к образовательному порталу университета дает много преимуществ в виде универсальности, возможности удаленной работы, интерактивности.

На рисунке 4.2 представлена программно-аппаратная структура Образовательного портала ВКТУ им.Д.Серикбаева. В структуру указанного образовательного портала входят следующие основные компоненты:

а) Система управления базами данных. Данный компонент предназначен для обеспечения функционирования реляционной базы данных для образовательного портала. В нашем случае в качестве сервера управления базами данных выступает Microsoft SQL Server 2017. Основными базами данных образовательного портала являются:

- SPortal – центральная база данных, которая хранит данные, связанные с обеспечение учебного процесса (контингент, учебные планы, дисциплины, ППС и др.)

- Файловое хранилище – база данных, предназначенная для хранения двоичных данных.

б) Программный комплекс SPortal. Данный программный комплекс представляет собой автономное windows-приложение для работы с базой данных образовательного портала в корпоративной среде ВУЗа. Данное приложение предназначено для административных служб университета – деканатов, кафедр, учебного управления для ведения данных, требуемых для организации учебного процесса и различных справочных данных. Данный программный комплекс включает в себя такие подсистемы как:

- Структура ВУЗа – данная подсистема предназначена для хранения организационной структуры ВУЗа и кадрового состава.

- Учебные планы – данная подсистема предназначена для хранения сведений о различных учебных планах, по которым проводится обучение в ВУЗе, формирование и распределение академических потоков и нагрузки ППС.

- Группы и обучающиеся – данная подсистема предназначена для ведения учета контингента ВУЗа

- Расписание – данная подсистема предназначена для составления расписания занятий и экзаменов

- Справочники – данная подсистема предназначена для ведения различных справочников, которые требуются для работы различных подсистем образовательного портала.

- Администрирование – данная подсистема предназначена для конфигурирования различных подсистем образовательного портала.

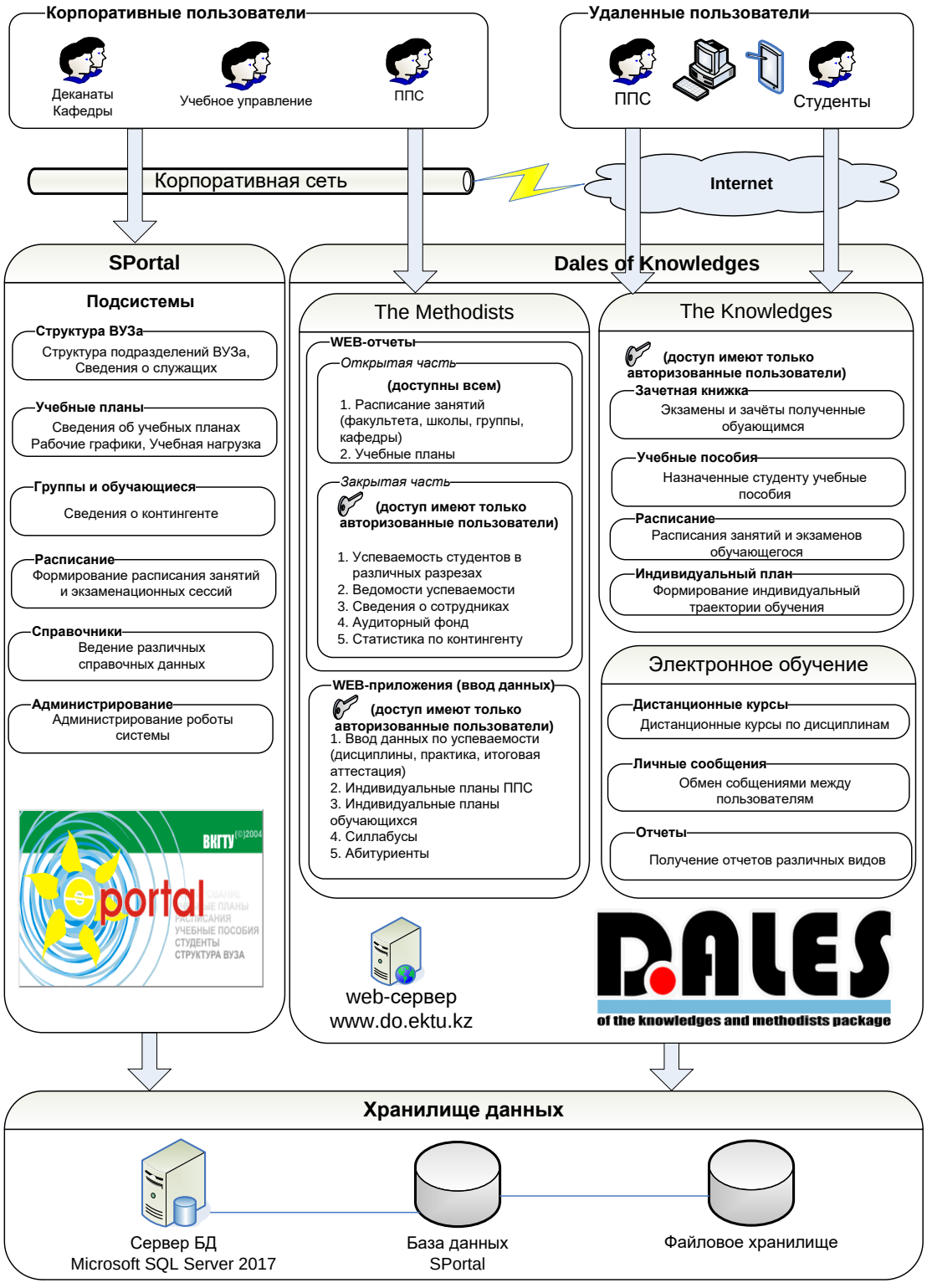


Рисунок 4.2 – Структура образовательного портала

Программный комплекс SPortal был разработан с использованием среды программирования Embarcadero Delphi 2009. Данная среда имеет продвинутые средства работы с базами данным и широкий набор средств для отображения и редактирования данных.

Взаимодействие с сервером баз данных Microsoft SQL Server 2017 строится на основе технологии ActiveX Data Object (ADO) с использованием провайдера данных Microsoft OLEDB Provider for SQL Server. Схема работы приложения с базой данной представлена на рисунке 4.3.

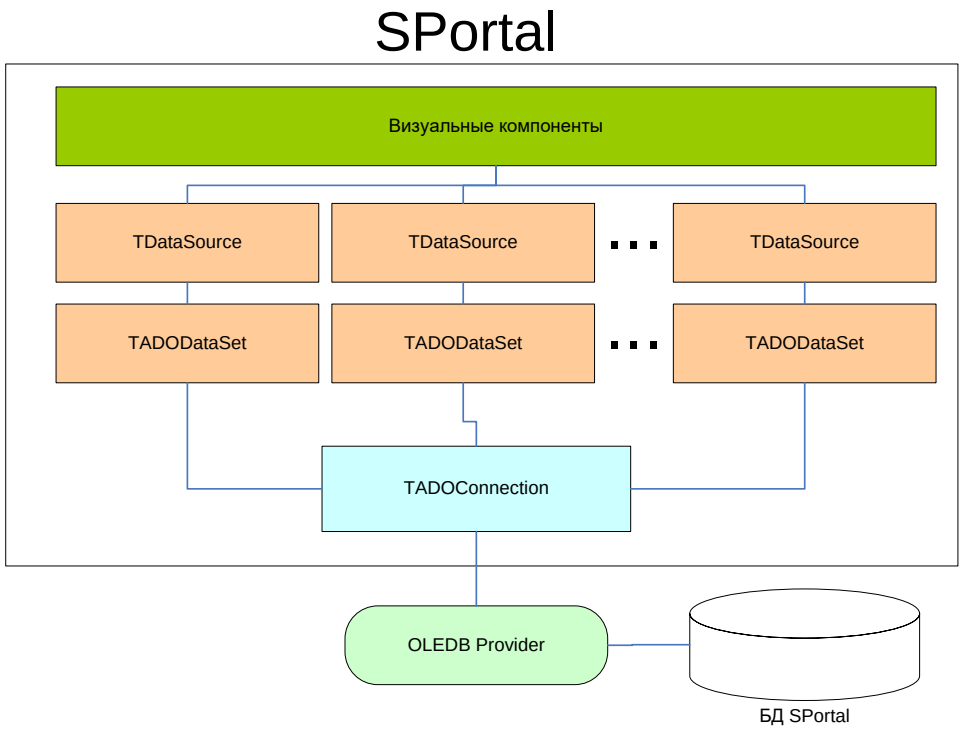


Рисунок 4.3 – Схема работы ПО SPortal с базой данных

Как видно из схемы, представленной на рисунке ниже, для подключения к базе данных используется компонент TADOConnection, который реализует подключение к базе данных посредством OLEDB провайдера. С данным компонентом взаимодействуют компоненты TADODataSet, который представляет собой набор данных в базе данных. Для взаимодействия с набором данным находящимся в TADODataSet и визуальными компонентами используется компоненты TDataSource. Данный компонент отвечает за извлечение данных из источника данных, передачу их визуальных компонентов для отображения, а также для передачи обновленных данных обратно в набор данных.

в) Web-службы Dales: The Knowledges. Данные службы представляют собой комплекс веб-приложений, которые предоставляют точку доступа к различному функционалу образовательного портала. Указанные веб-службы разделены на следующие категории:

- The Methodist – данные веб-службы предназначены для работы ППС и административного персонала с образовательным порталом. Данная часть разделена на:

1) открытая часть, которая доступна для всех пользователей без авторизации. В данную часть включены отчеты по расписанию занятий и учебным планам,

2) закрытая часть, которая доступна только для авторизованных сотрудников ВУЗа. В данную часть входят отчеты по успеваемости, контингенты, аудиторному фонду и др.,

3) веб-приложения, которые предназначены для ввода данных в базу данных образовательного портала различными сотрудниками ВУЗа. В данную часть входят журналы успеваемости и посещаемости, ведомости успеваемости, данные по абитуриентам, индивидуальные планы ППС и обучающихся и др.

- The knowledges – представляет собой личный кабинет обучающегося, где он может получить сведения о своей успеваемости, расписанию, плану обучения и др.

- Электронное обучение – представляет собой систему электронного обучения (LMS), в которой проводится обучение по различным курса ППС и обучающихся.

г) Службы каталога. Данный компонент архитектуры предназначен для хранения данных о пользователях, компьютеров, групп пользователей, членства в группах и проведения аутентификации и авторизации пользователей системы. В нашем случае в качестве службы каталогов выступает служба каталогов Microsoft Active Directory. Данная служба предоставляет данные на основе домена ektu.kz.

Как видно из описания образовательного портала ВКТУ им.Д.Серибаева в его составе имеются все необходимые компоненты, требуемые для составления расписания различных видов. Рассмотрим более подробно данные компоненты.

4.2. Модули образовательного портала ВКТУ им.Д.Серикбаева используемые для составления расписания

Для составления расписания на образовательном портале ВКТУ им.Д.Серикбаева имеются следующие компоненты, которые входят в состав программного комплекса SPortal:

а) Учебные планы – данный модуль содержит сведения об изучаемых дисциплинах по специальностям в разрезе периодов обучения, с указанием часов по видам занятий – лекции, практики, лабораторные.

б) Рабочий график – данный модуль содержит сведения по назначению ППС на различные виды занятий (лекции, практики, лабораторные) на изучаемые в течение семестра дисциплины.

в) Аудиторный фонд – данный модуль содержит сведения по аудиторному фонду университета.

г) Модуль «Расписание занятий». Данный модуль предназначен для составления расписания занятий на семестр. С помощью данного модуля

имеется возможность составить расписание для группы и преподавателя. Расписание занятий составляется на неделю. Внешний вид данного модуля представлен на рисунке 4.4.

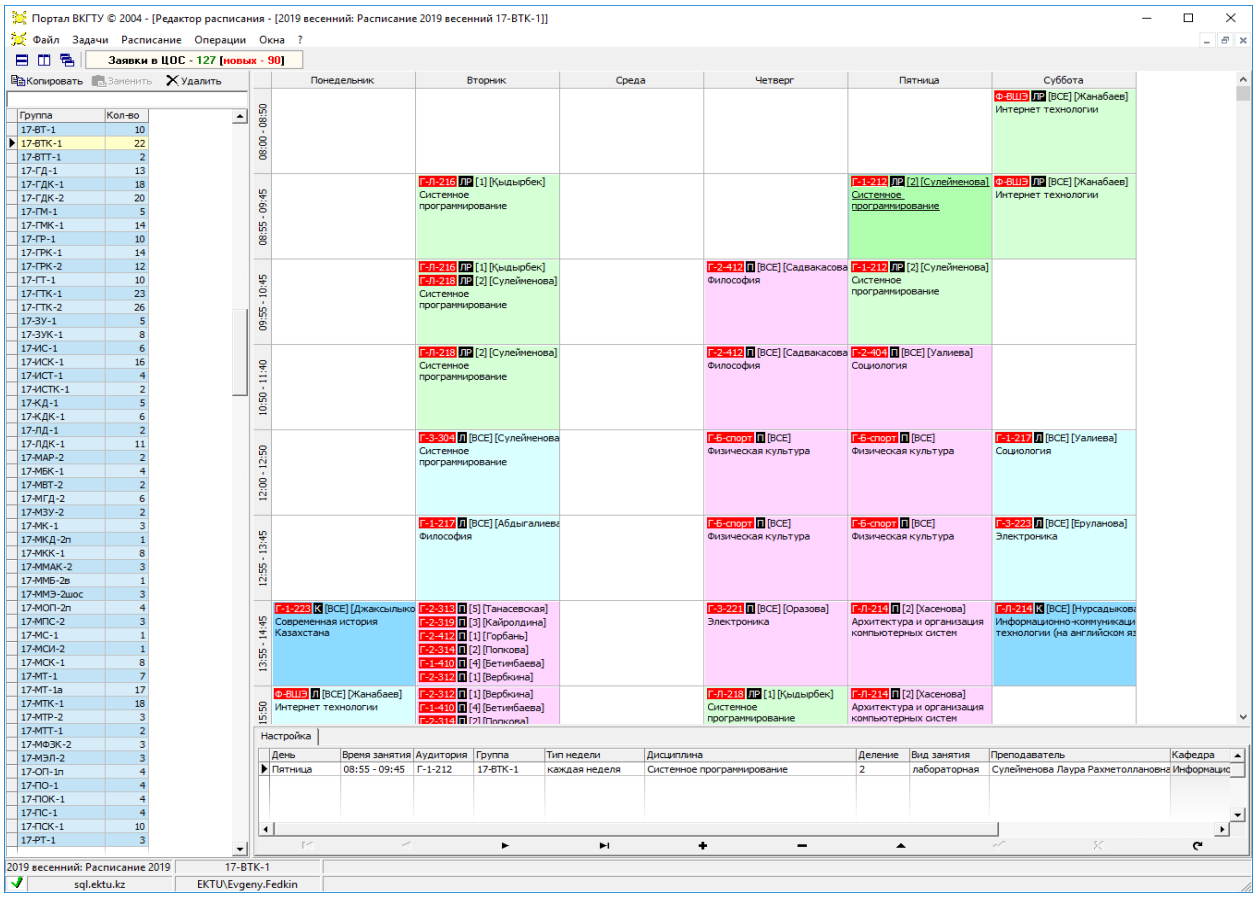


Рисунок 4.4 – Модуль «Расписание занятий»

д) Модуль «Расписание сессии». Данный модуль предназначен для составления расписания экзаменационной сессии на семестр. С помощью данного модуля имеется возможность составить расписание сессии для очного и заочного отделений. Расписание составляется по календарным дням. Внешний вид данного модуля приведен на рисунке 4.5.

Просмотр составленного расписания доступно в веб-службах Dales в различных разрезах в различных разрезах:

- по группе (рисунок 4.6),
- по преподавателю (рисунок 4.7),
- по аудитории (рисунок 4.8) и др.

Анализ данных, используемых в перечисленных выше модулях, показал, что их можно применять для составления расписания занятий с использованием, разработанных в 3-й главе данного исследования программных компонентов для составления расписания занятий.

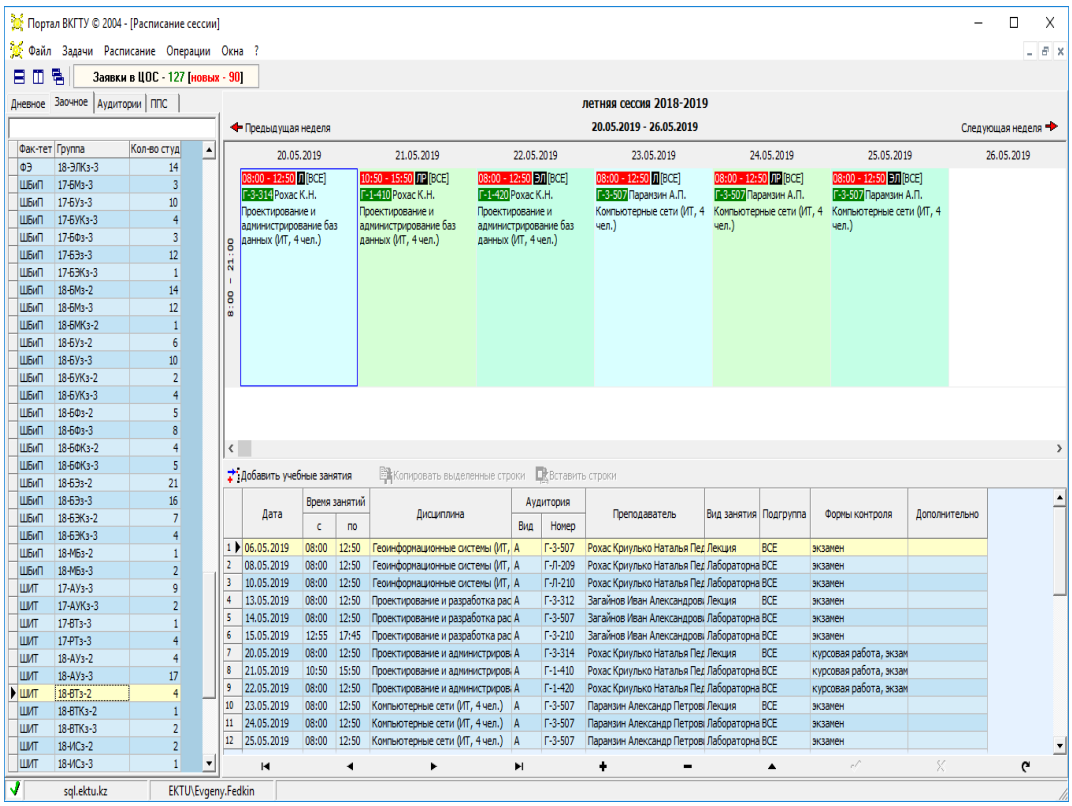


Рисунок 4.5 – Модуль «Расписание сессии»

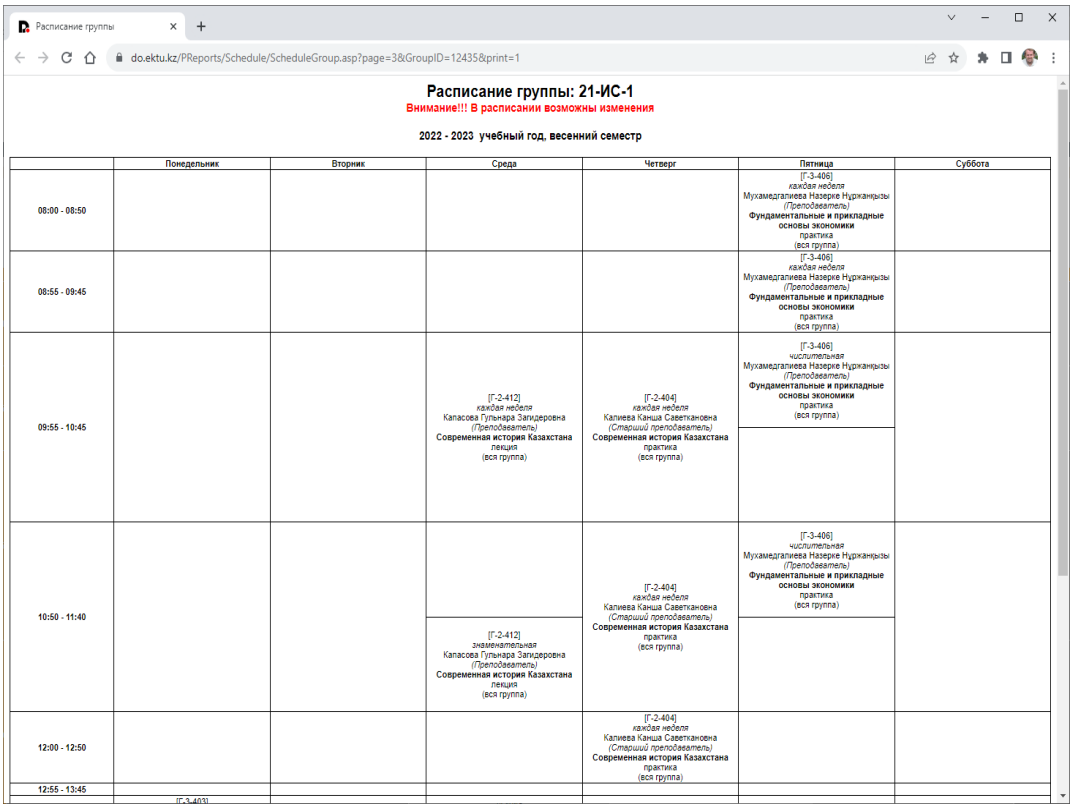
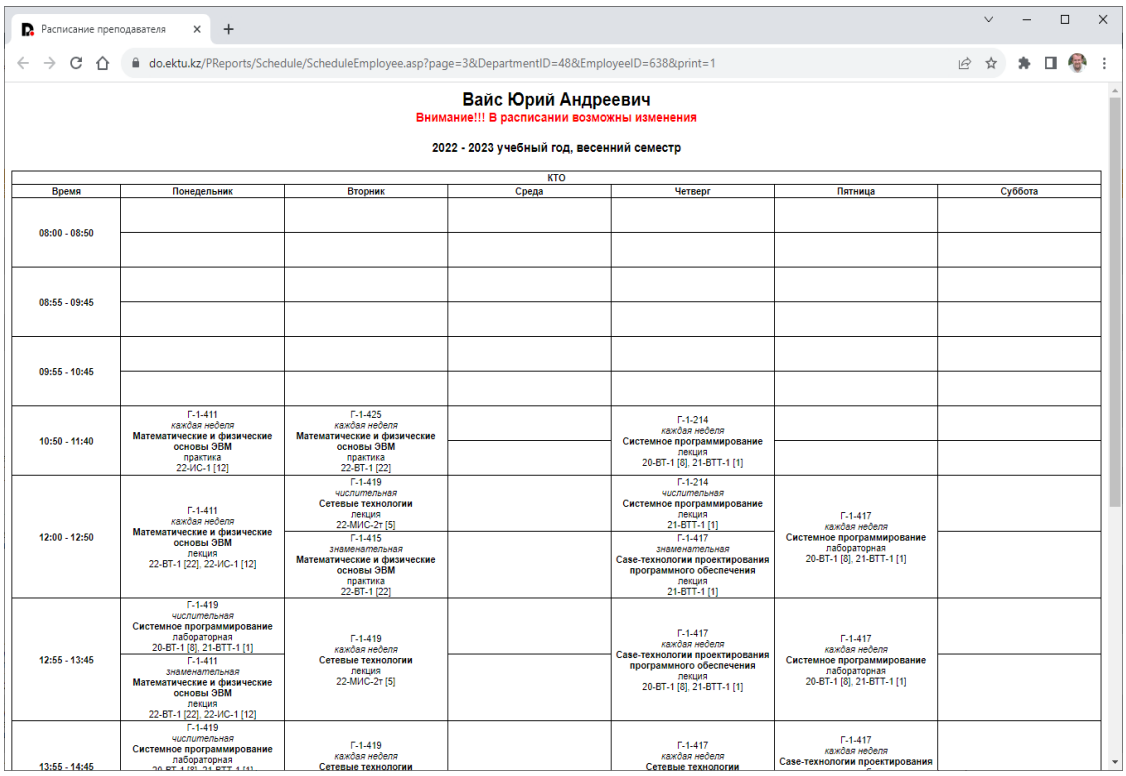


Рисунок 4.6 – Расписание группы



Расписание преподавателя

do.ektu.kz/PReports/Schedule/ScheduleEmployee.asp?page=3&DepartmentID=48&EmployeeID=638&print=1

Вайс Юрий Андреевич

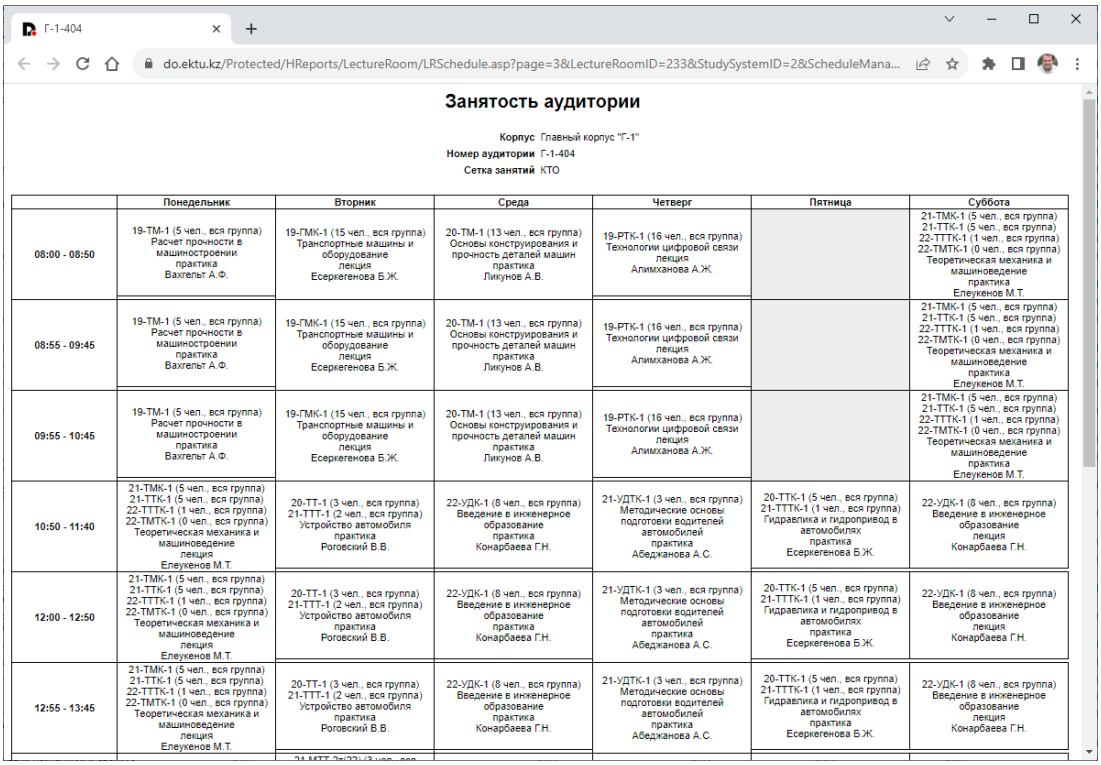
Внимание!!! В расписании возможны изменения

2022 - 2023 учебный год, весенний семестр

КТО

Время	Понедельник	Вторник	Среда	Четверг	Пятница	Суббота
08:00 - 08:50						
08:55 - 09:45						
09:55 - 10:45						
10:50 - 11:40	Г-1-411 *каждая неделя* **Математические и физические основы ЭВМ** практика 22-ИС-1 [12]	Г-1-425 *каждая неделя* **Математические и физические основы ЭВМ** практика 22-ВТ-1 [22]		Г-1-214 *каждая неделя* **Системное программирование** лекция 20-ВТ-1 [8], 21-ВТТ-1 [1]		
12:00 - 12:50	Г-1-411 *каждая неделя* **Математические и физические основы ЭВМ** лекция 22-ВТ-1 [22], 22-ИС-1 [12]	Г-1-419 *числительная* **Сетевые технологии** лекция 22-МИС-2т [5]		Г-1-214 *числительная* **Системное программирование** лекция 21-ВТТ-1 [1]	Г-1-417 *каждая неделя* **Системное программирование** лабораторная 20-ВТ-1 [8], 21-ВТТ-1 [1]	
		Г-1-415 *знаменательная* **Математические и физические основы ЭВМ** практика 22-ВТ-1 [22]		Г-1-417 *знаменательная* **Case-технологии проектирования программного обеспечения** лекция 21-ВТТ-1 [1]		
12:55 - 13:45	Г-1-419 *числительная* **Системное программирование** лабораторная 20-ВТ-1 [8], 21-ВТТ-1 [1]	Г-1-419 *каждая неделя* **Сетевые технологии** лекция 22-МИС-2т [5]		Г-1-417 *каждая неделя* **Case-технологии проектирования программного обеспечения** лекция 20-ВТ-1 [8], 21-ВТТ-1 [1]	Г-1-417 *каждая неделя* **Системное программирование** лабораторная 20-ВТ-1 [8], 21-ВТТ-1 [1]	
	Г-1-411 *знаменательная* **Математические и физические основы ЭВМ** лекция 22-ВТ-1 [22], 22-ИС-1 [12]					
13:55 - 14:45	Г-1-419 *числительная* **Системное программирование** лабораторная	Г-1-419 *каждая неделя* **Сетевые технологии**		Г-1-417 *каждая неделя* **Сетевые технологии**	Г-1-417 *каждая неделя* **Case-технологии проектирования**	

Рисунок 4.7 – Расписание преподавателя



Г-1-404

do.ektu.kz/Protected/HReports/LectureRoom/LRSchedule.asp?page=3&LectureRoomID=233&StudySystemID=2&ScheduleMana...

Занятость аудитории

Корпус Главный корпус "Г-1"
Номер аудитории Г-1-404
Сетка занятий КТО

	Понедельник	Вторник	Среда	Четверг	Пятница	Суббота
08:00 - 08:50	19-ТМ-1 (5 чел., вся группа) Расчет прочности в машиностроении практика Вахгельт А.Ф.	19-ГМК-1 (15 чел., вся группа) Транспортные машины и оборудование лекция Есеркегенова Б.Ж.	20-ТМ-1 (13 чел., вся группа) Основы конструирования и прочность деталей машин практика Ликунов А.В.	19-РТК-1 (16 чел., вся группа) Технологии цифровой связи лекция Алимханова А.Ж.		21-ТМК-1 (5 чел., вся группа) 21-ТТК-1 (5 чел., вся группа) 22-ТТТК-1 (1 чел., вся группа) 22-ТМТК-1 (0 чел., вся группа) Теоретическая механика и машиноведение практика Елеукенов М.Т.
08:55 - 09:45	19-ТМ-1 (5 чел., вся группа) Расчет прочности в машиностроении практика Вахгельт А.Ф.	19-ГМК-1 (15 чел., вся группа) Транспортные машины и оборудование лекция Есеркегенова Б.Ж.	20-ТМ-1 (13 чел., вся группа) Основы конструирования и прочность деталей машин практика Ликунов А.В.	19-РТК-1 (16 чел., вся группа) Технологии цифровой связи лекция Алимханова А.Ж.		21-ТМК-1 (5 чел., вся группа) 21-ТТК-1 (5 чел., вся группа) 22-ТТТК-1 (1 чел., вся группа) 22-ТМТК-1 (0 чел., вся группа) Теоретическая механика и машиноведение практика Елеукенов М.Т.
09:55 - 10:45	19-ТМ-1 (5 чел., вся группа) Расчет прочности в машиностроении практика Вахгельт А.Ф.	19-ГМК-1 (15 чел., вся группа) Транспортные машины и оборудование лекция Есеркегенова Б.Ж.	20-ТМ-1 (13 чел., вся группа) Основы конструирования и прочность деталей машин практика Ликунов А.В.	19-РТК-1 (16 чел., вся группа) Технологии цифровой связи лекция Алимханова А.Ж.		21-ТМК-1 (5 чел., вся группа) 21-ТТК-1 (5 чел., вся группа) 22-ТТТК-1 (1 чел., вся группа) 22-ТМТК-1 (0 чел., вся группа) Теоретическая механика и машиноведение практика Елеукенов М.Т.
10:50 - 11:40	21-ТМК-1 (5 чел., вся группа) 21-ТТК-1 (5 чел., вся группа) 22-ТТТК-1 (1 чел., вся группа) 22-ТМТК-1 (0 чел., вся группа) Теоретическая механика и машиноведение лекция Елеукенов М.Т.	20-ТТ-1 (3 чел., вся группа) 21-ТТТ-1 (2 чел., вся группа) Устройство автомобиля практика Роговский В.В.	22-УДК-1 (8 чел., вся группа) Введение в инженерное образование практика Конарбаева Г.Н.	21-УДТК-1 (3 чел., вся группа) Методические основы подготовки водителей автомобилей практика Абеджанова А.С.	20-ТТК-1 (5 чел., вся группа) 21-ТТТК-1 (1 чел., вся группа) Гидравлика и гидропривод в автомобилях практика Есеркегенова Б.Ж.	22-УДК-1 (8 чел., вся группа) Введение в инженерное образование лекция Конарбаева Г.Н.
12:00 - 12:50	21-ТМК-1 (5 чел., вся группа) 21-ТТК-1 (5 чел., вся группа) 22-ТТТК-1 (1 чел., вся группа) 22-ТМТК-1 (0 чел., вся группа) Теоретическая механика и машиноведение лекция Елеукенов М.Т.	20-ТТ-1 (3 чел., вся группа) 21-ТТТ-1 (2 чел., вся группа) Устройство автомобиля практика Роговский В.В.	22-УДК-1 (8 чел., вся группа) Введение в инженерное образование практика Конарбаева Г.Н.	21-УДТК-1 (3 чел., вся группа) Методические основы подготовки водителей автомобилей практика Абеджанова А.С.	20-ТТК-1 (5 чел., вся группа) 21-ТТТК-1 (1 чел., вся группа) Гидравлика и гидропривод в автомобилях практика Есеркегенова Б.Ж.	22-УДК-1 (8 чел., вся группа) Введение в инженерное образование лекция Конарбаева Г.Н.
12:55 - 13:45	21-ТМК-1 (5 чел., вся группа) 21-ТТК-1 (5 чел., вся группа) 22-ТТТК-1 (1 чел., вся группа) 22-ТМТК-1 (0 чел., вся группа) Теоретическая механика и машиноведение лекция Елеукенов М.Т.	20-ТТ-1 (3 чел., вся группа) 21-ТТТ-1 (2 чел., вся группа) Устройство автомобиля практика Роговский В.В.	22-УДК-1 (8 чел., вся группа) Введение в инженерное образование практика Конарбаева Г.Н.	21-УДТК-1 (3 чел., вся группа) Методические основы подготовки водителей автомобилей практика Абеджанова А.С.	20-ТТК-1 (5 чел., вся группа) 21-ТТТК-1 (1 чел., вся группа) Гидравлика и гидропривод в автомобилях практика Есеркегенова Б.Ж.	22-УДК-1 (8 чел., вся группа) Введение в инженерное образование лекция Конарбаева Г.Н.

Рисунок 4.8 – Расписание аудитории

Далее рассмотрим, как используя данные в рассмотренных нами модулях образовательного портала, можно составлять расписания на основе предложенного в данном исследовании алгоритма.

4.3 Анализ базы данных образовательного портала для определения данных, требуемых для составления расписания занятий

Для реализации генерации расписания занятий на основе образовательного портала ВКТУ нами была проанализирована база данных портала и были выявлены таблицы с требуемыми данными. Схема требуемых таблиц базы данных приведена на рисунке 4.9.

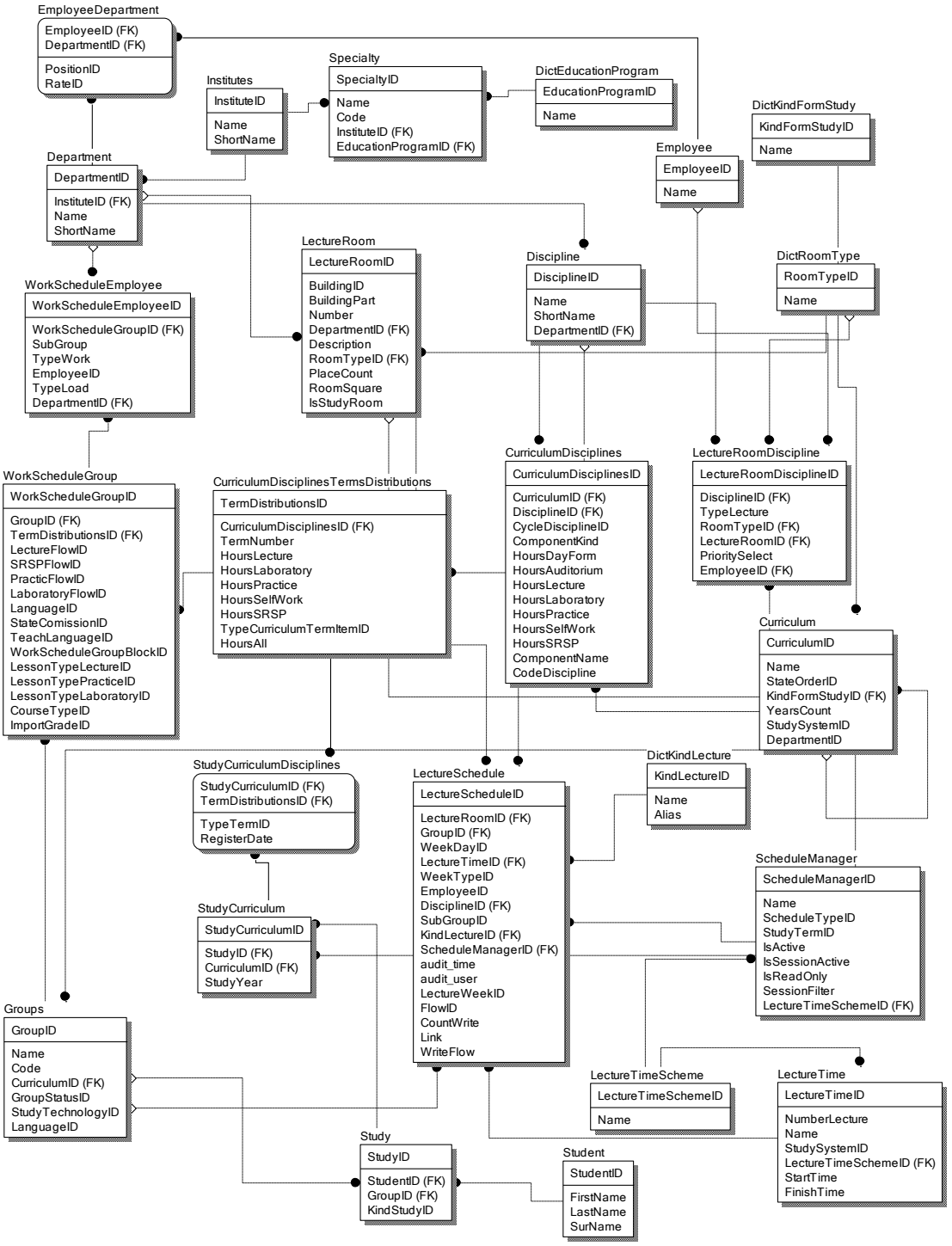


Рисунок 4.9 – Схема таблиц базы данных образовательного портала

На приведенной на рисунке 4.9 приведены следующие таблицы из базы данных образовательного портала:

- Curriculum – данная таблица содержит сведения о рабочих учебных планах,
- CurriculumDisciplines – данная таблица содержит сведения о дисциплинах, входящих в рабочий учебный план,
- CurriculumDisciplinesTermsDistributions – данная таблица содержит сведения о распределении дисциплин рабочего учебного по учебным периодам (семестрам, триместрам и др.)
- Department – данная таблица содержит сведения о кафедрах,
- DictEducationProgram – данная таблица содержит справочник об уровнях обучения (бакалавриат, магистратура, докторантура и др.)
- DictKindFormStudy – данная таблица содержит справочник о формах обучения,
- DictKindLecture – данная таблица содержит справочник о видах занятиях в расписании,
- DictRoomType – данная таблица содержит справочник типов аудиторий
- Discipline – данная таблица содержит справочник дисциплин,
- Employee – данная таблица содержит сведения о преподавателях,
- EmployeeDepartment – данная таблица содержит сведения о закреплении преподавателей за кафедрами,
- Groups – данная таблица содержит сведения об учебных группах,
- Institutes – данная таблица содержит сведения о факультетах и школах,
- LectureRoom – данная таблица содержит сведения об учебных аудиториях,
- LectureRoomDiscipline – данная таблица содержит сведения о закреплении учебных аудиторий за дисциплинами,
- LectureSchedule – данная таблица содержит расписание занятий,
- LectureTime – данная таблица содержит данные по временным интервалам проведения учебных занятий
- LectureTimeScheme – данная таблица содержит данные по группам временных интервалов
- ScheduleManager – данная таблица содержит данные о расписаниях по учебным периодам,
- Specialty – данная таблица содержит справочник образовательных программ,
- Student – данная таблица содержит сведения о обучающихся,
- Study – данная таблица содержит сведения о закреплении обучающихся за учебными группами,
- StudyCurriculum – данная таблица содержит сведения об индивидуальных учебных планах обучающихся,
- StudyCurriculumDisciplines – данная таблица содержит сведения о дисциплинах, включенных в индивидуальный учебный план обучающегося,

- WorkScheduleGroup – данная таблица содержит сведения о закреплении учебных групп и дисциплин учебного плана за академическими потоками,

- WorkScheduleEmployee – данная таблица содержит сведения о закреплении преподавателей за академическими потоками.

Данные из приведенных выше таблиц могут быть импортированы в базу данных составления расписания (раздел 3.1) и использоваться для составления расписаний занятий. В приложении Е приведены SQL-запросы для извлечения данных из базы данных образовательного портала в базу данных для составления расписания. С помощью данных скриптов в базу данных для составления расписания были извлечены данные для составления расписания занятий на 3 триместр 2022-2023 учебного года.

4.4 Программная реализация составления расписания занятий

Для программной реализации составления расписания на основе данных образовательного портала нами был взят за основу шаблон приложения из приложения Г и на его основе был заданы следующие параметры работы:

а) размер начальной популяции – 10,

б) размер популяции – 20,

в) реализована логика для отбора аудиторий для академических потоков,

г) реализована логика для отбора временных интервалов и определения приоритета их выбора для академических потоков,

д) определены ограничения для расписания.

Для составления расписания занятий мы использовали настройки ограничений, которые приведены в таблице 4.1

Таблица 4.1

Настройки ограничений для составления расписания занятий

Класс ограничения	Весовой коэффициент	Описание
«жесткие» ограничения		
RestrictEmployee	8000	Проверка накладок для преподавателей
RestrictGroup	15000	Проверка накладок для группы
RestrictRoom	7000	Проверка накладок для аудитории
RestrictOneDayDisciplines	5000	Проверка повторения занятия по дисциплине и виду занятия в течение дня для группы
RestrictGroupHours	5000	Проверка ограничения на количество занятий для группы – не более 8 часов
RestrictGroupFill	2000	Проверка разрывов в расписании группы в течение дня
«мягкие» ограничения		
RestrictGroupHours	50	Проверка ограничения на количество занятий для группы – не более 8 часов и не менее 2 часов
RestrictDayDiscipline	500	Проверка на количество различных дисциплин в течение дня у группы – не более

Класс ограничения	Весовой коэффициент	Описание
		3 лекций и не более 3 практических и лабораторных занятий
RestrictEmployeeFill	2	Проверка разрывов в расписании преподавателя в течение дня
RestrictEmployeeHours	500	Проверка ограничения на число часов в расписании преподавателя в течение дня – не более 9 часов

Исходный код разработанного приложения для составления расписания для 3-го триместра 2022-2023 учебного года приведен в приложении Ж.

Разработанное приложение выполняет следующую последовательность действий при своей работе:

а) загружает данные для составления расписания из базы данных.

б) производит определение возможных комбинаций дней и временных интервалов для каждого учебного потока.

в) производит определение возможных аудиторий для каждого учебного потока.

г) производит сортировку учебных потоков на основе алгоритма предложенного во 2-м разделе данного исследования.

д) производится генерация начальной популяции (рисунок 4.10). Полученная начальная популяция сохраняется в виде xml-файлов.

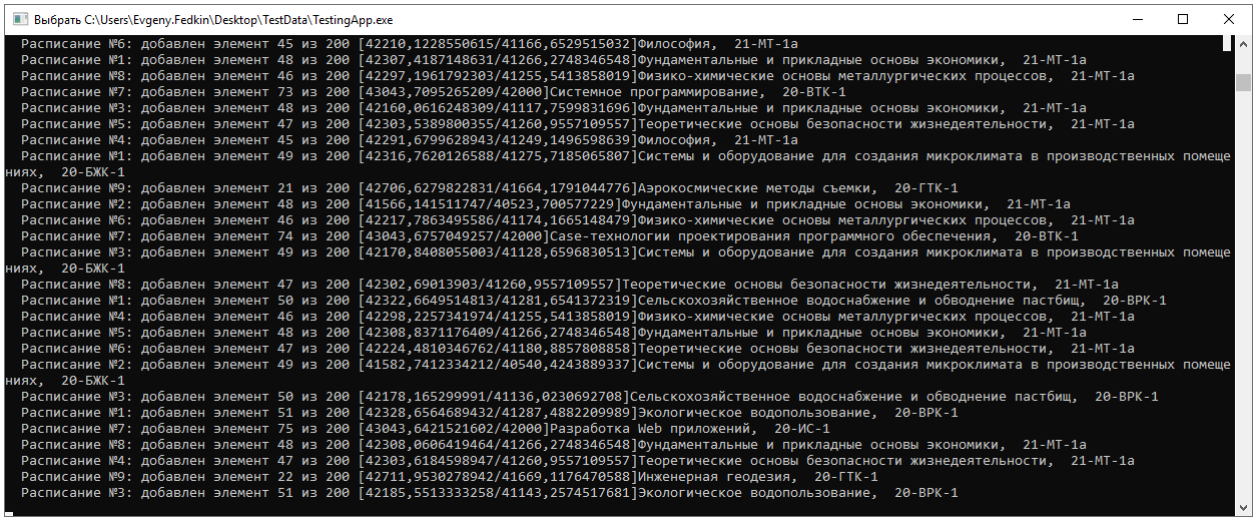


Рисунок 4.10 – Вывод программы при генерации начальной популяции

е) после завершения генерации начальной популяции запускается циклический процесс поиска оптимального решения на основе предложенной модификации генетического алгоритма в данном исследовании (рисунок 4.11).

ж) После завершения каждого шага поиска решения программа выводит таблицу со списком особей в популяции и значениями целевой функции

(рисунок 4.12), а также сохраняет полученную популяцию в xml-файлах в папке под номером шага.

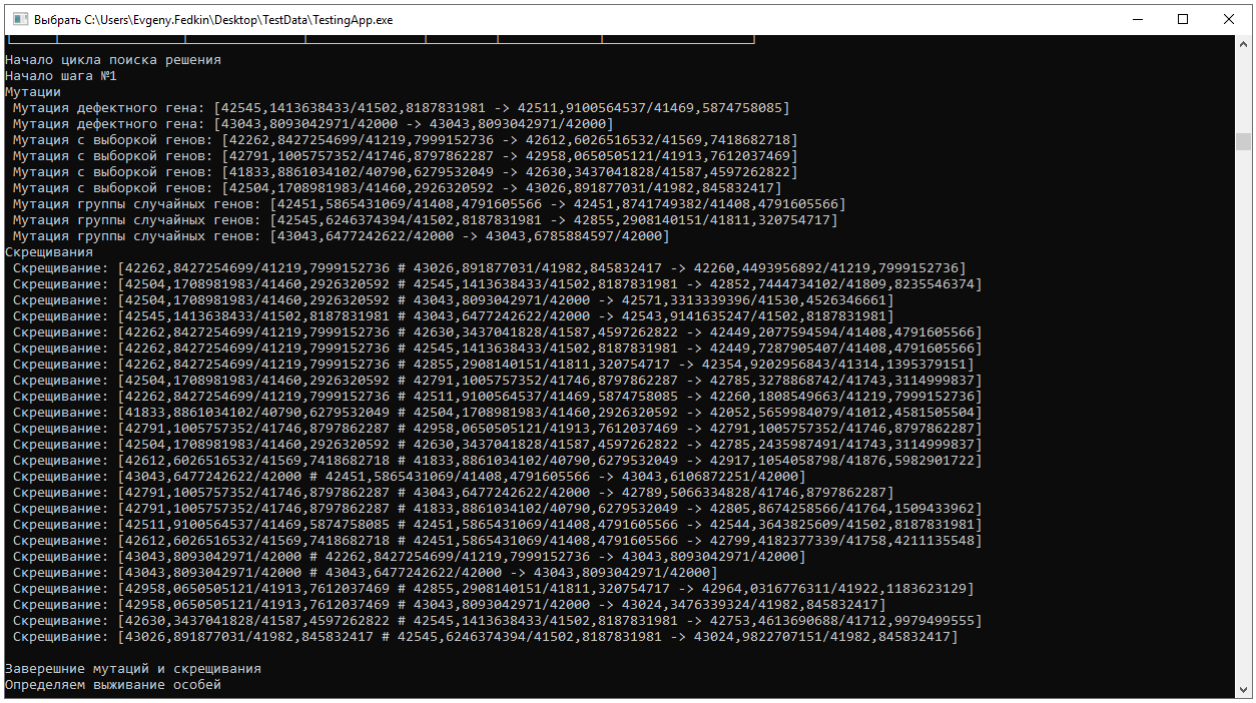


Рисунок 4.11 – Вывод программы применения генетических операторов

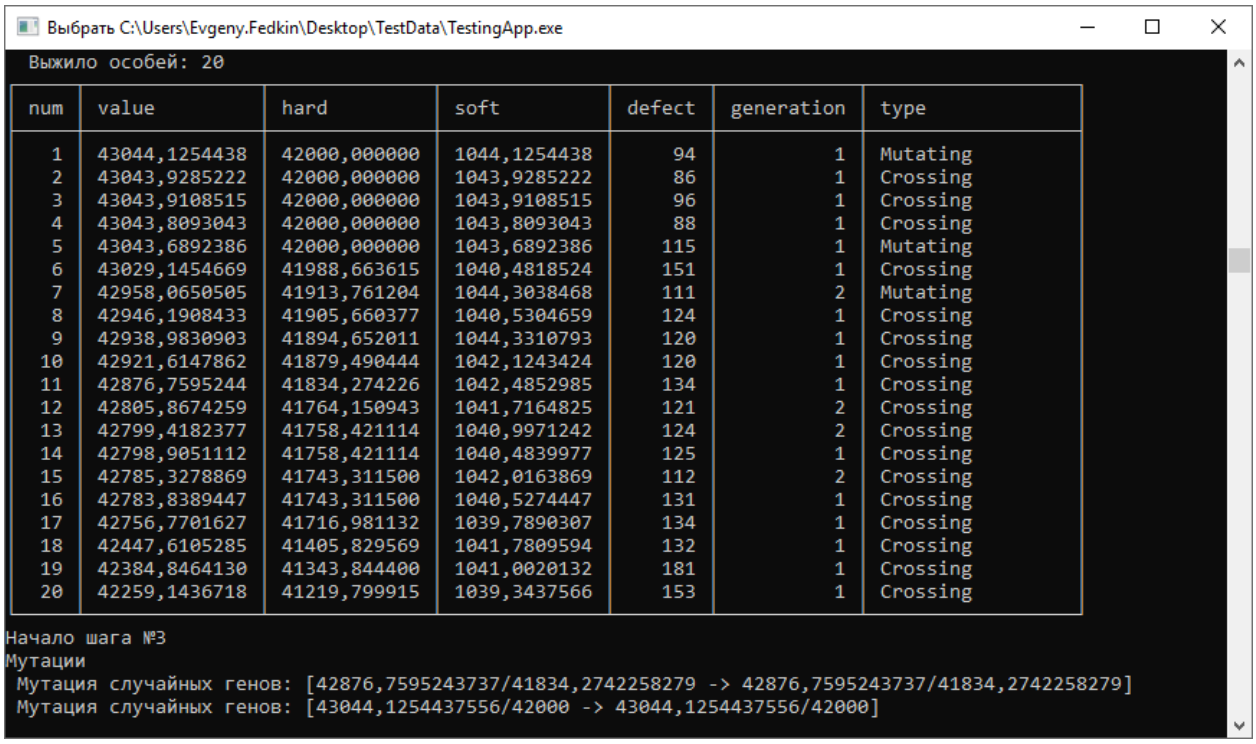


Рисунок 4.12 – Вывод таблицы с данными по завершению циклического шага генетического алгоритма.

Как было указано выше, каждое расписание из популяции сохранятся в виде xml-файла. Создаваемый xml-файл имеет следующую структуру:

а) Элемент расписания записывается в виде элемента с именем «s» в корневом элементе и содержит следующие атрибуты:

- day – идентификатор дня недели
- dayName – название дня недели
- room – идентификатор аудитории
- roomNumber – название аудитории
- type – вид занятия
- flowid – идентификатор потока

б) Временной интервал записывается в виде элемента с именем «lt» внутри элемента расписания. Данный элемент содержит следующий набор атрибутов:

- id – идентификатор временного интервала
- num – порядковый номер временного интервала
- start – время начала занятия
- finish – время завершения занятия

в) Сведения о преподавателе ведущий занятие записывается в виде элемента с именем «t» внутри элемента расписания. Данный элемент содержит следующий набор атрибутов:

- id – идентификатор преподавателя
- name – ФИО преподавателя

г) Сведения о академической группе, для которой проводится занятия записывается в виде элемента с именем «g» внутри элемента расписания. Данный элемент содержит следующий набор атрибутов:

- id – идентификатор учебной группы
- code – шифр учебной группы
- sub – подгруппа
- disc – идентификатор дисциплины
- discname – наименование дисциплины.

д) Сведения о нарушениях ограничений в элементе расписания записываются в виде элемента с именем «defect» внутри элемента расписания. Данный элемент содержит следующий набор атрибутов:

- name – название нарушения
- hard – логического значение определяющее относится ли ограничение к «жесткому» ограничению.

е) Значение отдельного вида ограничения записывается в виде элемента с именем «val». Данный элемент содержит следующий набор атрибутов:

- name – наименование ограничения
- value – значение оценки для ограничения

Внешний вид создаваемого xml-файла для расписания приведен на рисунке 4.13.

Для предварительного просмотра созданного расписания занятий было разработан html-файл, который позволяет считать созданных xml-файл с

расписанием и просмотреть расписание по группам, преподавателю или аудитории (рисунок 4.14)

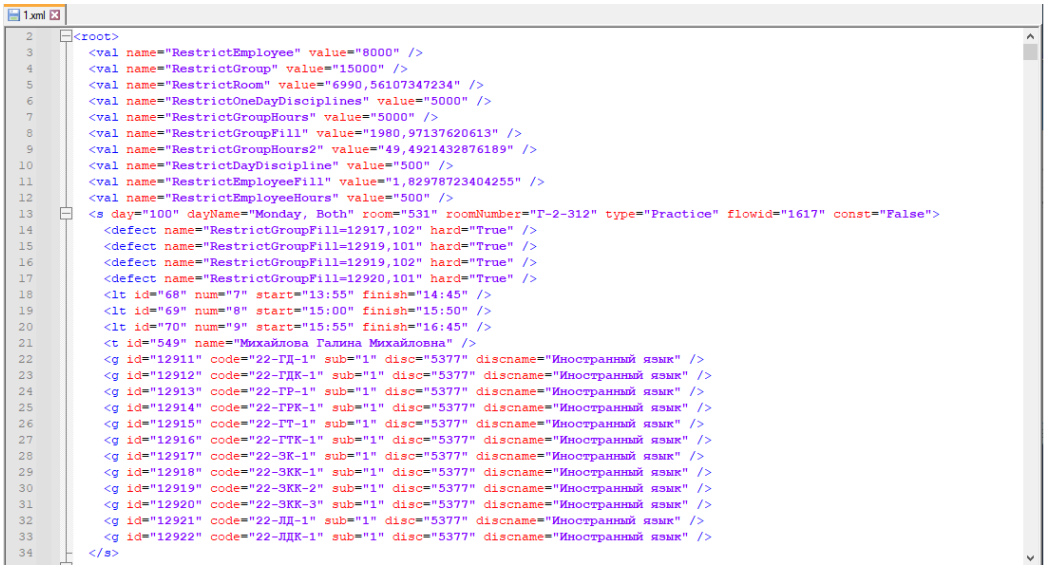


Рисунок 4.13 – xml-файл с расписанием

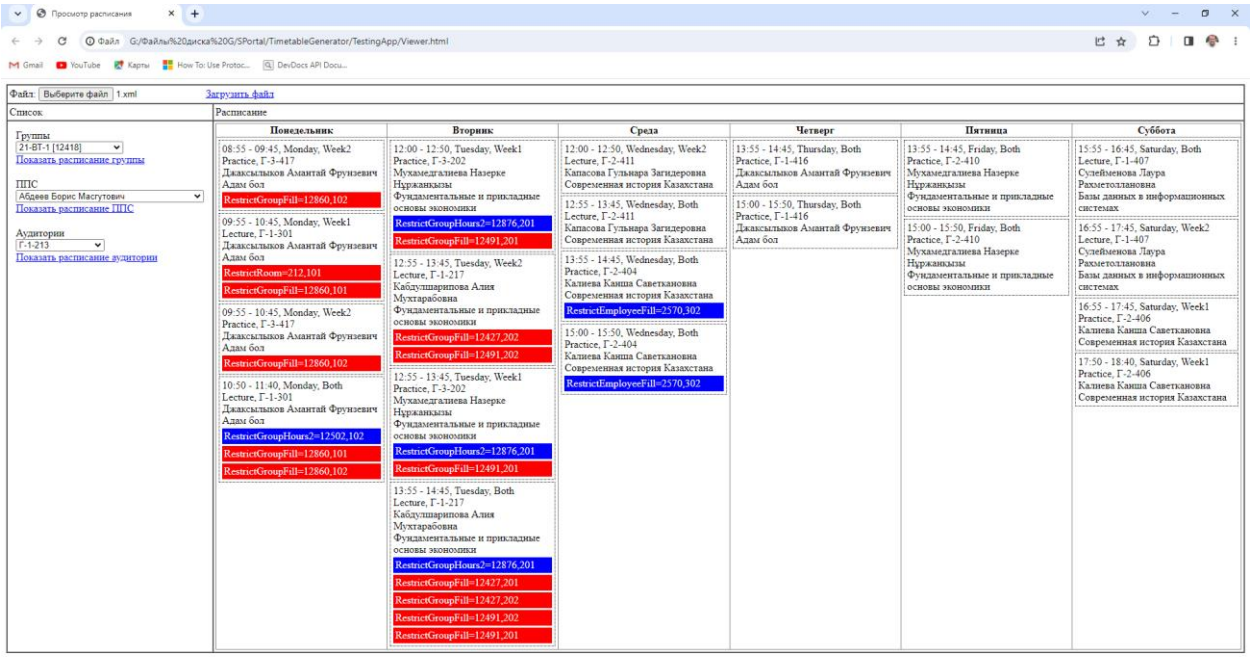


Рисунок 4.15 – Просмотр расписания

При просмотре расписания оно выводится по дням недели, а также выводится сведения о нарушении ограничений:

- красным цветом – нарушения «жестких» ограничений
- синим цветом – нарушения «мягких» ограничений.

Исходный код описанного выше html-файл приведен в приложении З.

Полученный в результате работы программы составления расписания xml-файл может быть импортирован в базу данных образовательного

портала. Для этого была разработана хранимая процедура импорта xml-файла в требуемые таблицы. Текст указанной процедуры приведен в приложении И.

4.5 Апробация приложения составления расписания

Для оценки эффективности разработанного алгоритма составления расписания занятий мы в рамках данного исследования провели вычислительный эксперимент [65, 102]. Данный вычислительный эксперимент проводился со следующими параметрами:

- Вычисление проводились на выделенном сервере с процессором, имеющим 8 ядер с тактовой частотой 2,3 ГГц, а также имеющий оперативную память в размере 16 Гб.
- Генерация расписания проводилась несколько раз с увеличением количества элементов в расписании. При первом прогоне программы из базы данных были извлечены данные по первым 200 академическим потокам, и по этим данным был произведен поиск решения. На каждом последующем шаге число извлекаемых данных по академическим потокам увеличивалось на 200 единиц. Таким образом, на последнем прогоне программы были представлены все академические потоки.
- Каждый прогон программы осуществлял построение начальной популяции и производил оптимизацию начальной популяции с использованием 100 циклов разработанного генетического алгоритма.

Результаты работы программы представлены в таблице 4.2

Таблица 4.2

Результаты прогона программы генерации расписания

№	Кол-во входных учебных потоков	Время работы	Кол-во временных интервалов	Кол-во ППС	Кол-во групп	Значение целевой функции						
						Начальная популяция	Финальная популяция	Прирост	% Прироста	Максимально-возможное значение	% начальной популяции от максимально-возможного	% финальной популяции от максимально-возможного
1	200	0:34:36	216	77	103	43043,96	43045,22	1,25	0,003%	43052	99,981%	99,984%
2	400	0:50:27	494	127	153	43035,28	43047,19	11,91	0,028%	43052	99,961%	99,989%
3	600	1:18:01	728	167	242	42997,92	43046,94	49,03	0,114%	43052	99,874%	99,988%
4	808	1:42:03	1056	196	259	42355,68	43046,64	690,96	1,631%	43052	98,383%	99,988%
5	1008	1:55:51	1420	207	259	42610,82	43049,29	438,47	1,029%	43052	98,975%	99,994%
6	1228	2:10:03	1884	221	259	42803,50	43046,88	243,38	0,569%	43052	99,423%	99,988%
7	1450	2:40:40	2328	231	261	42680,10	43047,77	367,66	0,861%	43052	99,136%	99,990%
8	1692	3:45:34	2862	239	262	42767,69	43034,45	266,76	0,624%	43052	99,340%	99,959%
9	1934	4:29:14	3384	249	262	42758,38	43022,85	264,47	0,619%	43052	99,318%	99,932%

Как видно из приведенной таблицы увеличение числа элементов в расписании приводит к увеличению времени генерации расписания. На рисунке 4.16 приведен график зависимости времени генерации расписания от количества элементов в расписании.

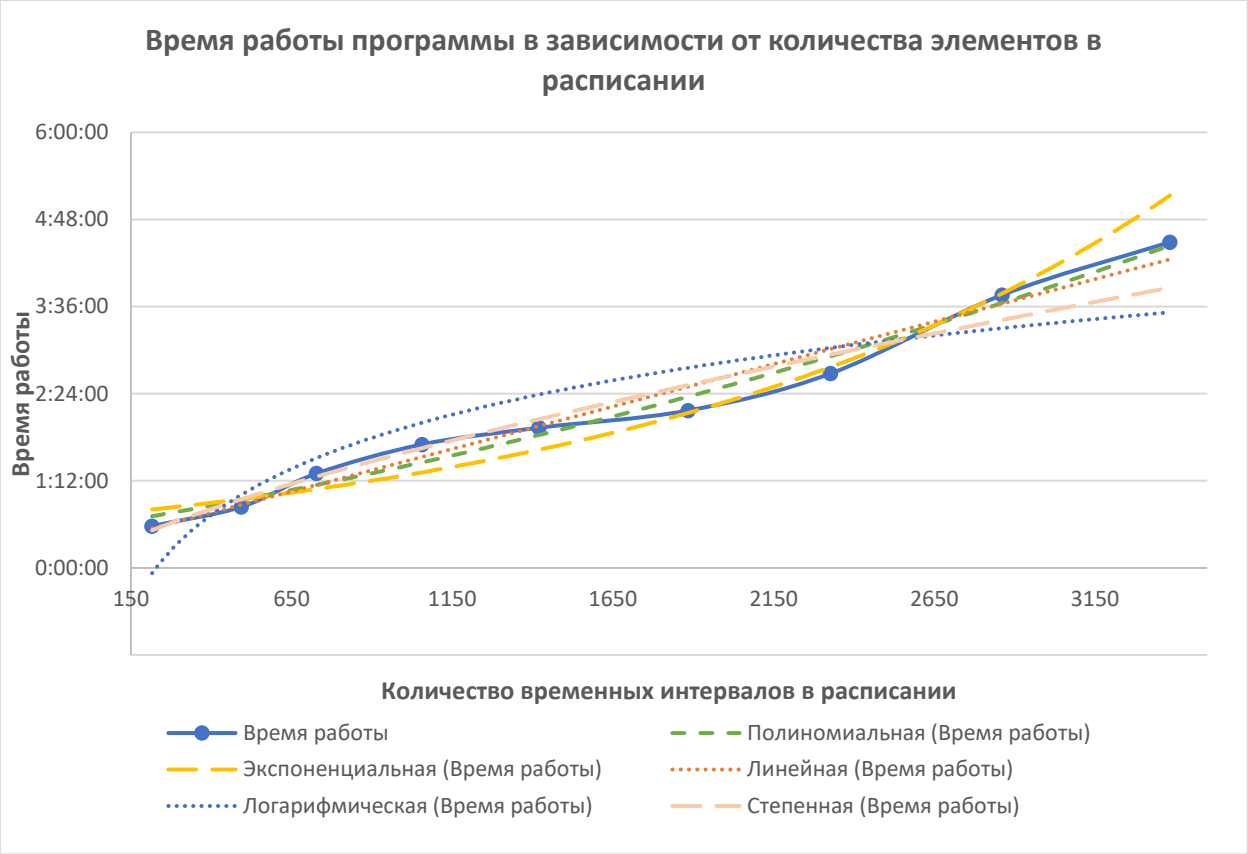


Рисунок 4.16 – Зависимость времени генерации расписания от количества элементов

Для оценки времени работы алгоритма в зависимости от количества элементов нами была произведена аппроксимация наших данных и были получены следующие результаты (таблица 4.3):

Таблица 4.3
Результаты аппроксимации

Вид аппроксимации	**Уравнение аппроксимации**	**Достоверность (R^2)**
Линейная	$y = 5 * 10^{-5}x + 0{,}0122$	0,9741
Полиномиальная	$y = 5 * 10^{-9}x^2 + 3 * 10^{-5}x + 0{,}0229$	0,9829
Экспоненциальная	$y = 0{,}029e^{0{,}0006x}$	0,9651
Логарифмическая	$y = 0{,}0544\ln(x) - 0{,}2957$	0,8216
Степенная	$y = 0{,}0004x^{0{,}7303}$	0,9559

Как видно из приведенных в таблице 4.3 данных, все виды аппроксимации имеют высокую степень достоверности, однако наиболее высокий уровень достоверности имеет полиномиальная аппроксимация. Таким образом, мы можем сделать вывод, что время работы разработанного алгоритма составления расписания имеет полиномиальное время работы от количества элементов в расписании.

Также на основе данных, приведенных в таблице 4.3, мы можем определить, насколько близко к оптимальному варианту расписания получается генерируемое расписание. На рисунке 4.17 приведены графики значений целевой функции для начальной и финальной популяции.

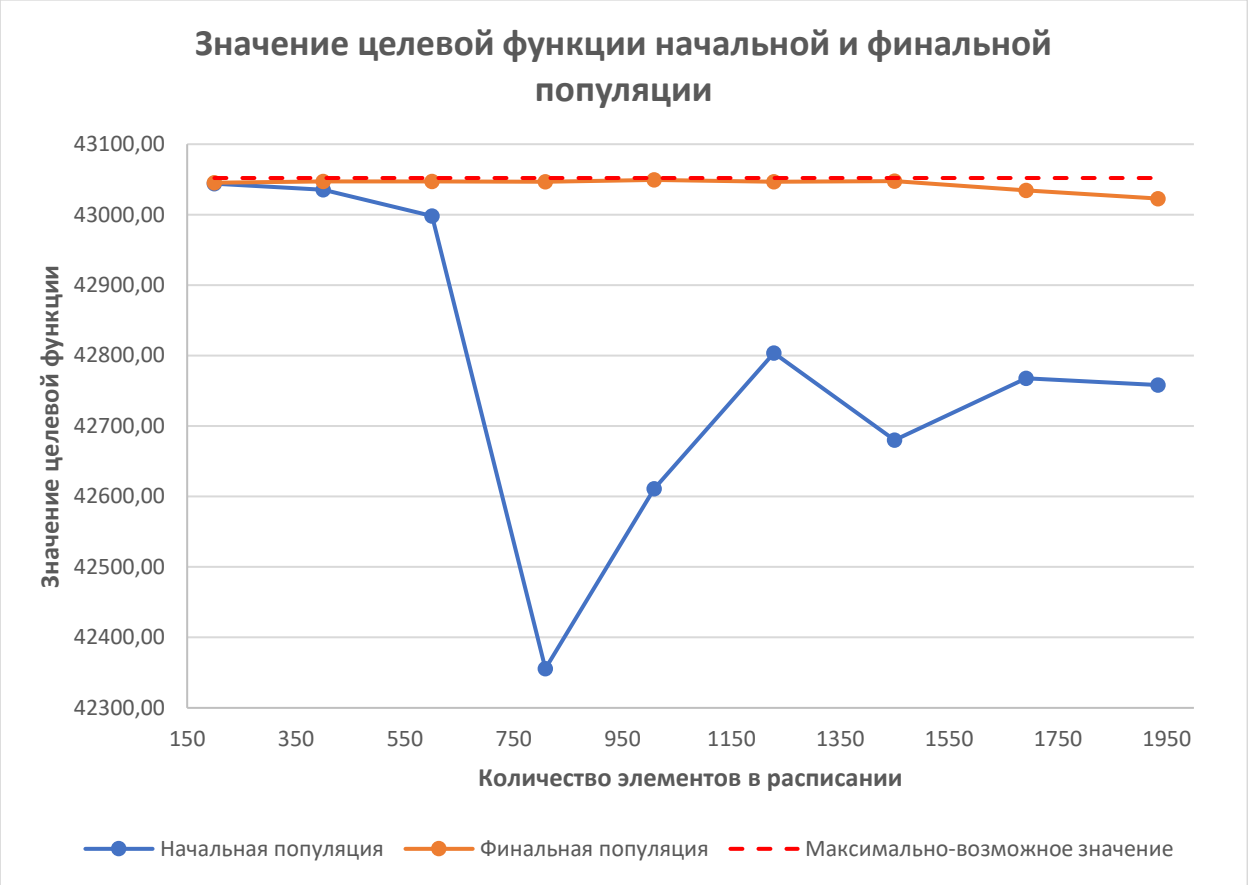


Рисунок 4.17 – Значение целевой функции начальной и финальной популяции

Как видно из приведенного графика, с увеличением числа элементов в расписании идет уменьшение значение целевой функции в начальной популяции. Однако, для финальной популяции значение целевой функции имеет незначительное отклонение от максимально-возможного значения.

Кроме того, исходя из значений целевой функции в начальной популяции для разного числа элементов в расписании видно, что её значение в процентном выражении от максимально-возможного значения имеет близкое к 100%. Это говорит о том, что предложенный алгоритм генерации

начальной популяции дает решение очень близкое к оптимальному решению, что в свою очередь позволяет получить более качественное итоговое решение по завершению работы циклического поиска в предложенном алгоритме.

4.6 Заключение по четвертому разделу

Апробация предложенной реализации алгоритма составления расписания занятий проводилась на базе образовательного портала ВКТУ им.Д.Серикбаева. В рамках данной части исследования нами были выполнены следующие действия:

а) Был произведен обзор архитектуры образовательного портала:

- было описано, из каких компонентов состоит указанный портал,
- были описаны основные программные модули, которые составляют архитектуру портала,
- были описаны механизмы взаимодействия между различными модулями портала.

б) Был произведен анализ данных образовательного портала с целью выявления данных, требуемых для составления расписания занятий. По итогам анализа нами был выявлен перечень таблиц имеющихся в базе данных образовательного портала, в которых содержится требуемые для составления расписания данные: учебные планы, группы, дисциплины, студенты и др. По итогам исследования были разработаны запросы к базе данных образовательного портала для извлечения данных для составления расписания. Текст указанных запросов приведен в приложении Е.

в) Для апробации разработанного алгоритма составления расписания на основе данных, полученных из образовательного портала, было реализовано приложение для генерации расписания для 3-го триместра 2022-2023 учебного года. Реализованное приложение на основе разработанной в 3-м разделе данного исследования библиотеки реализует генерацию расписания по определенным параметрам и ограничениям. Исходный код реализованного приложения приведен в приложении Ж.

г) Полученное в результате работы приложения расписание можно предварительно просмотреть в разработанном html-файле для оценки его пригодности для импорта в базу данных образовательного портала. Исходный код html-файла для просмотра созданного расписания приведен в приложении З, а исходный код хранимой процедуры для импорта данных в образовательный портал приведен в приложении И.

д) Для оценки временных затрат и эффективности работы предложенного генетического алгоритма в рамках данного исследования был произведен вычислительный эксперимент, который показал следующие результаты:

- время работы алгоритма имеет полиномиальную зависимость от количества элементов в расписании,

- предложенный алгоритм генерации начальной популяции позволяет получить близкое к оптимальному варианту расписание, что повышает качество итогового решения,

- при увеличении количества элементов расписания значение целевой функции для начальной популяции снижается, однако, последующая работа циклической части предложенного варианта генетического алгоритма приводит к росту значения целевой функции в финальной популяции к очень близким значениям, что свидетельствует о качестве работы алгоритма.

По результатам рассмотренных в данном разделе тем были опубликованы тезисы в сборниках 2-х конференций [63, 65, 102], 2 статьи в научных журналах РК [62, 64], а также получено свидетельство на разработанный программный продукт [67] (Приложение Д).

ЗАКЛЮЧЕНИЕ

В рамках данного научного исследования были рассмотрены вопросы и проблемы, связанные с решением задач составления расписаний занятий для высших учебных заведений. На основе проведенного исследования можно сделать следующие выводы:

- задача составления расписания относится к NP-полным задачам, что требует нахождения оптимального расписания путем полного перебора;

- для составления расписания занятий могут применяться различные методы и алгоритмы – точные и эвристические.

Применение точных методов является достаточно затруднительной задачей, поэтому все более часто для решения задачи составления расписания используются эвристические методы и алгоритмы. Одним из таких эвристических методов является генетический алгоритм. Данный вид алгоритма был взят за основу для данного научного исследования.

Для решения задачи составления расписания на основе генетического алгоритма были определены:

а) Состав и структура исходных данных, требуемых для составления расписания: учебные группы и подгруппы, преподаватели, аудитории, дисциплины, виды занятий, дни, временные интервалы и учебные потоки.

б) Выходные данные, получаемые при решении задачи: список картежей со следующим набором значений – учебный поток, аудитория, день и временные интервалы.

в) Ограничения, накладываемые на итоговое расписание, которые делятся на 2 группы «жесткие» (обязательные) и «мягкие» (рекомендации). Перечень и отнесение ограничения к указанным группам должно определяться на основе требований к расписанию. Список основных ограничений, которые могут накладываться на расписание: отсутствие «накладок» для учебной группы, преподавателя, аудитории, отсутствие «окон» в расписании учебной группы, преподавателя и др.

г) Для генетического алгоритма была определена целевая функция, строящаяся на применении оценки выполнения ограничений от 0 до 1, которые накладываются на итоговое расписание, и весовых коэффициентах для каждого ограничения. Величина весовых коэффициентов определяется, исходя из налагаемых требований к итоговому расписанию.

д) Для генетического алгоритма была предложена структура особи с одной хромосомой, генами которой являются отдельные занятия в расписании. Каждый ген имеет неизменяемую часть, которая описывает занятие в расписании и изменяемую часть, которая описывает, как данное занятие включается в расписание (день, время, аудитория)

е) Генерация начальной популяции в предложенном варианте генетического алгоритма строится на разбиении входных академических потоков на кластеры на основе семантической близости с использованием

алгоритма кластеризации FOREL. Полученные кластеры затем поочередно включаются в расписание на основе значений целевой функции.

ж) В циклической части генетического алгоритма используются 4 варианта оператора мутации и оператор скрещивания. Полученный набор особей используется для формирования нового поколения на основе отбора по значениям целевой функции и случайной выборки. Также на основе значения целевой функции производится оценка прекращения поиска решения и определения итогового решения задачи.

з) Для оптимизации вычислений выполнения ограничений, накладываемых на расписание в данном исследовании, была предложена схема построения функций для реализации проверки ограничений, которая предлагает ряд оптимизаций по сокращению возможных вычислений. Кроме того, для ряда ограничений предложены такие способы реализации, как ограничение на диапазон значений (перечень аудиторий, дней и др.) и использование дополнительных функций проверки на допустимость значений.

На основе предложенной модели и алгоритма составления расписания занятий были спроектированы и реализованы:

а) база данных для хранения исходных данных для составления расписания,

б) динамическая библиотека на базе программного каркаса .Net Framework, которая реализует работу предложенного генетического алгоритма: генерация начальной популяции, операторы мутации и скрещивания, формирование новой популяции и реализацию функций проверки ограничений, накладываемых на расписание.

Предложенная реализация составления расписания на основе генетического алгоритма была апробирована на базе образовательного портала ВКТУ им. Д.Серикбаева.

Также в рамках исследования была произведена оценка работы предложенной реализации генетического алгоритма, которая показала следующие результаты:

- время работы алгоритма имеет полиномиальную зависимость от количества элементов в расписании,

- предложенный алгоритм генерации начальной популяции позволяет получить близкое к оптимальному варианту расписание, что повышает качество итогового решения,

- при увеличении количества элементов расписания значение целевой функции для начальной популяции снижается, однако, последующая работа циклической части предложенного варианта генетического алгоритма приводит к росту значения целевой функции в финальной популяции к очень близким значениям, что свидетельствует о качестве работы алгоритма.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Р.Сети, Дж.Л.Бруно, Дж.Ульман, Р.Грэхем, В.Коглер, К.Штиглиц. Теория расписаний и вычислительные машины. Москва: Наука, 1984. 335 с.
2. Аль-Габри В.М.Н. Обзор литературных источников по теме «автоматизация составления расписания занятий и экзаменов в высших учебных заведениях» // Вестник Донского государственного технического университета, Т. 17, № 1 (88), 2017. С. 132-143. DOI: 10.23947/1992-5980-2017-17-1-132-143.
3. Аничкин А.С., Семенов В.А. Современные модели и методы теории расписаний // Труды Института системного программирования РАН, Т. 26, № 3, 2014. С. 5-50. DOI: 10.15514/ISPRAS-2014-26(3)-1.
4. Снитюк В.Є., Сіпко О.М. Технологія еволюційного формування розкладів у закладах вищої освіти. Київ: ФОП Піча Ю.В., 2022. 136 рр.
5. Лазарев А.А., Гафаров Е.Р. Теория расписаний. Задачи и алгоритмы. Москва. 2011. 222 с.
6. Сидорин А.Б, Ликучева Л.В, Дворянкин А.М. Методы автоматизации составления расписания занятий Часть 1. Классические методы // Известия Волгоградского государственного технического университета, Т. 7, № 12 (60), 2009. С. 116-120.
7. Сидорин А.Б., Ликучева Л.В., Дворянкин А.М. Методы автоматизации составления расписания занятий Часть 2. Эвристические методы оптимизации // Известия Волгоградского государственного технического университета, Т. 7, № 12 (60), 2009. С. 120-123.
8. Танаев В.С., Шкурба В.В. Введение в теорию расписаний. Москва. 1975. 256 с.
9. Конвей Р. В., Максвелл В. Л., Миллер Л. В. Теория расписаний. М.: Наука, 1975. 360 с.
10. Семенов С.П., Татаринцев Я.Б. Сравнительный анализ подходов к автоматизации составления расписаний учебных занятий в образовательных учреждениях // Известия Алтайского государственного университета. 2010. №1-1. с. 103-105.
11. Воробович Н.П., Лопатеева О.Н. О NP-полноте задач формирования расписания в вузе // Вестник Красноярского государственного аграрного университета. 2006. №11. с. 385-391.
12. Добрынин В.Н., Миловидова А.А. Технология оценки сложности для выбора метода решения задачи составления расписания // Системный анализ в науке и образовании, № 4 (14), 2011. С. 12-38.
13. Волков А.И., Ермакова А.Ю. Перспективы развития научных исследований в 21 веке сборник материалов X Международной научно-практической конференции // Методика автоматизации составления расписания экзаменационной сессии в вузе. 2016. С. 25-28.
14. Кабальнов Ю.С., Шехтман Л.И., Низамова Г.Ф., Земченкова Н.А.

Композиционный генетический алгоритм составления расписания учебных занятий // Университетский научный журнал «Вестник УГАТУ». №2. 2015. с. 99-107.

15. Муртазина Г.Ю., Нургалиева А.А. Распараллеливание композиционного генетического алгоритма для составления расписания в вузе // WORLD SCIENCE: PROBLEMS AND INNOVATIONS: сборник статей XXXVIII Международной научно-практической конференции: в 2 ч., Пенза, 25 декабря 2019 года. Том Часть 1. Пенза: "Наука и Просвещение" (ИП Гуляев Г.Ю.), 2019. С. 111-116.

16. Клеванский Н.Н. Алгоритмы формирования расписания занятий высших учебных заведений // Фундаментальные исследования, № 10-3, 2017. С. 454-458.

17. Каширина И.Л., Ухин А.Л. Моделирование и алгоритмизация задачи составления расписания учебных занятий // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии, № 2, 2015. С. 66-71.

18. Бурнасов П.В. Математическая постановка задачи составления расписания занятий // Вестник Иркутского государственного технического университета, No. 4 (87), 2014. pp. 12-18.

19. Клеванский Н.Н. Метаэвристики формирования расписаний // Мехатроника, автоматика и робототехника. 2017. №1. с. 85-88.

20. Thompson J., Dowsland K. Variants of simulated annealing for the examination timetabling problem // Annals of Operational Research, No. 63, 1996. pp. 105-128. DOI: 10.1007/BF02601641.

21. Cowling P., Kendall G., Soubeiga E. Hyperheuristics: A Robust Optimisation Method Applied to Nurse Scheduling // Proceedings of the VII Parallel Problem Solving From Nature (PPSN VlI), Lecture Notes in Computer Science, 2002. – Vol. 2439, Springer. – pp. 7-11. DOI: 10.1007/3-540-45712-7_82.

22. Horn J. Niche Distributions on the Pareto Optimal Front // Proceedings of the 2nd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003), Faro Portugal, Lecture Notes in Computer Science, 2003 – Vol. 2632, Springer. – pp. 365-375. DOI: 10.1007/3-540-36970-8_26.

23. Socha K., Knowles J., Samples M. A Max-Min Ant System for the University Course Timetabling Problem // Ant Algorithms: Proceedings of the Third International Workshop (ANTS 2002), Lecture Notes in Computer Science, 2002. – Vol. 2463, Springer. – pp. 1-13. DOI: 10.1007/3-540-45724-0_1

24. Socha K., Kisiel-Dorohinicki M. Agent-based Evolutionary Multiobjective Optimization // Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), 2002. – Hawaii USA, IEEE Press. – pp. 109-114. DOI: 10.1109/CEC.2002.1006218

25. Jin H., Wong M. L. Adaptive Diversity Maintenance and Convergence Guarantee in Multiobjective Evolutionary Algorithms // Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003), 2003 – Camberra

Australia, IEEE Press. – pp. 2498-2505. DOI: 10.1109/CEC.2003.1299402

26. Kumar R., Rockett P. Improved Sampling of the Pareto-front in Multiobjective Genetic Optimization by Steady-state Evolution: A Pareto Converging Genetic Algorithm // Evolutionary Computation, 2002 – Vol. 10. – № 3. – Pp. 283-314. DOI: 10.1162/106365602760234117.

27. Petrovic, S., E.K. Burke. University timetabling // Ch. 45. J.Y-T. Leung, ed. Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press, USA. 2004. pp. 45.1-45.24

28. Burke E., Petrovic S. Recent Research Directions in Automated Timetabling // European Journal of Operational Research, 2002, 140(2). pp. 266-280. DOI: 10.1016/s0377-2217(02)00069-3.

29. Laumams M., Thiele L., Deb K., Zitzler E. Combining Convergence and Diversity in Evolutionary Multiobjective Optimization // Evolutionary Computation, 2002 – Vol. 10. – № 3. – pp. 263-282. DOI: 10.1162/106365602760234108

30. Могилев А.А. Обзор методов решения задач теории расписаний // Информатика, вычислительная техника и инженерное образование. 2019. № 4(37). с. 19-32.

31. Беллман Р. Динамическое программирование: Пер. с англ. Москва: Издательство иностранной литературы, 1960. 400 с.

32. Кофман А., Анри-Лабордер А. Методы и модели исследования операций. Целочисленное программирование, Учебник. Москва: Мир, 1977. 432 с.

33. Anisha Jain, Ganapathy S C Aiyer, Harshita Goel, Rishabh Bhandari. A Literature Review on Timetable generation algorithms based on Genetic Algorithm and Heuristic approach // International Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, No. 4, April 2015. pp. 159-163.

34. Казбекова Г., Нуртаза А. Сабақ кестесін құрудың қолданыстағы әдістеріне шолу және талдау // Қ.А. Ясауи атындағы Халықаралық қазақ-түрік университетінің хабарлары (математика, физика, информатика сериясы). №1(24). 2023. С. 106-115. DOI: 10.47526/2023-1/2524-0080.10

35. E.K. Burke, D.G.Elliman, R.F.Weare. A University Timetabling System Based on Graph Coloring and Constraint Manipulation // Journal of Research on Computing in Education, No. 27, 1994. pp. 1-18. DOI: 10.1080/08886504.1994.10782112

36. Б.А. Логоша, А.В. Петропавловская. Комплекс моделей и методов оптимизации расписания занятий в вузе // Экономика и математические методы., Т. 29, № 4, 1993, С. 668-675.

37. Корбут А.А, Сигал И.Х., Финкельштейн Ю.Ю. Методы ветвей // Operations Forsch. Statist., Ser. Optimiz., Т. 8, № 2, 1977. С. 253-280.

38. Вагнер Г. Основы исследования операций. Т. 2. Москва: Мир, 1973. 488 с.

39. Емеличев В.А., Мельников О.И., Сарванов В.И., Тышкевич Р.И.

Лекции по теории графов. Москва: Наука, 1990. 384 с.

40. Завьялов А.М., Новиков А.В. Автоматизация задачи составления учебного расписания [электронный ресурс] // Системный анализ в науке и образовании: электрон. журн. 2009. № 1. http://www.sanse.ru/archive/12 (дата обращения: 26.05.2023).

41. Жданова Е.Г. Теория расписаний: учебник. Москва: МГУ, 1999. 222 с.

42. Kovalenko Y.V. On complexity of optimal recombination for flowshop scheduling problems // Journal of Applied and Industrial Mathematics. – 2016. – Vol. 10, No. 2. – P. 220-231. – DOI 10.1134/S1990478916020071.

43. И.А. Устинов, И.Н. Набродова Анализ существующих алгоритмов для составления расписания занятий // Известия Тульского государственного университета. Технические науки. 2022. №9. с. 109-112. DOI 10.24412/2071-6168-2022-9-109-113.

44. Безгинов А.Н., Трегубов С.Ю. Обзор существующих методов составления расписаний // Информационные технологии и программирование: Межвуз. сб. статей. М., 2005. Вып. 2. № 14. 60 с.

45. Lach G., Lübbecke M.E.: Curriculum based course timetabling: new solutions to Udine benchmark instances. Annals OR, 2012. 194(1). pp. 255-272. DOI: 10.1007/s10479-010-0700-7.

46. Song T., Liu S., Tang X., Peng X., Chen M. An iterated local search algorithm for the University Course Timetabling Problem // Applied Soft Computing, v. 68 (2018), pp. 597–608. DOI: 0.1016/j.asoc.2018.04.034.

47. Aziz N. L. A., Aizam N. A. H. A Brief Review on the features of university course timetabling problem. In AIP Conference Proceedings, volume 2016, page 020001. AIP Publishing LLC, 2018. DOI: 10.1063/1.5055403.

48. Матвеев А.И. Алгоритм оптимизации планирования ресурсов (на примере метода отжига) // Перспективные информационные технологии (ПИТ 2018): труды международной научно-практической конференции / под ред. С. А. Прохорова. 2018. С. 1046–1059.

49. Козлов М.В. Распределение значений переменных в методе имитации отжига // Моделирование, декомпозиция и оптимизация сложных динамических процессов. 2016. Т. 31, № 1(31). с. 165-173.

50. Алферьева, Н.А. Построение и анализ алгоритма муравьиной колонии для двух-стадийной задачи размещения // Молодёжь третьего тысячелетия: Сборник научных статей, Омск, 10–28 апреля 2017 года. – Омск: Омский государственный университет им. Ф.М. Достоевского, 2017. С. 1061-1065.

51. Яковлев В.В., Васильев Ю.Г., Калмыков Б.М. Решение задачи составления расписания занятий в вузе с помощью нейроподобных сетей // Вестник Чувашского университета, № 2, 2008. С. 204-210.

52. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И.Д. Рудинского. Москва: Горячая линия – Телеком, 2006. 452 с.

53. Астахова И.Ф., Фирас А.М. Составление расписания учебных занятий на основе генетического алгоритма // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии, № 2, 2013. С. 93-99.

54. Клеванский Н.Н. Формирование расписания занятий высших учебных заведений // Образовательные ресурсы и технологии. 2015. №1 (9). с. 34-44

55. Bykovyy, P., Kochan, V., Sachenko, A., Markowsky, G. Genetic algorithm implementation for perimeter security systems CAD 2007 2007 4th IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS, pp. 634-638. DOI: 10.1109/IDAACS.2007.4488498

56. Mazumder, M. M. R., & Phillips, C. (2020). partitioning known environments for multi-robot task allocation using genetic algorithms. International Journal of Computing, 19(3), 480-490. DOI: 10.47839/ijc.19.3.1897

57. Izah R., Suliadi S., Maselan A., Siti N.A.M. Razali: A Heuristics Approach for Classroom Scheduling Using Genetic Algorithm Technique, Journal of Physics: Conference Series, Volume 995, Issue 1 (2018), DOI: 10.1088/1742-6596/995/1/012050

58. Teoh C.-K., Haron H., Wibowo A., Ngadiman, M.S.: Integrating a repairing-based genetic algorithm- neighborhood search structure in solving the course timetabling problem. Journal of Computer Science, Volume 12, Issue 10, pp. 510-516 (2016), DOI: 10.3844/jcssp.2016.510.516

59. Nugroho M.A., Hermawan G.: Solving University Course Timetabling Problem Using Memetic Algorithms and Rule-based Approaches, IOP Conference Series: Materials Science and Engineering, Volume 407, Issue 1 (2018), DOI: 10.1088/1757- 899X/407/1/012012

60. Aziz N.L.A., Aizam, N.A.H.: A brief review on the features of university course timetabling problem, AIP Conference Proceedings, Volume 2016 (2018), DOI: 10.1063/1.5055403

61. Izah R., Suliadi S., Maselan A., Siti N.A.M. Razali: A Heuristics Approach for Classroom Scheduling Using Genetic Algorithm Technique, Journal of Physics: Conference Series, Volume 995, Issue 1 (2018), DOI: 10.1088/1742-6596/995/1/012050

62. Fedkin E.M., Kumargazhanova S.K., Zhomartkyzy G., Konurbayeva Zh.T. Automating class scheduling for the academic portal of the university // Совместный выпуск «Вестник Восточно-Казахстанского государственного технического университета им. Д.Серикбаева» и «Вычислительные технолгии», Сентябрь, 2018 г., №3, Т. 1 Ч.2. С. 126-131.

63. Kumargazhanova S., Suleimenova L., Fedkin Y., Urkumbaeva A. SIBIRCON 2019 - International Multi-Conference on Engineering, Computer and Information Sciences, Proceedings // Development and implementation of an automation algorithm for class scheduling process at universities. Novosibirsk. 2019. pp. 20-25. DOI: 10.1109/SIBIRCON48586.2019.8958249

64. Федькин Е., Кумаргажанова С., Увалиева И. Автоматизации процесса составления расписания занятий на основе генетического алгоритма // Вестник академии гражданской авиации, № 3 (14), 2019. С. 91-100.

65. Fedkin E., Denissova N., Krak I., Dyomina I. Proceedings of the 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021 // Automation of Scheduling Training Sessions in Educational Institutions using Genetic Algorithms. Cracow. 2021. Vol. 1. pp. 278-283. DOI: 10.1109/IDAACS53288.2021.9660939

66. Смолий В.Н., Карпенко А.В. Постановка задач разработки автоматизированной системы обучения для высшего учебного заведения // Вестник Херсонского национального технического университета. 2013. №1 (46). с. 397-401.

67. Кумаргажанова С.К., Федькин Е.М., Денисова Н.Ф. Программа для составления расписания семестровых занятий для ВУЗов, Свидетельство 20431, Сентябрь 23, 2021.

68. Козак Л.Я., Сташкова О.В., Гарбузняк Е.С., Шестопал О.В. Математическое и программное обеспечение поддержки принятия решений в процессе формирования расписания аудиторных занятий // Автоматизация и моделирование в проектировании и управлении. 2018. №2 (2). с. 9-20. DOI: 10.30987/article_5c387d60b9b3f6.51323603

69. Толстых Е.С., Толстых А.А. Автоматизация составления расписания в системе управления учебным процессом // Территория науки. 2014. №1. с. 41-48.

70. Шишканова Т.А. Разработка методики построения автоматизированной системы управления учебным процессом // Горный информационно-аналитический бюллетень (научно-технический журнал). 2013. №12. с. 366-372.

71. Шишканова Т.А. Разработка методики построения автоматизированной системы управления учебным процессом // Горный информационно-аналитический бюллетень (научно-технический журнал). 2011. №S6. с. 518-529

72. Азизова Д.Г. Автоматизация составления расписания в системе управления учебным процессом // Евразийский научный журнал. 2018. №6. с. 171-172.

73. Козак Л.Я., Сташкова О.В., Гарбузняк Е.С., Шестопал О.В. Математическое и программное обеспечение поддержки принятия решений в процессе формирования расписания аудиторных занятий // Автоматизация и моделирование в проектировании и управлении. 2018. № 2(2). с. 9-20. DOI 10.30987/article_5c387d60b9b3f6.51323603.

74. Ризун Н.О. Применение методов декомпозиции при решении многокритериальной задачи автоматизации составления расписания учебных занятий в ВУЗе // Восточно-Европейский журнал передовых технологий. 2010. №4 (44). с. 59-70. DOI: 10.15587/1729-4061.2010.2657

75. Сычев Е.В. Организационно-технический подход к составлению расписания учебных занятий в ВУЗе // Современные наукоемкие технологии. 2009. № 11. с. 87-90.

76. Жилин, Д.Г. Автоматизация расписания учебных занятий образовательной организации // Инновации. Наука. Образование. 2020. №23. с. 71-76.

77. Хасухаджиев А.С.-А., Сибикина И.В. Обобщенный алгоритм составления расписания в вузе с учетом новых требований федеральных государственных образовательных стандартов // Вестник АГТУ. Серия: Управление, вычислительная техника и информатика. 2016. С.78-68.

78. Масляев Д. А. Современное состояние задачи автоматизации составления оптимального учебного расписания в вузе // Вестник Сыктывкарского университета. Серия 1. Математика. Механика. Информатика. 2022. №1 (42). с. 23-40. DOI: 10.34130/1992-2752_2022_1_23

79. Деканова М.В. Конкретизация постановочных принципов проблемы многокритериальной оптимизации расписания занятий в университете // Вестник Полоцкого государственного университета. Серия С. Фундаментальные науки. 2014. №4. с. 56-66.

80. Ячменев Евгений Федорович Математическое моделирование задачи составления расписания занятий вуза // Экономика строительства и природопользования. 2014. №3. с. 46-51.

81. Затонский А.В., Варламова С.А. Информационное обеспечение поддержки принятия решений на примере составления расписания занятий образовательной организации // Вестник Южно-Уральского государственного университета. Серия: Компьютерные технологии, управление, радиоэлектроника. 2018. №3. с. 88-106. DOI: 10.14529/ctcr180310

82. Sakaliuk O.Y., Trishyn F.A. Mathematical Model and Efficiency of Courses Timetable Creation Process // Mathematical Modelling of Engineering Problems, Vol. 8, No. 3, June, 2021, pp. 377-385. DOI: 10.18280/mmep.080306

83. Хабипов, Р.Р. И. Автоматизация построения расписания занятий в вузе: математическая модель и методы реализации // Электронные библиотеки. 2018. Т. 21, № 5. с. 461-470.

84. Васянович, М. К. Методы автоматизации составления расписания занятий // International Journal of Advanced Studies. – 2018. – Т. 8, № 1-2. – С. 48-54.

85. Жукова М.Ю., Аль-Габри В.М. Автоматизация построения расписания экзаменов вуза с использованием генетического алгоритма [электронный ресурс] // Инженерный вестник Дона. 2017. №3 (46). с. 61.

86. Павлова А.И., Сергунин Д.А., Шевцов Е.И. Применение генетических алгоритмов для составления расписания занятий // Информационные технологии в прикладных исследованиях: сборник научных трудов. Том Выпуск 4. – Новосибирск: Новосибирский государственный университет экономики и управления "НИНХ", 2015. с. 107-114.

87. Хасухаджиев А.С.-А. Формирование системы показателей для автоматизации учебного расписания типового вуза // Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика. 2017. №3. с. 117-127. DOI: 10.24143/2072-9502-2017-3-117-127

88. Попов Г.А. Формализация задачи составления учебного расписания в высшем учебном заведении // Нефтегазовые технологии и экологическая безопасность. 2006. №1. с. 120-140.

89. Хасухаджиев А.С-А. Формализация нормативных и общесистемных требований к учебному расписанию типового вуза // Инженерный вестник Дона. 2019. №9 (60). с. 36.

90. Клеванский Н.Н. Ресурсо-ориентированный подход к формированию расписаний // Экономико-математические методы анализа деятельности предприятий АПК : Материалы II Международной научно-практической конференции, Саратов, 19–20 апреля 2018 года / Под редакцией С.И. Ткачева. – Саратов: Общество с ограниченной ответственностью "Амирит", 2018. – С. 202-207.

91. Снитюк В. Е., Сипко Е. Н. Об особенностях формирования целевой функции и ограничений в задаче составления расписания занятий // Математические машины и системы. 2014. №3. с. 88-95.

92. Шевелев С. Б. Процедура составления расписания как целочисленной задачи без потоков // Программные продукты и системы. 2009. №3. с. 21-23.

93. Лазарев, А.А. Метрики в задачах теории расписаний // Доклады Академии наук. 2010. Т. 432, № 6. с. 746-749.

94. Яндыбаева, Н. В. Генетический алгоритм в задаче оптимизации учебного расписания вуза // Современные наукоемкие технологии. 2009. №11. с. 97-98.

95. Разработка модели on-line образования для высших учебных заведений РК: Монография. – С.К. Кумаргажанова, С.С. Смаилова, Н.Ф. Денисова, С.Ж. Рахметуллина, Е.М. Федькин, В.Н. Зуев, Л.Ж. Какишева – Усть-Каменогорск: ВКТУ, 2022. – 146 с. ISBN 978-601-208-800-7

96. Mokerov, V. (2014). An approach to service-oriented information systems architecture development based on semantic closure measure. 2014 Federated Conference on Computer Science and Information Systems, pp. 269-272. DOI: 10.15439/2014F25

97. Mokerov, V., Wójcik, W., & Balova, T. (2014). Ontology based method for process oriented systems design. Przegląd Elektrotechniczny, pp. 209-213. DOI: 10.12915/pe.2014.10.51

98. Самсонова Н.В., Симонов А.Б. Составление расписания в высшем учебном заведении: математические методы и программные продукты // E-Management. 2018. №1. с. 60-69.

99. Яковлева М.С., Вайтекунене Е.Л. Автоматизация процесса составления расписания учебных занятий // Решетневские чтения. 2016. №20.

с. 176-178.

100. Колосов Д.Э. Российские автоматизированные системы управления образовательным процессом. Часть 2 // Управление образованием: теория и практика. 2011. №3 (3). с. 67-80.

101. Fedkin E., Kumargazhanova S., Smailova S., Denissova N., Györök G. Considering the functioning of an e-learning system, based on a model for assessing the performance and reliability of the system // Acta Polytechnica Hungarica, Vol. 19, No. 2, 2022. pp. 93-112. DOI:10.12700/aph.19.2.2022.2.6

102. Fedkin E., Kumargazhanova S., Denissova N., Smailova S., Rakhmetullina S., Kakisheva L., Krak I. Implementation of an integration gateway for the interaction of the learning management system with external systems and services of state information systems // Eastern-European Journal of Enterprise Technologies, Vol. 3, No. 2-117, 2022. pp. 30-38. DOI: 0.15587/1729-4061.2022.258089

103. Зимин С.Н. Составление учебного расписания, используя теорию графов // Современные наукоемкие технологии. 2007. № 11. с. 74-75.

104. В.Е. Снитюк, Е.Н. Сипко Штрафные функции в задаче составления расписания занятий // Математические машины и системы. 2015. №3. с. 158-164.

105. Шмелев В.В. Точные штрафные функции в линейном и целочисленном линейном программировании / В.В. Шмелев // Автоматика и телемеханика. 1992. № 5. с. 106-115.

106. Yeniay Ö. Penalty function methods for constrained optimization with genetic algorithms / Ö. Yeniay // Mathematical and Computational Applications. 2005. Vol. 10, N 1. pp. 45-56. DOI:10.3390/mca10010045

107. Chehouri A., Younes R., Perron J., Ilinca A. A constraint-handling technique for genetic algorithms using a violation factor // Journal of Computer Sciences 2016, 12 (7). pp. 350-362. DOI:10.3844/jcssp.2016.350.362

108. Seitakhmetova Z., Kumargazhanova S., Fedkin E., Sadvakassova A., Bobrov L. Project of a personal educational platform to support personalized learning of high school students // Вестник Евразийского национального университета имени Л.Н. Гумилева. Серия Математика. Информатика. Механика, Vol. 137, No. 4, 2021. pp. 14-24. DOI: https://doi.org/10.32523/bulmathenu.2021/4.2

109. Кумаргажанова С.К., Денисова Н.Ф., Рахметуллина С.Ж., Смаилова С.С., Федькин Е.М. Программа для модульного построения LMS, Свидетельство 29591, Октябрь 20, 2022.

110. Каменев А.В., Акинчев А.И., Мекшенева А.А., Шестопалова А.Ю., Артемов А.В., Новиков С.В. Концепция представления расписания занятий в официальном Интернет-представительстве вуза // Вестник науки и образования. 2016. №11 (23). с. 45-47.

Приложение А – Скрипт базы данных Timetable

```
CREATE DATABASE [Timetable]
  ON  (NAME  =  N'Timetable',  FILENAME  =
N'C:\timetable_db\Timetable.mdf'  ,  SIZE  =
29696KB , MAXSIZE = UNLIMITED, FILEGROWTH =
1024KB)
  LOG ON (NAME = N'Timetable_log', FILENAME =
N'C:\timetable_db\Timetable_log.ldf' , SIZE =
2614848KB , MAXSIZE = 2048GB , FILEGROWTH =
10%)
GO

USE [Timetable]
GO

CREATE SCHEMA [Timetable]
GO

CREATE TABLE [Timetable].[Days](
       [TimetableDayID]  [int]  IDENTITY(1,1)
NOT NULL,
       [ManagerID] [int] NOT NULL,
       [Name] [nvarchar](100) NOT NULL,
       [Mask] [nvarchar](356) NULL,
  CONSTRAINT  [PK_Timetable_Days]  PRIMARY  KEY
([TimetableDayID])
)
GO

CREATE TABLE [Timetable].[Discipline](
       [TimetableDisciplineID]          [int]
IDENTITY(1,1) NOT NULL,
       [ManagerID] [int] NOT NULL,
       [DisciplineID] [int] NOT NULL,
       [Name] [nvarchar](4000) NOT NULL,
  CONSTRAINT        [PK_Timetable_Discipline]
PRIMARY KEY ([TimetableDisciplineID])
)
GO

CREATE TABLE [Timetable].[DisciplineRooms](
       [DisciplineRoomID]               [int]
IDENTITY(1,1) NOT NULL,
       [ManagerID] [int] NOT NULL,
       [DisciplineID] [int] NOT NULL,
       [RoomID] [int] NULL,
       [TeacherID] [int] NULL,
       [RoomTypeID] [int] NULL,
       [TypeLecture] [int] NOT NULL,
       [PrioritySelect] [int] NOT NULL,
  CONSTRAINT   [PK_Timetable_DisciplineRooms]
PRIMARY KEY ([DisciplineRoomID])
)
GO

CREATE TABLE [Timetable].[FlowAttributes](
       [FlowAttributeID]  [int]  IDENTITY(1,1)
NOT NULL,
       [FlowID] [int] NOT NULL,
       [Name] [nvarchar](1000) NOT NULL,
       [Value] [nvarchar](4000) NOT NULL,
  CONSTRAINT    [PK_Timetable_FlowAttributes]
PRIMARY KEY ([FlowAttributeID])
)
GO

CREATE TABLE [Timetable].[FlowGroups](
       [FlowGroupID] [int] IDENTITY(1,1) NOT
NULL,
       [FlowID] [int] NOT NULL,
       [GroupID] [int] NOT NULL,
       [CountStudent] [int] NOT NULL,
       [DisciplineID] [int] NOT NULL,
       [TermNumber] [int] NULL,
       [SubGroup] [int] NOT NULL,
       [SubGroupMask] [nvarchar](4000) NULL,
  CONSTRAINT        [PK_Timetable_FlowGroups]
PRIMARY KEY ([FlowGroupID])
)
GO

CREATE TABLE [Timetable].[Flows](
       [FlowID]   [int]   IDENTITY(1,1)   NOT
NULL,
       [ManagerID] [int] NOT NULL,
       [HoursCount] [int] NOT NULL,
       [CountElements] [int] NOT NULL,
       [TypeLecture] [int] NOT NULL,
  CONSTRAINT [PK_Timetable_Flows] PRIMARY KEY
([FlowID])
)
GO

CREATE TABLE [Timetable].[FlowTeachers](
       [FlowTeacherID]  [int]  IDENTITY(1,1)
NOT NULL,
       [FlowID] [int] NOT NULL,
       [TeacherID] [int] NOT NULL,
       [SubGroup] [int] NOT NULL,
  CONSTRAINT      [PK_Timetable_FlowTeachers]
PRIMARY KEY ([FlowTeacherID])
)
GO

CREATE TABLE [Timetable].[GroupAttributes](
       [GroupAttributeID]               [int]
IDENTITY(1,1) NOT NULL,
       [TimetableGroupID] [int] NOT NULL,
       [Name] [nvarchar](1000) NOT NULL,
       [Value] [nvarchar](4000) NOT NULL,
  CONSTRAINT   [PK_Timetable_GroupAttributes]
PRIMARY KEY ([GroupAttributeID])
)
GO

CREATE TABLE [Timetable].[Groups](
       [TimetableGroupID]               [int]
IDENTITY(1,1) NOT NULL,
       [ManagerID] [int] NOT NULL,
       [GroupID] [int] NOT NULL,
       [Code] [nvarchar](100) NOT NULL,
  CONSTRAINT   [PK_Timetable_Groups]   PRIMARY
KEY ([TimetableGroupID])
)
GO

CREATE TABLE [Timetable].[LectureTime](
       [TimetableLectureTimeID]         [int]
IDENTITY(1,1) NOT NULL,
       [ManagerID] [int] NOT NULL,
       [LectureTimeID] [int] NOT NULL,
       [StartTime] [time](7) NOT NULL,
       [FinishTime] [time](7) NOT NULL,
       [NumberLecture] [int] NOT NULL,
  CONSTRAINT       [PK_Timetable_LectureTime]
PRIMARY KEY ([TimetableLectureTimeID])
)
GO

CREATE TABLE [Timetable].[Manager](
       [ManagerID]  [int]  IDENTITY(1,1)  NOT
NULL,
       [Name] [nvarchar](500) NOT NULL,
  CONSTRAINT  [PK_Timetable_Manager]  PRIMARY
KEY ([ManagerID])
)
GO

CREATE TABLE [Timetable].[RoomAttributes](
       [RoomAttributeID] [int] IDENTITY(1,1)
NOT NULL,
```

```
	[TimetableRoomID] [int] NOT NULL,
	[Name] [nvarchar](1000) NOT NULL,
	[Value] [nvarchar](4000) NOT NULL,
  CONSTRAINT    [PK_Timetable_RoomAttributes]
PRIMARY KEY ([RoomAttributeID])
)
GO

CREATE TABLE [Timetable].[Rooms](
	[TimetableRoomID] [int] IDENTITY(1,1)
NOT NULL,
	[ManagerID] [int] NOT NULL,
	[RoomID] [int] NOT NULL,
	[Places] [int] NOT NULL,
	[RoomTypeID] [int] NULL,
	[Name] [nvarchar](50) NOT NULL,
  CONSTRAINT [PK_Timetable_Rooms] PRIMARY KEY
([TimetableRoomID])
)
GO

CREATE TABLE [Timetable].[RoomTypes](
	[RoomTypeID] [int] NOT NULL,
	[Name] [nvarchar](100) NOT NULL,
  CONSTRAINT [PK_Timetable_RoomTypes] PRIMARY
KEY ([RoomTypeID])
)
GO

CREATE TABLE [Timetable].[TeacherAttributes](
	[TeacherAttributeID]	[int]
IDENTITY(1,1) NOT NULL,
	[TimetableTeacherID] [int] NOT NULL,
	[Name] [nvarchar](1000) NOT NULL,
	[Value] [nvarchar](4000) NOT NULL,
  CONSTRAINT [PK_Timetable_TeacherAttributes]
PRIMARY KEY ([TeacherAttributeID])
)
GO

CREATE TABLE [Timetable].[Teachers](
	[TimetableTeacherID]	[int]
IDENTITY(1,1) NOT NULL,
	[ManagerID] [int] NOT NULL,
	[TeacherID] [int] NOT NULL,
	[Name] [nvarchar](500) NOT NULL,
  CONSTRAINT  [PK_Timetable_Teachers]  PRIMARY
KEY ([TimetableTeacherID])
)
GO

CREATE TABLE [Timetable].[TypeLecture](
	[TypeLecture] [int] NOT NULL,
	[Name] [nvarchar](100) NOT NULL,
  CONSTRAINT	[PK_Timetable_TypeLecture]
PRIMARY KEY ([TypeLecture])
)
GO

CREATE	UNIQUE	NONCLUSTERED	INDEX
[IX_Timetable_Discipline]	ON
[Timetable].[Discipline]	([ManagerID],
[DisciplineID])
GO

CREATE	NONCLUSTERED	INDEX
[IX_Timetable_FlowsGroups_1]	ON
[Timetable].[FlowGroups]	([GroupID],
[FlowID])
GO

CREATE	NONCLUSTERED	INDEX
[IX_Timetable_FlowsGroups_2]	ON
[Timetable].[FlowGroups] ([FlowID])
GO

CREATE NONCLUSTERED INDEX [IX_FlowTeachers_1]
ON  [Timetable].[FlowTeachers]  ([TeacherID],
[FlowID])
GO

CREATE NONCLUSTERED INDEX [IX_FlowTeachers_2]
ON [Timetable].[FlowTeachers] ([FlowID])
GO

CREATE	UNIQUE	NONCLUSTERED	INDEX
[IX_Timetable_Groups] ON [Timetable].[Groups]
([GroupID], [ManagerID])
GO

CREATE	UNIQUE	NONCLUSTERED	INDEX
[IX_Timetable_LectureTime]	ON
[Timetable].[LectureTime]	([ManagerID],
[LectureTimeID])
GO

CREATE	UNIQUE	NONCLUSTERED	INDEX
[IX_Timetable_Rooms]  ON  [Timetable].[Rooms]
([ManagerID], [RoomID])
GO

CREATE	UNIQUE	NONCLUSTERED	INDEX
[IX_Timetable_Teachers]	ON
[Timetable].[Teachers]	([ManagerID],
[TeacherID])
GO

ALTER TABLE [Timetable].[Days]
  ADD CONSTRAINT  [FK_Timetable_Days_Manager]
FOREIGN	KEY([ManagerID])	REFERENCES
[Timetable].[Manager] ([ManagerID])
GO

ALTER TABLE [Timetable].[Discipline]
  ADD	CONSTRAINT
[FK_Timetable_Discipline_Manager]	FOREIGN
KEY([ManagerID])	REFERENCES
[Timetable].[Manager] ([ManagerID])
GO

ALTER TABLE [Timetable].[DisciplineRooms]
  ADD	CONSTRAINT
[FK_Timetable_DisciplineRooms_Manager]
FOREIGN	KEY([ManagerID])	REFERENCES
[Timetable].[Manager] ([ManagerID])
GO

ALTER TABLE [Timetable].[DisciplineRooms]
  ADD	CONSTRAINT
[FK_Timetable_DisciplineRooms_TypeLecture]
FOREIGN	KEY([TypeLecture])	REFERENCES
[Timetable].[TypeLecture] ([TypeLecture])
GO

ALTER TABLE [Timetable].[FlowAttributes]
  ADD	CONSTRAINT
[FK_Timetable_FlowAttributes_Flows]	FOREIGN
KEY([FlowID])  REFERENCES  [Timetable].[Flows]
([FlowID]) ON DELETE CASCADE
GO

ALTER TABLE [Timetable].[FlowGroups]
  ADD	CONSTRAINT
[FK_Timetable_FlowGroups_Flows]	FOREIGN
KEY([FlowID])  REFERENCES  [Timetable].[Flows]
([FlowID]) ON DELETE CASCADE
GO

ALTER TABLE [Timetable].[Flows]
  ADD CONSTRAINT [FK_Timetable_Flows_Manager]
FOREIGN	KEY([ManagerID])	REFERENCES
[Timetable].[Manager] ([ManagerID])
GO

ALTER TABLE [Timetable].[Flows]
  ADD	CONSTRAINT
[FK_Timetable_Flows_TypeLecture]	FOREIGN
```

```
KEY([TypeLecture])                REFERENCES
[Timetable].[TypeLecture] ([TypeLecture])
GO

ALTER TABLE [Timetable].[FlowTeachers]
  ADD                             CONSTRAINT
[FK_Timetable_FlowTeachers_Flows]     FOREIGN
KEY([FlowID])  REFERENCES  [Timetable].[Flows]
([FlowID]) ON DELETE CASCADE
GO

ALTER TABLE [Timetable].[GroupAttributes]
  ADD                             CONSTRAINT
[FK_Timetable_GroupAttributes_Groups] FOREIGN
KEY([TimetableGroupID])           REFERENCES
[Timetable].[Groups] ([TimetableGroupID])
GO

ALTER TABLE [Timetable].[Groups]
  ADD                             CONSTRAINT
[FK_Timetable_Groups_Manager]         FOREIGN
KEY([ManagerID])                  REFERENCES
[Timetable].[Manager] ([ManagerID])
GO

ALTER TABLE [Timetable].[LectureTime]
  ADD                             CONSTRAINT
[FK_Timetable_LectureTime_Manager]    FOREIGN
KEY([ManagerID])                  REFERENCES
[Timetable].[Manager] ([ManagerID])
GO

ALTER TABLE [Timetable].[RoomAttributes]
  ADD                             CONSTRAINT
[FK_Timetable_RoomAttributes_Rooms]   FOREIGN
KEY([TimetableRoomID])            REFERENCES
[Timetable].[Rooms] ([TimetableRoomID])
GO

ALTER TABLE [Timetable].[Rooms]
  ADD CONSTRAINT [FK_Timetable_Rooms_Manager]
FOREIGN      KEY([ManagerID])      REFERENCES
[Timetable].[Manager] ([ManagerID])
GO

ALTER TABLE [Timetable].[Rooms]
  ADD                             CONSTRAINT
[FK_Timetable_Rooms_RoomTypes]        FOREIGN
KEY([RoomTypeID])                 REFERENCES
[Timetable].[RoomTypes] ([RoomTypeID])
GO

ALTER TABLE [Timetable].[TeacherAttributes]
  ADD                             CONSTRAINT
[FK_Timetable_TeacherAttributes_Teachers]
FOREIGN KEY([TimetableTeacherID]) REFERENCES
[Timetable].[Teachers] ([TimetableTeacherID])
GO

ALTER TABLE [Timetable].[Teachers]
  ADD                             CONSTRAINT
[FK_Timetable_Teachers_Manager]       FOREIGN
KEY([ManagerID])                  REFERENCES
[Timetable].[Manager] ([ManagerID])
GO
```

Приложение Б – Исходный код клиентского приложения для работы с БД для составления расписания занятий

Файл TimetableEditor.dpr

```
program TimetableEditor;

uses
  Forms,
  Un_TimetableEdit in 'Un_TimetableEdit.pas'
{frmTimetableEdit},
  Un_TimetableData in 'Un_TimetableData.pas'
{DM: TDataModule},
  Un_TimetableManager                    in
'Un_TimetableManager.pas'
{frmTimetableManager};

{$R *.res}

begin
  Application.Initialize;
  Application.MainFormOnTaskbar := True;
  Application.CreateForm(TfrmTimetableEdit,
frmTimetableEdit);
  Application.CreateForm(TDM, DM);
  Application.Run;
end.
```

Файл TimetableEditor.dproj

```
        <Project
xmlns="http://schemas.microsoft.com/developer
/msbuild/2003">
                <PropertyGroup>

        <ProjectGuid>{DB03F332-2D15-48C5-
B0B5-1C7E6EE6C26C}</ProjectGuid>

        <ProjectVersion>12.0</ProjectVersion>

        <MainSource>TimetableEditor.dpr</Main
Source>
                        <Config
Condition="'$(Config)'==''">Debug</Config>

        <DCC_DCCCompiler>DCC32</DCC_DCCCompil
er>
                </PropertyGroup>
                <PropertyGroup
Condition="'$(Config)'=='Base'             or
'$(Base)'!=''">
                        <Base>true</Base>
                </PropertyGroup>
                <PropertyGroup
Condition="'$(Config)'=='Release'          or
'$(Cfg_1)'!=''">
                        <Cfg_1>true</Cfg_1>

        <CfgParent>Base</CfgParent>
                        <Base>true</Base>
                </PropertyGroup>
                <PropertyGroup
Condition="'$(Config)'=='Debug'            or
'$(Cfg_2)'!=''">
                        <Cfg_2>true</Cfg_2>

        <CfgParent>Base</CfgParent>
                        <Base>true</Base>
                </PropertyGroup>
                <PropertyGroup
Condition="'$(Base)'!=''">

        <DCC_DependencyCheckOutputName>Timeta
bleEditor.exe</DCC_DependencyCheckOutputName>

        <DCC_UnitAlias>WinTypes=Windows;WinPr
ocs=Windows;DbiTypes=BDE;DbiProcs=BDE;DbiErrs
=BDE;$(DCC_UnitAlias)</DCC_UnitAlias>

        <DCC_ImageBase>00400000</DCC_ImageBas
e>

        <DCC_Platform>x86</DCC_Platform>
                </PropertyGroup>
                <PropertyGroup
Condition="'$(Cfg_1)'!=''">

        <DCC_LocalDebugSymbols>false</DCC_Loc
alDebugSymbols>

        <DCC_Define>RELEASE;$(DCC_Define)</DC
C_Define>

        <DCC_SymbolReferenceInfo>0</DCC_Symbo
lReferenceInfo>

        <DCC_DebugInformation>false</DCC_Debu
gInformation>
                </PropertyGroup>
                <PropertyGroup
Condition="'$(Cfg_2)'!=''">

        <DCC_Define>DEBUG;$(DCC_Define)</DCC_
Define>
                </PropertyGroup>
                <ItemGroup>
                        <DelphiCompile
Include="TimetableEditor.dpr">

        <MainSource>MainSource</MainSource>
                        </DelphiCompile>
                        <DCCReference
Include="Un_TimetableEdit.pas">

        <Form>frmTimetableEdit</Form>
                        </DCCReference>
                        <DCCReference
Include="Un_TimetableData.pas">
                                <Form>DM</Form>

        <DesignClass>TDataModule</DesignClass
>
                        </DCCReference>
                        <DCCReference
Include="Un_TimetableManager.pas">

        <Form>frmTimetableManager</Form>
                        </DCCReference>
                        <BuildConfiguration
Include="Base">
                                <Key>Base</Key>
                        </BuildConfiguration>
                        <BuildConfiguration
Include="Release">

        <Key>Cfg_1</Key>

        <CfgParent>Base</CfgParent>
                        </BuildConfiguration>
                        <BuildConfiguration
Include="Debug">

        <Key>Cfg_2</Key>

        <CfgParent>Base</CfgParent>
                        </BuildConfiguration>
                </ItemGroup>
                <Import
Project="$(BDS)\Bin\CodeGear.Delphi.Targets"
```

```
Condition="Exists('$(BDS)\Bin\CodeGear.Delphi
.Targets')"/>
              <ProjectExtensions>

       <Borland.Personality>Delphi.Personali
ty.12</Borland.Personality>
                     <Borland.ProjectType/>
                     <BorlandProject>

       <Delphi.Personality>
                                    <Source>

       <Source
Name="MainSource">TimetableEditor.dpr</Source
>

       </Source>

       <Parameters>

       <Parameters
Name="UseLauncher">False</Parameters>

       <Parameters
Name="LoadAllSymbols">True</Parameters>

       <Parameters
Name="LoadUnspecifiedSymbols">False</Parameter
rs>

       </Parameters>

       <VersionInfo>

       <VersionInfo
Name="IncludeVerInfo">False</VersionInfo>

       <VersionInfo
Name="AutoIncBuild">False</VersionInfo>

       <VersionInfo
Name="MajorVer">1</VersionInfo>

       <VersionInfo
Name="MinorVer">0</VersionInfo>

       <VersionInfo
Name="Release">0</VersionInfo>

       <VersionInfo
Name="Build">0</VersionInfo>

       <VersionInfo
Name="Debug">False</VersionInfo>

       <VersionInfo
Name="PreRelease">False</VersionInfo>

       <VersionInfo
Name="Special">False</VersionInfo>

       <VersionInfo
Name="Private">False</VersionInfo>

       <VersionInfo
Name="DLL">False</VersionInfo>

       <VersionInfo
Name="Locale">1049</VersionInfo>

       <VersionInfo
Name="CodePage">1251</VersionInfo>

       </VersionInfo>

       <VersionInfoKeys>

       <VersionInfoKeys Name="CompanyName"/>

       <VersionInfoKeys
Name="FileDescription"/>

       <VersionInfoKeys
Name="FileVersion">1.0.0.0</VersionInfoKeys>

       <VersionInfoKeys
Name="InternalName"/>

       <VersionInfoKeys
Name="LegalCopyright"/>

       <VersionInfoKeys
Name="LegalTrademarks"/>

       <VersionInfoKeys
Name="OriginalFilename"/>

       <VersionInfoKeys Name="ProductName"/>

       <VersionInfoKeys
Name="ProductVersion">1.0.0.0</VersionInfoKey
s>

       <VersionInfoKeys Name="Comments"/>

       </VersionInfoKeys>

       </Delphi.Personality>
                     </BorlandProject>

       <ProjectFileVersion>12</ProjectFileVe
rsion>
              </ProjectExtensions>
       </Project>
```

Файл Un_TimetableManager.dfm

```
object                    frmTimetableManager:
TfrmTimetableManager
  Left = 0
  Top = 0
  BorderStyle = bsDialog
  Caption                                    =
#1052#1077#1085#1077#1076#1078#1077#1088'
'#1088#1072#1089#1087#1080#1089#1072#1085#108
0#1081
  ClientHeight = 308
  ClientWidth = 438
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'Tahoma'
  Font.Style = []
  OldCreateOrder = False
  Position = poMainFormCenter
  PixelsPerInch = 96
  TextHeight = 13
  object dbg1: TDBGridEh
    Left = 0
    Top = 0
    Width = 438
    Height = 290
    Align = alClient
    DataSource = dsManager
    DynProps = <>
    Flat = True
    TabOrder = 0
    object                        RowDetailData:
TRowDetailPanelControlEh
    end
  end
  object dbnvgrEdit: TDBNavigator
    Left = 0
    Top = 290
    Width = 438
    Height = 18
```

```
    DataSource = dsManager
    Align = alBottom
    Flat = True
    TabOrder = 1
  end
  object dsManager: TDataSource
    DataSet = DM.dsManager
    Left = 296
    Top = 184
  end
end
```

Файл Un_TimetableManager.pas

```
unit Un_TimetableManager;

interface

uses
  Windows, Messages, SysUtils, Variants,
Classes, Graphics, Controls, Forms, Dialogs,
DBGridEhGrouping, ToolCtrlsEh,
DBGridEhToolCtrls, DynVarsEh, ExtCtrls,
DBCtrls, DB, EhLibVCL, GridsEh,
DBAxisGridsEh, DBGridEh;

type
  TfrmTimetableManager = class(TForm)
    dbg1: TDBGridEh;
    dsManager: TDataSource;
    dbnvgrEdit: TDBNavigator;
  private
    { Private declarations }
  public
    { Public declarations }
    constructor Create; reintroduce; virtual;
  end;

var
  frmTimetableManager: TfrmTimetableManager;

implementation

uses
  Un_TimetableData;

{$R *.dfm}

{ TfrmTimetableManager }

constructor TfrmTimetableManager.Create;
begin
  inherited Create(Application);
  DM.dsManager.Active := True;
  ShowModal;
end;

end.
```

Файл Un_TimetableData.dfm

```
object DM: TDM
  OldCreateOrder = False
  OnCreate = DataModuleCreate
  Height = 479
  Width = 715
  object conDB: TADOConnection
    ConnectionString =
      'Provider=SQLOLEDB.1;Integrated
Security=SSPI;Persist Security In' +
      'fo=False;Initial
Catalog=Timetable;Data     Source=(local);Use
Proce' +
      'dure        for        Prepare=1;Auto
Translate=True;Packet Size=4096;Workstat' +
      'ion  ID=TP-VI-DEV;Use  Encryption  for
Data=False;Tag with column c' +
      'ollation when possible=False;'
    LoginPrompt = False
    Provider = 'SQLOLEDB.1'
    Left = 40
    Top = 24
  end
  object dsManager: TADODataSet
    Connection = conDB
    CursorType = ctStatic
    CommandText    =    'select    *    from
Timetable.Manager order by ManagerID desc'
    Parameters = <>
    Left = 120
    Top = 24
    object fManagerManagerID: TAutoIncField
      DisplayLabel = 'ID'
      FieldName = 'ManagerID'
      ReadOnly = True
    end
    object fManagerName: TWideStringField
      DisplayLabel                          =
#1053#1072#1079#1074#1072#1085#1080#1077
      DisplayWidth = 50
      FieldName = 'Name'
      Size = 500
    end
  end
  object dsGroups: TADODataSet
    Connection = conDB
    CursorType = ctStatic
    AfterOpen = dsGroupsAfterOpen
    AfterClose = dsGroupsAfterClose
    AfterPost = dsGroupsAfterPost
    AfterDelete = dsGroupsAfterPost
    AfterScroll = dsGroupsAfterOpen
    AfterRefresh = dsGroupsAfterPost
    OnNewRecord = dsGroupsNewRecord
    CommandText =
      'select  *  from  Timetable.Groups  WHERE
ManagerID = :ID order by Co' +
      'de'
    Parameters = <
      item
        Name = 'ID'
        Attributes = [paSigned]
        DataType = ftInteger
        Precision = 10
        Size = 4
        Value = Null
      end>
    Left = 472
    Top = 32
    object           fGroupsTimetableGroupID:
TAutoIncField
      FieldName = 'TimetableGroupID'
      ReadOnly = True
      Visible = False
    end
    object fGroupsManagerID: TIntegerField
      FieldName = 'ManagerID'
      Visible = False
    end
    object fGroupsGroupID: TIntegerField
      DisplayLabel = #1048#1044
      FieldName = 'GroupID'
    end
    object fGroupsCode: TWideStringField
      DisplayLabel = #1064#1080#1092#1088
      DisplayWidth = 18
      FieldName = 'Code'
      Size = 100
    end
  end
  object dsTeachers: TADODataSet
    Connection = conDB
    CursorType = ctStatic
    AfterOpen = dsTeachersAfterOpen
    AfterClose = dsTeachersAfterClose
    AfterPost = dsTeachersAfterPost
    AfterDelete = dsTeachersAfterPost
    AfterScroll = dsTeachersAfterOpen
```

```
    AfterRefresh = dsTeachersAfterPost
    OnNewRecord = dsGroupsNewRecord
    CommandText =
      'select * from Timetable.Teachers where
ManagerID = :ID order by ' +
      'Name'
    Parameters = <
      item
        Name = 'ID'
        Attributes = [paSigned]
        DataType = ftInteger
        Precision = 10
        Size = 4
        Value = Null
      end>
    Left = 472
    Top = 88
    object       fTeachersTimetableTeacherID:
TAutoIncField
      FieldName = 'TimetableTeacherID'
      ReadOnly = True
      Visible = False
    end
    object fTeachersManagerID: TIntegerField
      FieldName = 'ManagerID'
      Visible = False
    end
    object fTeachersTeacherID: TIntegerField
      DisplayLabel = #1048#1044
      FieldName = 'TeacherID'
    end
    object fTeachersName: TWideStringField
      DisplayLabel = #1060#1048#1054
      DisplayWidth = 80
      FieldName = 'Name'
      Size = 500
    end
  end
  object dsRooms: TADODataSet
    Connection = conDB
    CursorType = ctStatic
    AfterOpen = dsRoomsAfterOpen
    AfterClose = dsRoomsAfterClose
    AfterPost = dsRoomsAfterPost
    AfterDelete = dsRoomsAfterPost
    AfterScroll = dsRoomsAfterOpen
    AfterRefresh = dsRoomsAfterPost
    OnNewRecord = dsGroupsNewRecord
    CommandText =
      'select  *  from  Timetable.Rooms  where
ManagerID = :ID order by nam' +
      'e'
    Parameters = <
      item
        Name = 'ID'
        Attributes = [paSigned]
        DataType = ftInteger
        Precision = 10
        Size = 4
        Value = Null
      end>
    Left = 472
    Top = 144
    object             fRoomsTimetableRoomID:
TAutoIncField
      FieldName = 'TimetableRoomID'
      ReadOnly = True
      Visible = False
    end
    object fRoomsManagerID: TIntegerField
      FieldName = 'ManagerID'
      Visible = False
    end
    object fRoomsRoomID: TIntegerField
      DisplayLabel = #1048#1044
      FieldName = 'RoomID'
    end
    object fRoomsName: TWideStringField
      DisplayLabel                           =
#1053#1086#1084#1077#1088
      DisplayWidth = 10
      FieldName = 'Name'
      Size = 50
    end
    object fRoomsPlaces: TIntegerField
      DisplayLabel = #1052#1077#1089#1090
      FieldName = 'Places'
    end
    object fRoomsRoomTypeID: TIntegerField
      DisplayLabel = #1058#1080#1087
      FieldName = 'RoomTypeID'
    end
  end
  object dsLectureTime: TADODataSet
    Connection = conDB
    CursorType = ctStatic
    OnNewRecord = dsGroupsNewRecord
    CommandText =
      'select  *  from  Timetable.LectureTime
where ManagerID = :ID order ' +
      'by NumberLecture'
    Parameters = <
      item
        Name = 'ID'
        Attributes = [paSigned]
        DataType = ftInteger
        Precision = 10
        Size = 4
        Value = Null
      end>
    Left = 280
    Top = 24
    object
fLectureTimeTimetableLectureTimeID:
TAutoIncField
      FieldName = 'TimetableLectureTimeID'
      ReadOnly = True
      Visible = False
    end
    object             fLectureTimeManagerID:
TIntegerField
      FieldName = 'ManagerID'
      Visible = False
    end
    object         fLectureTimeLectureTimeID:
TIntegerField
      DisplayLabel = #1048#1044
      FieldName = 'LectureTimeID'
    end
    object         fLectureTimeNumberLecture:
TIntegerField
      DisplayLabel                           =
#1053#1086#1084#1077#1088
      FieldName = 'NumberLecture'
    end
    object             fLectureTimeStartTime:
TWideStringField
      DisplayLabel                           =
#1053#1072#1095#1072#1083#1086
      FieldName = 'StartTime'
      Size = 16
    end
    object            fLectureTimeFinishTime:
TWideStringField
      DisplayLabel                           =
#1047#1072#1074#1077#1088#1096#1077#1085#1080
#1077
      FieldName = 'FinishTime'
      Size = 16
    end
  end
  object dsDiscipline: TADODataSet
    Connection = conDB
    CursorType = ctStatic
    AfterOpen = dsDisciplineAfterOpen
    AfterClose = dsDisciplineAfterClose
    AfterPost = dsDisciplineAfterPost
```

```
    AfterDelete = dsDisciplineAfterPost
    AfterScroll = dsDisciplineAfterOpen
    AfterRefresh = dsDisciplineAfterPost
    CommandText =
      'select  *  from  Timetable.Discipline
WHERE ManagerID = :ID order b' +
      'y Name'
    Parameters = <
      item
        Name = 'ID'
        Attributes = [paSigned]
        DataType = ftInteger
        Precision = 10
        Size = 4
        Value = Null
      end>
    Left = 472
    Top = 208
    object  fDisciplineTimetableDisciplineID:
TAutoIncField
      FieldName = 'TimetableDisciplineID'
      ReadOnly = True
      Visible = False
    end
    object              fDisciplineManagerID:
TIntegerField
      FieldName = 'ManagerID'
      Visible = False
    end
    object           fDisciplineDisciplineID:
TIntegerField
      DisplayLabel = #1048#1044
      FieldName = 'DisciplineID'
    end
    object fDisciplineName: TWideStringField
      DisplayLabel                          =
#1053#1072#1079#1074#1072#1085#1080#1077
      DisplayWidth = 50
      FieldName = 'Name'
      Size = 4000
    end
  end
  object dsDiscRooms: TADODataSet
    Connection = conDB
    CursorType = ctStatic
    OnNewRecord = dsDiscRoomsNewRecord
    CommandText =
      'SELECT      *'#13#10'            FROM
Timetable.DisciplineRooms'#13#10'       WHERE
ManagerID = ' +
      ':ID                         AND'#13#10'
DisciplineID = :DID'
    Parameters = <
      item
        Name = 'ID'
        Attributes = [paSigned]
        DataType = ftInteger
        Precision = 10
        Size = 4
        Value = Null
      end
      item
        Name = 'DID'
        Attributes = [paSigned]
        DataType = ftInteger
        Precision = 10
        Size = 4
        Value = Null
      end>
    Left = 552
    Top = 208
    object        fDiscRoomsDisciplineRoomID:
TAutoIncField
      FieldName = 'DisciplineRoomID'
      ReadOnly = True
      Visible = False
    end
    object fDiscRoomsManagerID: TIntegerField
      FieldName = 'ManagerID'
      Visible = False
    end
    object            fDiscRoomsDisciplineID:
TIntegerField
      FieldName = 'DisciplineID'
      Visible = False
    end
    object             fDiscRoomsTypeLecture:
TIntegerField
      DisplayLabel     =     #1058#1080#1087'
'#1079#1072#1085#1103#1090#1080#1103
      FieldName = 'TypeLecture'
    end
    object              fDiscRoomsTypeRoomID:
TIntegerField
      DisplayLabel     =     #1042#1080#1076'
'#1072#1091#1076#1080#1090#1086#1088#1080#108
0
      FieldName = 'RoomTypeID'
    end
    object fDiscRoomsRoomID: TIntegerField
      DisplayLabel                          =
#1040#1091#1076#1080#1090#1086#1088#1080#1103
      FieldName = 'RoomID'
    end
    object fDiscRoomsTeacherID: TIntegerField
      DisplayLabel                          =
#1055#1088#1077#1087#1086#1076#1072#1074#1072
#1090#1077#1083#1100
      FieldName = 'TeacherID'
    end
    object          fDiscRoomsPrioritySelect:
TIntegerField
      DisplayLabel                          =
#1055#1088#1080#1086#1088#1080#1090#1077#1090
      FieldName = 'PrioritySelect'
    end
  end
  object dsFlows: TADODataSet
    Connection = conDB
    CursorType = ctStatic
    AfterOpen = dsFlowsAfterOpen
    AfterScroll = dsFlowsAfterOpen
    OnNewRecord = dsGroupsNewRecord
    CommandText =
      'declare  @FilterGroup  nvarchar(500)  =
:FilterGroup'#13#10'declare @Filt' +
      'erDisc         nvarchar(500)         =
:FilterDisc'#13#10'declare     @FilterTeacher
nvarc' +
      'har(500)                             =
:FilterTeacher'#13#10'declare  @ManagerID  int
= :ID;'#13#10#13#10'WITH' +
      '   fi   AS   ('#13#10'SELECT   T.FlowID,
'#13#10'        STRING_AGG(CAST(Code AS nvar'
+
      'char(max)),     '#39',     '#39')     AS
Code,'#13#10'         STRING_AGG(CAST(Name  AS
nvarc' +
      'har(max)),     '#39',     '#39')     AS
Name'#13#10'      FROM   (SELECT   G.Code   AS
Code,'#13#10'      ' +
      '                    FG.FlowID,'#13#10'
CASE WHEN ROW_NUMBER() OVER ' +
      '(PARTITION BY FG.FlowID, D.Name ORDER
BY D.Name) = 1 THEN D.Name' +
      ' END AS Name'#13#10'              FROM
Timetable.FlowGroups FG INNER JOIN ' +
      'Timetable.Flows               F'#13#10'
ON FG.FlowID = F.FlowID INNE' +
      'R   JOIN   Timetable.Groups   G'#13#10'
ON FG.GroupID = G.Gr' +
      'oupID                      AND'#13#10'
G.ManagerID = F.ManagerID INNER J' +
      'OIN   Timetable.Discipline   D'#13#10'
ON FG.DisciplineID ' +
      '=      D.DisciplineID      and'#13#10'
D.ManagerID = F.Manage' +
```

```
      'rID)     T'#13#10'         GROUP     BY
T.FlowID'#13#10')'#13#10#13#10'SELECT
F.*,'#13#10'       CAST(fi.Na' +
      'me      AS      nvarchar(4000))      AS
Discipline,'#13#10'           CAST(fi.Code  AS
nva' +
      'rchar(4000))  AS  Groups'#13#10'   FROM
Timetable.Flows F LEFT OUTER JOIN' +
      ' fi  '#13#10'            ON  F.FlowID  =
fi.FlowID'#13#10'  WHERE ManagerID = @Man' +
      'agerID                      and'#13#10'
((@FilterGroup IS NULL AND @FilterDisc IS NU'
+
      'LL   AND   @FilterTeacher   IS   NULL)
OR'#13#10'           (EXISTS(SELECT *'#13#10'
' +
      '                                    FROM
Timetable.FlowGroups FG INNER JOIN Timetabl'
+
      'e.Groups                        G'#13#10'
ON FG.GroupID = G.GroupID ' +
      'AND'#13#10'
G.ManagerID = @ManagerID, stri' +
      'ng_split(@FilterGroup,    '#39'|'#39')
Fl'#13#10'                    WHERE FG.FlowI' +
      'D      =      F.FlowID      AND'#13#10'
G.Code LIKE Fl.value) ' +
      'AND'#13#10'                EXISTS(SELECT
*'#13#10'                   FROM Timetable.' +
      'FlowTeachers     FT     INNER     JOIN
Timetable.Teachers T'#13#10'                 '
+
      '                    ON  FT.TeacherID  =
T.TeacherID AND'#13#10'                      ' +
      '             T.ManagerID = @ManagerID,
string_split(@FilterTeacher' +
      ',        '#39'|'#39')        Fl'#13#10'
WHERE FT.FlowID = F.FlowID AND'#13#10'   ' +
      '                           T.Name  LIKE
Fl.value) AND'#13#10'         EXISTS(' +
      'SELECT *'#13#10'                    FROM
Timetable.FlowGroups FG INNER J' +
      'OIN   Timetable.Discipline   D'#13#10'
ON FG.Disc' +
      'iplineID = D.DisciplineID AND'#13#10'
D.Ma' +
      'nagerID          =          @ManagerID,
string_split(@FilterDisc,        '#39'|'#39')
Fl'#13#10'       ' +
      '            WHERE FG.FlowID = F.FlowID
AND'#13#10'                        ' +
      '          D.Name LIKE Fl.value)))'
    Parameters = <
      item
        Name = 'FilterGroup'
        DataType = ftWideString
        Size = 500
        Value = Null
      end
      item
        Name = 'FilterDisc'
        DataType = ftWideString
        Size = 500
        Value = Null
      end
      item
        Name = 'FilterTeacher'
        DataType = ftWideString
        Size = 500
        Value = Null
      end
      item
        Name = 'ID'
        Attributes = [paSigned]
        DataType = ftInteger
        Precision = 10
        Size = 4
        Value = Null
      end>
    Left = 40
    Top = 120
    object fFlowsFlowID: TAutoIncField
      FieldName = 'FlowID'
      ReadOnly = True
      Visible = False
    end
    object fFlowsManagerID: TIntegerField
      FieldName = 'ManagerID'
      Visible = False
    end
    object fFlowsTypeLecture: TIntegerField
      DisplayLabel     =     #1058#1080#1087'
'#1087#1086#1090#1086#1082#1072
      FieldName = 'TypeLecture'
    end
    object fFlowsHoursCount: TIntegerField
      DisplayLabel = #1063#1072#1089#1099
      FieldName = 'HoursCount'
    end
    object fFlowsCountElements: TIntegerField
      DisplayLabel     =     #1050#1086#1083'-
'#1074#1086
      FieldName = 'CountElements'
    end
    object fFlowsGroups: TWideStringField
      DisplayLabel                          =
#1043#1088#1091#1087#1087#1099
      DisplayWidth = 30
      FieldName = 'Groups'
      ReadOnly = True
      Size = 4000
    end
    object fFlowsDiscipline: TWideStringField
      DisplayLabel                          =
#1044#1080#1089#1094#1080#1087#1083#1080#1085
#1072
      DisplayWidth = 25
      FieldName = 'Discipline'
      ReadOnly = True
      Size = 4000
    end
  end
  object dsFlowGroup: TADODataSet
    Connection = conDB
    CursorType = ctStatic
    OnNewRecord = dsFlowGroupNewRecord
    CommandText     =     'select     *     from
Timetable.FlowGroups WHERE FlowID = :FID'
    Parameters = <
      item
        Name = 'FID'
        Attributes = [paSigned]
        DataType = ftInteger
        Precision = 10
        Size = 4
        Value = Null
      end>
    Left = 112
    Top = 120
    object             fFlowGroupFlowGroupID:
TAutoIncField
      FieldName = 'FlowGroupID'
      ReadOnly = True
      Visible = False
    end
    object fFlowGroupFlowID: TIntegerField
      FieldName = 'FlowID'
      Visible = False
    end
    object fFlowGroupGroupID: TIntegerField
      DisplayLabel                          =
#1043#1088#1091#1087#1087#1072
      FieldName = 'GroupID'
    end
    object            fFlowGroupDisciplineID:
TIntegerField
```

```
      DisplayLabel                          =
#1044#1080#1089#1094#1080#1087#1083#1080#1085
#1072
      FieldName = 'DisciplineID'
    end
    object            fFlowGroupCountStudent:
TIntegerField
      DisplayLabel    =    #1050#1086#1083'-
'#1074#1086
      FieldName = 'CountStudent'
    end
    object              fFlowGroupTermNumber:
TIntegerField
      DisplayLabel                          =
#1057#1077#1084#1077#1089#1090#1088
      FieldName = 'TermNumber'
    end
    object fFlowGroupSubGroup: TIntegerField
      DisplayLabel                          =
#1055#1086#1076#1075#1088#1091#1087#1087#1072
      FieldName = 'SubGroup'
    end
    object            fFlowGroupSubGroupMask:
TWideStringField
      DisplayLabel                          =
#1052#1072#1089#1082#1072'
'#1087#1086#1076#1075#1088#1091#1087#1087#109
9
      FieldName = 'SubGroupMask'
      Size = 4000
    end
  end
  object dsFlowTeacher: TADODataSet
    Connection = conDB
    CursorType = ctStatic
    OnNewRecord = dsFlowGroupNewRecord
    CommandText    =    'select    *    from
Timetable.FlowTeachers where FlowID = :FID'
    Parameters = <
      item
        Name = 'FID'
        Attributes = [paSigned]
        DataType = ftInteger
        Precision = 10
        Size = 4
        Value = Null
      end>
    Left = 192
    Top = 120
    object         fFlowTeacherFlowTeacherID:
TAutoIncField
      FieldName = 'FlowTeacherID'
      ReadOnly = True
      Visible = False
    end
    object fFlowTeacherFlowID: TIntegerField
      FieldName = 'FlowID'
      Visible = False
    end
    object             fFlowTeacherTeacherID:
TIntegerField
      DisplayLabel                          =
#1055#1088#1077#1087#1086#1076#1072#1074#1072
#1090#1077#1083#1100
      FieldName = 'TeacherID'
    end
    object              fFlowTeacherSubGroup:
TIntegerField
      DisplayLabel                          =
#1055#1086#1076#1075#1088#1091#1087#1087#1072
      FieldName = 'SubGroup'
    end
  end
  object dsGroupsAttr: TADODataSet
    Connection = conDB
    CursorType = ctStatic
    OnNewRecord = dsGroupsAttrNewRecord
    CommandText =
      'SELECT      *'#13#10'            FROM
Timetable.GroupAttributes'#13#10'       WHERE
TimetableGro' +
      'upID = :GID'
    Parameters = <
      item
        Name = 'GID'
        DataType = ftInteger
        Size = -1
        Value = Null
      end>
    Left = 552
    Top = 32
    object       fGroupsAttrGroupAttributeID:
TAutoIncField
      FieldName = 'GroupAttributeID'
      ReadOnly = True
    end
    object       fGroupsAttrTimetableGroupID:
TIntegerField
      FieldName = 'TimetableGroupID'
    end
    object fGroupsAttrName: TWideStringField
      DisplayLabel                          =
#1040#1090#1088#1080#1073#1091#1090
      DisplayWidth = 20
      FieldName = 'Name'
      Size = 1000
    end
    object fGroupsAttrValue: TWideStringField
      DisplayLabel                          =
#1047#1085#1072#1095#1077#1085#1080#1077
      DisplayWidth = 50
      FieldName = 'Value'
      Size = 4000
    end
  end
  object dsTeacherAttr: TADODataSet
    Connection = conDB
    CursorType = ctStatic
    OnNewRecord = dsTeacherAttrNewRecord
    CommandText =
      'SELECT      *'#13#10'            FROM
Timetable.TeacherAttributes'#13#10'     WHERE
TimetableT' +
      'eacherID = :TID'
    Parameters = <
      item
        Name = 'TID'
        Attributes = [paSigned]
        DataType = ftInteger
        Precision = 10
        Size = 4
        Value = Null
      end>
    Left = 552
    Top = 88
    object    fTeacherAttrTeacherAttributeID:
TAutoIncField
      FieldName = 'TeacherAttributeID'
      ReadOnly = True
    end
    object    fTeacherAttrTimetableTeacherID:
TIntegerField
      FieldName = 'TimetableTeacherID'
    end
    object fTeacherAttrName: TWideStringField
      DisplayLabel                          =
#1040#1090#1088#1080#1073#1091#1090
      DisplayWidth = 20
      FieldName = 'Name'
      Size = 1000
    end
    object                 fTeacherAttrValue:
TWideStringField
      DisplayLabel                          =
#1047#1085#1072#1095#1077#1085#1080#1077
      DisplayWidth = 50
      FieldName = 'Value'
```

```
      Size = 4000
    end
  end
  object dsRoomTypes: TADODataSet
    Connection = conDB
    CursorType = ctStatic
    CommandText  =  'SELECT  *'#13#10'    FROM
Timetable.RoomTypes'
    Parameters = <>
    Left = 192
    Top = 24
  end
  object dsRoomAttr: TADODataSet
    Connection = conDB
    CursorType = ctStatic
    OnNewRecord = dsRoomAttrNewRecord
    CommandText =
      'SELECT      *'#13#10'            FROM
Timetable.RoomAttributes'#13#10'       WHERE
TimetableRoom' +
      'ID = :RID'
    Parameters = <
      item
        Name = 'RID'
        Attributes = [paSigned]
        DataType = ftInteger
        Precision = 10
        Size = 4
        Value = Null
      end>
    Left = 552
    Top = 144
    object          fRoomAttrRoomAttributeID:
TAutoIncField
      FieldName = 'RoomAttributeID'
      ReadOnly = True
    end
    object          fRoomAttrTimetableRoomID:
TIntegerField
      FieldName = 'TimetableRoomID'
    end
    object fRoomAttrName: TWideStringField
      DisplayLabel                          =
#1040#1090#1088#1080#1073#1091#1090
      DisplayWidth = 20
      FieldName = 'Name'
      Size = 1000
    end
    object fRoomAttrValue: TWideStringField
      DisplayLabel                          =
#1047#1085#1072#1095#1077#1085#1080#1077
      DisplayWidth = 50
      FieldName = 'Value'
      Size = 4000
    end
  end
  object dsDays: TADODataSet
    Connection = conDB
    CursorType = ctStatic
    OnNewRecord = dsGroupsNewRecord
    CommandText  =  'SELECT  *'#13#10'    FROM
Timetable.Days'#13#10'    WHERE  ManagerID  =
:ID'
    Parameters = <
      item
        Name = 'ID'
        Attributes = [paSigned]
        DataType = ftInteger
        Precision = 10
        Size = 4
        Value = Null
      end>
    Left = 352
    Top = 24
    object fDaysTimetableDayID: TAutoIncField
      FieldName = 'TimetableDayID'
      ReadOnly = True
      Visible = False
    end
    object fDaysManagerID: TIntegerField
      FieldName = 'ManagerID'
      Visible = False
    end
    object fDaysName: TWideStringField
      DisplayLabel = #1044#1077#1085#1100
      DisplayWidth = 20
      FieldName = 'Name'
      Size = 100
    end
    object fDaysMask: TWideStringField
      DisplayLabel                          =
#1052#1072#1089#1082#1072
      DisplayWidth = 20
      FieldName = 'Mask'
      Size = 356
    end
  end
  object dsTypeLecture: TADODataSet
    Connection = conDB
    CursorType = ctStatic
    CommandText    =    'SELECT    *    FROM
Timetable.TypeLecture'
    Parameters = <>
    Left = 472
    Top = 280
    object           fTypeLectureTypeLecture:
TIntegerField
      FieldName = 'TypeLecture'
    end
    object fTypeLectureName: TWideStringField
      FieldName = 'Name'
      Size = 100
    end
  end
  object dsTeacherList: TADODataSet
    LockType = ltBatchOptimistic
    Parameters = <>
    Left = 40
    Top = 280
  end
  object dsGroupList: TADODataSet
    Parameters = <>
    Left = 40
    Top = 224
  end
  object dsRoomList: TADODataSet
    Parameters = <>
    Left = 120
    Top = 280
  end
  object dsDiscList: TADODataSet
    Parameters = <>
    Left = 120
    Top = 224
  end
  object dsFlowAttr: TADODataSet
    Connection = conDB
    CursorType = ctStatic
    OnNewRecord = dsFlowGroupNewRecord
    CommandText  =  'SELECT  *'#13#10'    FROM
Timetable.FlowAttributes'#13#10'       WHERE
FlowID = :FID'
    Parameters = <
      item
        Name = 'FID'
        Attributes = [paSigned]
        DataType = ftInteger
        Precision = 10
        Size = 4
        Value = Null
      end>
    Left = 272
    Top = 120
    object          fFlowAttrFlowAttributeID:
TAutoIncField
      FieldName = 'FlowAttributeID'
      ReadOnly = True
    end
```

```
    object fFlowAttrFlowID: TIntegerField
      FieldName = 'FlowID'
    end
    object fFlowAttrName: TWideStringField
      DisplayLabel                          =
#1040#1090#1088#1080#1073#1091#1090
      DisplayWidth = 20
      FieldName = 'Name'
      Size = 1000
    end
    object fFlowAttrValue: TWideStringField
      DisplayLabel                          =
#1047#1085#1072#1095#1077#1085#1080#1077
      DisplayWidth = 50
      FieldName = 'Value'
      Size = 4000
    end
  end
end
```

Файл Un_TimetableData.pas

```
unit Un_TimetableData;

interface

uses
  SysUtils, Classes, DB, ADODB, IniFiles;

type
  TDM = class(TDataModule)
    conDB: TADOConnection;
    dsManager: TADODataSet;
    fManagerManagerID: TAutoIncField;
    fManagerName: TWideStringField;
    dsGroups: TADODataSet;
    fGroupsTimetableGroupID: TAutoIncField;
    fGroupsManagerID: TIntegerField;
    fGroupsGroupID: TIntegerField;
    fGroupsCode: TWideStringField;
    dsTeachers: TADODataSet;
    fTeachersTimetableTeacherID:
TAutoIncField;
    fTeachersManagerID: TIntegerField;
    fTeachersTeacherID: TIntegerField;
    fTeachersName: TWideStringField;
    dsRooms: TADODataSet;
    dsLectureTime: TADODataSet;
    fLectureTimeTimetableLectureTimeID:
TAutoIncField;
    fLectureTimeManagerID: TIntegerField;
    fLectureTimeLectureTimeID: TIntegerField;
    fLectureTimeStartTime: TWideStringField;
    fLectureTimeFinishTime: TWideStringField;
    fLectureTimeNumberLecture: TIntegerField;
    fRoomsTimetableRoomID: TAutoIncField;
    fRoomsManagerID: TIntegerField;
    fRoomsRoomID: TIntegerField;
    fRoomsPlaces: TIntegerField;
    fRoomsRoomTypeID: TIntegerField;
    fRoomsName: TWideStringField;
    dsDiscipline: TADODataSet;
    fDisciplineTimetableDisciplineID:
TAutoIncField;
    fDisciplineManagerID: TIntegerField;
    fDisciplineDisciplineID: TIntegerField;
    fDisciplineName: TWideStringField;
    dsDiscRooms: TADODataSet;
    fDiscRoomsDisciplineRoomID:
TAutoIncField;
    fDiscRoomsManagerID: TIntegerField;
    fDiscRoomsDisciplineID: TIntegerField;
    fDiscRoomsRoomID: TIntegerField;
    fDiscRoomsTeacherID: TIntegerField;
    fDiscRoomsTypeRoomID: TIntegerField;
    fDiscRoomsTypeLecture: TIntegerField;
    fDiscRoomsPrioritySelect: TIntegerField;
    dsFlows: TADODataSet;
    fFlowsFlowID: TAutoIncField;
    fFlowsManagerID: TIntegerField;
    fFlowsHoursCount: TIntegerField;
    fFlowsCountElements: TIntegerField;
    fFlowsTypeLecture: TIntegerField;
    fFlowsGroups: TWideStringField;
    fFlowsDiscipline: TWideStringField;
    dsFlowGroup: TADODataSet;
    fFlowGroupFlowGroupID: TAutoIncField;
    fFlowGroupFlowID: TIntegerField;
    fFlowGroupGroupID: TIntegerField;
    fFlowGroupCountStudent: TIntegerField;
    fFlowGroupDisciplineID: TIntegerField;
    fFlowGroupTermNumber: TIntegerField;
    fFlowGroupSubGroup: TIntegerField;
    dsFlowTeacher: TADODataSet;
    fFlowTeacherFlowTeacherID: TAutoIncField;
    fFlowTeacherFlowID: TIntegerField;
    fFlowTeacherTeacherID: TIntegerField;
    fFlowTeacherSubGroup: TIntegerField;
    fFlowGroupSubGroupMask: TWideStringField;
    dsGroupsAttr: TADODataSet;
    fGroupsAttrGroupAttributeID:
TAutoIncField;
    fGroupsAttrTimetableGroupID:
TIntegerField;
    fGroupsAttrName: TWideStringField;
    fGroupsAttrValue: TWideStringField;
    dsTeacherAttr: TADODataSet;
    fTeacherAttrTeacherAttributeID:
TAutoIncField;
    fTeacherAttrTimetableTeacherID:
TIntegerField;
    fTeacherAttrName: TWideStringField;
    fTeacherAttrValue: TWideStringField;
    dsRoomTypes: TADODataSet;
    dsRoomAttr: TADODataSet;
    fRoomAttrRoomAttributeID: TAutoIncField;
    fRoomAttrTimetableRoomID: TIntegerField;
    fRoomAttrName: TWideStringField;
    fRoomAttrValue: TWideStringField;
    dsDays: TADODataSet;
    fDaysTimetableDayID: TAutoIncField;
    fDaysManagerID: TIntegerField;
    fDaysName: TWideStringField;
    fDaysMask: TWideStringField;
    dsTypeLecture: TADODataSet;
    fTypeLectureTypeLecture: TIntegerField;
    fTypeLectureName: TWideStringField;
    dsTeacherList: TADODataSet;
    dsGroupList: TADODataSet;
    dsRoomList: TADODataSet;
    dsDiscList: TADODataSet;
    dsFlowAttr: TADODataSet;
    fFlowAttrFlowAttributeID: TAutoIncField;
    fFlowAttrFlowID: TIntegerField;
    fFlowAttrName: TWideStringField;
    fFlowAttrValue: TWideStringField;
    procedure        DataModuleCreate(Sender:
TObject);
    procedure      dsGroupsNewRecord(DataSet:
TDataSet);
    procedure  dsDisciplineAfterOpen(DataSet:
TDataSet);
    procedure   dsDiscRoomsNewRecord(DataSet:
TDataSet);
    procedure      dsGroupsAfterOpen(DataSet:
TDataSet);
    procedure       dsFlowsAfterOpen(DataSet:
TDataSet);
    procedure   dsFlowGroupNewRecord(DataSet:
TDataSet);
    procedure
dsExcludeTeacherNewRecord(DataSet: TDataSet);
    procedure    dsTeachersAfterOpen(DataSet:
TDataSet);
    procedure       dsRoomsAfterOpen(DataSet:
TDataSet);
    procedure  dsGroupsAttrNewRecord(DataSet:
TDataSet);
```

```
    procedure dsTeacherAttrNewRecord(DataSet:
TDataSet);
    procedure    dsRoomAttrNewRecord(DataSet:
TDataSet);
    procedure    dsTeachersAfterPost(DataSet:
TDataSet);
    procedure   dsTeachersAfterClose(DataSet:
TDataSet);
    procedure      dsGroupsAfterPost(DataSet:
TDataSet);
    procedure     dsGroupsAfterClose(DataSet:
TDataSet);
    procedure       dsRoomsAfterPost(DataSet:
TDataSet);
    procedure      dsRoomsAfterClose(DataSet:
TDataSet);
    procedure  dsDisciplineAfterPost(DataSet:
TDataSet);
    procedure dsDisciplineAfterClose(DataSet:
TDataSet);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure ReOpen(ds: TADODataSet; names:
array of string; values: array of Variant);
overload;
    procedure  ReOpen(ds:  TADODataSet;  name:
string; value: Variant); overload;
  end;

var
  DM: TDM;

implementation

{$R *.dfm}

procedure        TDM.DataModuleCreate(Sender:
TObject);
var ini: TIniFile;
begin
  ini := TIniFile.Create('settings.ini');
  conDB.ConnectionString                    :=
ini.ReadString('DB', 'con', '');
  conDB.Connected := True;
  dsRoomTypes.Active := True;
  dsTypeLecture.Active := True;
  ini.Free;
end;

procedure TDM.dsDisciplineAfterClose(DataSet:
TDataSet);
begin
  dsDiscList.Close;
end;

procedure  TDM.dsDisciplineAfterOpen(DataSet:
TDataSet);
begin
  ReOpen(dsDiscRooms,     ['ID',     'DID'],
[fDisciplineManagerID.Value,
fDisciplineDisciplineID.Value]);
  if   dsDiscList.Recordset   =   nil   then
dsDiscList.Clone(dsDiscipline);
end;

procedure  TDM.dsDisciplineAfterPost(DataSet:
TDataSet);
begin
  dsDiscList.Clone(dsDiscipline);
end;

procedure   TDM.dsDiscRoomsNewRecord(DataSet:
TDataSet);
begin
  DataSet['ManagerID']                      :=
fManagerManagerID.Value;
  DataSet['DisciplineID']                   :=
fDisciplineDisciplineID.Value;
end;

procedure
TDM.dsExcludeTeacherNewRecord(DataSet:
TDataSet);
begin
  DataSet['ManagerID']                      :=
fTeachersManagerID.Value;
  DataSet['TeacherID']                      :=
fTeachersTeacherID.Value;
end;

procedure   TDM.dsFlowGroupNewRecord(DataSet:
TDataSet);
begin
  DataSet['FlowID'] := fFlowsFlowID.Value;
end;

procedure       TDM.dsFlowsAfterOpen(DataSet:
TDataSet);
begin
  ReOpen(dsFlowGroup,                   'FID',
fFlowsFlowID.Value);
  ReOpen(dsFlowTeacher,                 'FID',
fFlowsFlowID.Value);
  ReOpen(dsFlowAttr,                    'FID',
fFlowsFlowID.Value);
end;

procedure     TDM.dsGroupsAfterClose(DataSet:
TDataSet);
begin
  dsGroupList.Close;
end;

procedure      TDM.dsGroupsAfterOpen(DataSet:
TDataSet);
begin
  ReOpen(dsGroupsAttr,                ['GID'],
[fGroupsTimetableGroupID.Value]);
  if   dsGroupList.Recordset   =   nil   then
dsGroupList.Clone(dsGroups);
end;

procedure      TDM.dsGroupsAfterPost(DataSet:
TDataSet);
begin
  dsGroupList.Clone(dsGroups);
end;

procedure  TDM.dsGroupsAttrNewRecord(DataSet:
TDataSet);
begin
  fGroupsAttrTimetableGroupID.Value        :=
fGroupsTimetableGroupID.Value;
end;

procedure      TDM.dsGroupsNewRecord(DataSet:
TDataSet);
begin
  DataSet['ManagerID']                      :=
fManagerManagerID.Value;
end;

procedure    TDM.dsRoomAttrNewRecord(DataSet:
TDataSet);
begin
  fRoomAttrTimetableRoomID.Value           :=
fRoomsTimetableRoomID.Value;
end;

procedure      TDM.dsRoomsAfterClose(DataSet:
TDataSet);
begin
  dsRoomList.Close;
end;
```

```
procedure       TDM.dsRoomsAfterOpen(DataSet:
TDataSet);
begin
  DM.ReOpen(dsRoomAttr,              ['RID'],
[DM.fRoomsTimetableRoomID.Value]);
  if   dsRoomList.Recordset   =   nil   then
dsRoomList.Clone(dsRooms);
end;

procedure       TDM.dsRoomsAfterPost(DataSet:
TDataSet);
begin
  dsRoomList.Clone(dsRooms);
end;

procedure TDM.dsTeacherAttrNewRecord(DataSet:
TDataSet);
begin
  fTeacherAttrTimetableTeacherID.Value     :=
fTeachersTimetableTeacherID.Value;
end;

procedure   TDM.dsTeachersAfterClose(DataSet:
TDataSet);
begin
  dsTeacherList.Close;
end;

procedure    TDM.dsTeachersAfterOpen(DataSet:
TDataSet);
begin
  DM.ReOpen(DM.dsTeacherAttr,        ['TID'],
[DM.fTeachersTimetableTeacherID.Value]);
  if  dsTeacherList.Recordset  =  nil  then
dsTeacherList.Clone(dsTeachers);
end;

procedure    TDM.dsTeachersAfterPost(DataSet:
TDataSet);
begin
  dsTeacherList.Clone(dsTeachers);
end;

procedure  TDM.ReOpen(ds:  TADODataSet;  name:
string; value: Variant);
begin
  ReOpen(ds, [name], [value]);
end;

procedure TDM.ReOpen(ds: TADODataSet; names:
array of string; values: array of Variant);
var i: Integer;
begin
  ds.Active := False;
  for i := 0 to Length(names) - 1 do
    ds.Parameters.ParamValues[names[i]]    :=
values[i];
  ds.Active := True;
end;

end.
```

Файл Un_TimetableEdit.dfm

```
object frmTimetableEdit: TfrmTimetableEdit
  Left = 0
  Top = 0
  Caption                                  =
#1043#1077#1085#1077#1088#1072#1094#1080#1103
'
'#1088#1072#1089#1087#1080#1089#1072#1085#108
0#1103' '#1079#1072#1085#1103#1090#1080#1081
  ClientHeight = 605
  ClientWidth = 1004
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'Tahoma'
  Font.Style = []
  OldCreateOrder = False
  PixelsPerInch = 96
  TextHeight = 13
  object tlb1: TToolBar
    Left = 0
    Top = 0
    Width = 1004
    Height = 23
    AutoSize = True
    ButtonHeight = 19
    ButtonWidth = 117
    Caption = 'tlb1'
    EdgeBorders  =  [ebLeft,  ebTop,  ebRight,
ebBottom]
    List = True
    ShowCaptions = True
    TabOrder = 0
    object btnSelectManager: TToolButton
      Left = 0
      Top = 0
      AutoSize = True
      Caption   =   #1042#1099#1073#1086#1088'
'#1088#1072#1089#1087#1080#1089#1072#1085#108
0#1103'...'
      ImageIndex = 0
      OnClick = btnSelectManagerClick
    end
  end
  object pgcData: TPageControl
    Left = 0
    Top = 23
    Width = 1004
    Height = 564
    ActivePage = tsGroups
    Align = alClient
    TabOrder = 1
    object tsGroups: TTabSheet
      Caption                                  =
#1043#1088#1091#1087#1087#1099
      object spGroups: TSplitter
        Left = 329
        Top = 25
        Width = 5
        Height = 511
        Color = 13434879
        ParentColor = False
        ExplicitLeft = 323
        ExplicitTop = 23
        ExplicitHeight = 457
      end
      object pnl6: TPanel
        Left = 0
        Top = 0
        Width = 996
        Height = 25
        Align = alTop
        TabOrder = 0
        object l7: TLabel
          AlignWithMargins = True
          Left = 4
          Top = 4
          Width = 30
          Height = 17
          Align = alLeft
          Caption = #1055#1086#1080#1089#1082
          ExplicitHeight = 13
        end
        object edtFindGroup: TEdit
          AlignWithMargins = True
          Left = 40
          Top = 4
          Width = 952
          Height = 17
          Align = alClient
          TabOrder = 0
          OnChange = edtFindGroupChange
          ExplicitHeight = 21
        end
```

```
      end
      object dbgGroups: TDBGridEh
        Left = 0
        Top = 25
        Width = 329
        Height = 511
        Align = alLeft
        Color = 13294591
        DataSource = dsGroups
        DynProps = <>
        EvenRowColor = 16769734
        Flat = True
        IndicatorOptions                    =
[gioShowRowIndicatorEh, gioShowRecNoEh]
        OddRowColor = 16777147
        OptionsEh       =       [dghFixed3D,
dghHighlightFocus,         dghClearSelection,
dghRowHighlight,               dghShowRecNo,
dghColumnResize,              dghColumnMove,
dghExtendVertLines]
        RowDetailPanel.Height = 200
        TabOrder = 1
        OnEnter = dbgGroupsEnter
        Columns = <
          item
            CellButtons = <>
            DynProps = <>
            EditButtons = <>
            FieldName = 'GroupID'
            Footers = <>
          end
          item
            CellButtons = <>
            DynProps = <>
            EditButtons = <>
            FieldName = 'Code'
            Footers = <>
            Width = 178
          end>
        object                 RowDetailData:
TRowDetailPanelControlEh
        end
      end
      object dbgGroupAttr: TDBGridEh
        Left = 334
        Top = 25
        Width = 662
        Height = 511
        Align = alClient
        Color = 16777177
        DataSource = dsGroupAttr
        DynProps = <>
        EvenRowColor = 16769734
        Flat = True
        OddRowColor = 16777147
        TabOrder = 2
        OnEnter = dbgGroupsEnter
        Columns = <
          item
            CellButtons = <>
            DynProps = <>
            EditButtons = <>
            FieldName = 'Name'
            Footers = <>
          end
          item
            CellButtons = <>
            DynProps = <>
            EditButtons = <>
            FieldName = 'Value'
            Footers = <>
          end>
        object                 RowDetailData:
TRowDetailPanelControlEh
        end
      end
    end
    object tsTeachers: TTabSheet
      Caption                               =
#1055#1088#1077#1087#1086#1076#1072#1074#1072
#1090#1077#1083#1080
      ImageIndex = 1
      ExplicitLeft = 0
      ExplicitTop = 0
      ExplicitWidth = 987
      ExplicitHeight = 482
      object spTeacher: TSplitter
        Left = 417
        Top = 25
        Width = 5
        Height = 511
        Color = 13434879
        ParentColor = False
        ExplicitLeft = 479
        ExplicitTop = 6
        ExplicitHeight = 457
      end
      object dbgTeachers: TDBGridEh
        Left = 0
        Top = 25
        Width = 417
        Height = 511
        Align = alLeft
        Color = 13294591
        DataSource = dsTeachers
        DynProps = <>
        EvenRowColor = 16769734
        Flat = True
        IndicatorOptions                    =
[gioShowRowIndicatorEh, gioShowRecNoEh]
        OddRowColor = 16777147
        OptionsEh       =       [dghFixed3D,
dghHighlightFocus,         dghClearSelection,
dghRowHighlight,               dghShowRecNo,
dghColumnResize,              dghColumnMove,
dghExtendVertLines]
        RowDetailPanel.Height = 200
        TabOrder = 0
        OnEnter = dbgGroupsEnter
        Columns = <
          item
            CellButtons = <>
            DynProps = <>
            EditButtons = <>
            FieldName = 'TeacherID'
            Footers = <>
          end
          item
            CellButtons = <>
            DynProps = <>
            EditButtons = <>
            FieldName = 'Name'
            Footers = <>
            Width = 291
          end>
        object                 RowDetailData:
TRowDetailPanelControlEh
        end
      end
      object pnl7: TPanel
        Left = 0
        Top = 0
        Width = 996
        Height = 25
        Align = alTop
        TabOrder = 1
        ExplicitWidth = 987
        object l8: TLabel
          AlignWithMargins = True
          Left = 4
          Top = 4
          Width = 30
          Height = 13
          Align = alLeft
          Caption = #1055#1086#1080#1089#1082
        end
        object edtFilterTutor: TEdit
```

```
        AlignWithMargins = True
        Left = 40
        Top = 4
        Width = 952
        Height = 17
        Align = alClient
        TabOrder = 0
        OnChange = edtFilterTutorChange
        ExplicitWidth = 943
        ExplicitHeight = 21
      end
    end
    object dbgTeacherAttr: TDBGridEh
      Left = 422
      Top = 25
      Width = 574
      Height = 511
      Align = alClient
      Color = 16777177
      DataSource = dsTeacherAttr
      DynProps = <>
      EvenRowColor = 16769734
      Flat = True
      OddRowColor = 16777147
      TabOrder = 2
      OnEnter = dbgGroupsEnter
      Columns = <
        item
          CellButtons = <>
          DynProps = <>
          EditButtons = <>
          FieldName = 'Name'
          Footers = <>
        end
        item
          CellButtons = <>
          DynProps = <>
          EditButtons = <>
          FieldName = 'Value'
          Footers = <>
        end>
      object                RowDetailData:
TRowDetailPanelControlEh
      end
    end
  end
  object tsRooms: TTabSheet
    Caption                                =
#1040#1091#1076#1080#1090#1086#1088#1080#1080
    ImageIndex = 2
    ExplicitLeft = 0
    ExplicitTop = 0
    ExplicitWidth = 987
    ExplicitHeight = 482
    object spGroups11: TSplitter
      Left = 417
      Top = 0
      Width = 5
      Height = 536
      Color = 13434879
      ParentColor = False
      ExplicitLeft = 324
      ExplicitTop = -2
      ExplicitHeight = 482
    end
    object dbgRooms: TDBGridEh
      Left = 0
      Top = 0
      Width = 417
      Height = 536
      Align = alLeft
      Color = 13294591
      DataSource = dsRooms
      DynProps = <>
      EvenRowColor = 16769734
      Flat = True
      IndicatorOptions                     =
[gioShowRowIndicatorEh, gioShowRecNoEh]
      OddRowColor = 16777147
      OptionsEh       =       [dghFixed3D,
dghHighlightFocus,         dghClearSelection,
dghRowHighlight,               dghShowRecNo,
dghColumnResize,              dghColumnMove,
dghExtendVertLines]
      TabOrder = 0
      OnEnter = dbgGroupsEnter
      Columns = <
        item
          CellButtons = <>
          DynProps = <>
          EditButtons = <>
          FieldName = 'RoomID'
          Footers = <>
        end
        item
          CellButtons = <>
          DynProps = <>
          EditButtons = <>
          FieldName = 'Name'
          Footers = <>
        end
        item
          CellButtons = <>
          DynProps = <>
          EditButtons = <>
          FieldName = 'Places'
          Footers = <>
        end
        item
          CellButtons = <>
          DynProps = <>
          EditButtons = <>
          FieldName = 'RoomTypeID'
          Footers = <>
          LookupParams.KeyFieldNames      =
'RoomTypeID'
          LookupParams.LookupDataSet      =
DM.dsRoomTypes

LookupParams.LookupDisplayFieldName = 'Name'
          LookupParams.LookupKeyFieldNames
= 'RoomTypeID'
          Width = 158
        end>
      object                 RowDetailData:
TRowDetailPanelControlEh
      end
    end
    object dbgGroupAttr11: TDBGridEh
      Left = 422
      Top = 0
      Width = 574
      Height = 536
      Align = alClient
      Color = 16777177
      DataSource = dsRoomAttr
      DynProps = <>
      EvenRowColor = 16769734
      Flat = True
      OddRowColor = 16777147
      TabOrder = 1
      OnEnter = dbgGroupsEnter
      Columns = <
        item
          CellButtons = <>
          DynProps = <>
          EditButtons = <>
          FieldName = 'Name'
          Footers = <>
        end
        item
          CellButtons = <>
          DynProps = <>
          EditButtons = <>
          FieldName = 'Value'
          Footers = <>
        end>
```

```
        object                  RowDetailData:
TRowDetailPanelControlEh
        end
      end
    end
    object tsLectureTime: TTabSheet
      Caption   =   #1042#1088#1077#1084#1103'
'#1079#1072#1085#1103#1090#1080#1081
      ImageIndex = 3
      ExplicitLeft = 0
      ExplicitTop = 0
      ExplicitWidth = 987
      ExplicitHeight = 482
      object dbgLectureTime: TDBGridEh
        Left = 0
        Top = 0
        Width = 996
        Height = 536
        Align = alClient
        Color = 13294591
        DataSource = dsLectureTime
        DynProps = <>
        EvenRowColor = 16769734
        Flat = True
        IndicatorOptions                        =
[gioShowRowIndicatorEh, gioShowRecNoEh]
        OddRowColor = 16777147
        OptionsEh       =       [dghFixed3D,
dghHighlightFocus,         dghClearSelection,
dghRowHighlight,               dghShowRecNo,
dghColumnResize,               dghColumnMove,
dghExtendVertLines]
        TabOrder = 0
        OnEnter = dbgGroupsEnter
        object                  RowDetailData:
TRowDetailPanelControlEh
        end
      end
    end
    object tsDays: TTabSheet
      Caption = #1044#1085#1080
      ImageIndex = 6
      ExplicitLeft = 0
      ExplicitTop = 0
      ExplicitWidth = 987
      ExplicitHeight = 482
      object dbgDays: TDBGridEh
        Left = 0
        Top = 0
        Width = 996
        Height = 536
        Align = alClient
        Color = 13294591
        DataSource = dsDays
        DynProps = <>
        EvenRowColor = 16769734
        Flat = True
        IndicatorOptions                        =
[gioShowRowIndicatorEh, gioShowRecNoEh]
        OddRowColor = 16777147
        OptionsEh       =       [dghFixed3D,
dghHighlightFocus,         dghClearSelection,
dghRowHighlight,               dghShowRecNo,
dghColumnResize,               dghColumnMove,
dghExtendVertLines]
        TabOrder = 0
        OnEnter = dbgGroupsEnter
        object                  RowDetailData:
TRowDetailPanelControlEh
        end
      end
    end
    object tsDisciplines: TTabSheet
      Caption                                   =
#1044#1080#1089#1094#1080#1087#1083#1080#1085
#1099
      ImageIndex = 4
      ExplicitLeft = 0
      ExplicitTop = 0
      ExplicitWidth = 987
      ExplicitHeight = 482
      object spGroups12: TSplitter
        Left = 0
        Top = 329
        Width = 996
        Height = 5
        Cursor = crVSplit
        Align = alBottom
        Color = 13434879
        ParentColor = False
        ExplicitLeft = 24
        ExplicitTop = 413
        ExplicitWidth = 987
      end
      object dbgDisciplines: TDBGridEh
        Left = 0
        Top = 25
        Width = 996
        Height = 304
        Align = alClient
        Color = 13294591
        DataSource = dsDisciplines
        DynProps = <>
        EvenRowColor = 16769734
        Flat = True
        IndicatorOptions                        =
[gioShowRowIndicatorEh, gioShowRecNoEh]
        OddRowColor = 16777147
        OptionsEh       =       [dghFixed3D,
dghHighlightFocus,         dghClearSelection,
dghRowHighlight,               dghShowRecNo,
dghColumnResize,               dghColumnMove,
dghExtendVertLines]
        RowDetailPanel.Height = 200
        TabOrder = 0
        OnEnter = dbgGroupsEnter
        Columns = <
          item
            CellButtons = <>
            DynProps = <>
            EditButtons = <>
            FieldName = 'DisciplineID'
            Footers = <>
          end
          item
            CellButtons = <>
            DynProps = <>
            EditButtons = <>
            FieldName = 'Name'
            Footers = <>
            Width = 594
          end>
        object                  RowDetailData:
TRowDetailPanelControlEh
        end
      end
      object pnl8: TPanel
        Left = 0
        Top = 0
        Width = 996
        Height = 25
        Align = alTop
        TabOrder = 1
        ExplicitWidth = 987
        object l9: TLabel
          AlignWithMargins = True
          Left = 4
          Top = 4
          Width = 30
          Height = 13
          Align = alLeft
          Caption = #1055#1086#1080#1089#1082
        end
        object edtFilterDisc: TEdit
          AlignWithMargins = True
          Left = 40
          Top = 4
          Width = 952
```

```
          Height = 17
          Align = alClient
          TabOrder = 0
          OnChange = edtFilterDiscChange
          ExplicitWidth = 943
          ExplicitHeight = 21
        end
      end
      object pnl1: TPanel
        Left = 0
        Top = 334
        Width = 996
        Height = 202
        Align = alBottom
        Caption = 'pnl1'
        TabOrder = 2
        ExplicitTop = 280
        ExplicitWidth = 987
        object lRoomDiscCap: TLabel
          AlignWithMargins = True
          Left = 4
          Top = 4
          Width = 138
          Height = 13
          Align = alTop
          Alignment = taCenter
          Caption                                =
#1040#1091#1076#1080#1090#1086#1088#1080#1080
'                          '#1076#1083#1103'
'#1076#1080#1094#1080#1087#1083#1080#1085#109
9
        end
        object dbgDiscRooms: TDBGridEh
          Left = 1
          Top = 20
          Width = 994
          Height = 181
          Align = alClient
          Color = 13294591
          DataSource = dsRoomDisc
          DynProps = <>
          EvenRowColor = 16769734
          Flat = True
          IndicatorOptions                    =
[gioShowRowIndicatorEh, gioShowRecNoEh]
          OddRowColor = 16777147
          OptionsEh       =       [dghFixed3D,
dghHighlightFocus,          dghClearSelection,
dghRowHighlight,                dghShowRecNo,
dghColumnResize,                dghColumnMove,
dghExtendVertLines]
          TabOrder = 0
          OnEnter = dbgGroupsEnter
          Columns = <
            item
              Alignment = taLeftJustify
              CellButtons = <>
              DynProps = <>
              EditButtons = <>
              FieldName = 'TypeLecture'
              Footers = <>
              LookupParams.KeyFieldNames     =
'TypeLecture'
              LookupParams.LookupDataSet     =
DM.dsTypeLecture

LookupParams.LookupDisplayFieldName = 'Name'

LookupParams.LookupKeyFieldNames             =
'TypeLecture'
              Width = 100
            end
            item
              CellButtons = <>
              DynProps = <>
              EditButtons = <>
              FieldName = 'RoomTypeID'
              Footers = <>
              LookupParams.KeyFieldNames     =
'RoomTypeID'
              LookupParams.LookupDataSet     =
DM.dsRoomTypes

LookupParams.LookupDisplayFieldName = 'Name'

LookupParams.LookupKeyFieldNames             =
'RoomTypeID'
              Width = 104
            end
            item
              CellButtons = <>
              DynProps = <>
              EditButtons = <>
              FieldName = 'RoomID'
              Footers = <>
              LookupParams.KeyFieldNames     =
'RoomID'
              LookupParams.LookupCache       =
False
              LookupParams.LookupDataSet     =
DM.dsRoomList

LookupParams.LookupDisplayFieldName = 'Name'

LookupParams.LookupKeyFieldNames = 'RoomID'
              Width = 102
            end
            item
              CellButtons = <>
              DynProps = <>
              EditButtons = <>
              FieldName = 'TeacherID'
              Footers = <>
              LookupParams.KeyFieldNames     =
'TeacherID'
              LookupParams.LookupCache       =
False
              LookupParams.LookupDataSet     =
DM.dsTeacherList

LookupParams.LookupDisplayFieldName = 'Name'

LookupParams.LookupKeyFieldNames             =
'TeacherID'
              Width = 218
            end
            item
              CellButtons = <>
              DynProps = <>
              EditButtons = <>
              FieldName = 'PrioritySelect'
              Footers = <>
            end>
          object                RowDetailData:
TRowDetailPanelControlEh
          end
        end
      end
    end
    object tsFlows: TTabSheet
      Caption                                    =
#1055#1086#1090#1086#1082#1080
      ImageIndex = 5
      object dbgFlow: TDBGridEh
        Left = 0
        Top = 70
        Width = 996
        Height = 466
        Align = alClient
        Color = 13294591
        DataSource = dsFlows
        DrawMemoText = True
        DynProps = <>
        EvenRowColor = 16769734
        Flat = True
        IndicatorOptions                         =
[gioShowRowIndicatorEh, gioShowRecNoEh]
```

```
        OddRowColor = 16777147
        OptionsEh       =       [dghFixed3D,
dghHighlightFocus,         dghClearSelection,
dghRowHighlight,               dghShowRecNo,
dghColumnResize,              dghColumnMove,
dghExtendVertLines]
        RowDetailPanel.Active = True
        RowDetailPanel.Height = 400
        TabOrder = 0
        OnEnter = dbgGroupsEnter
        Columns = <
          item
            CellButtons = <>
            DynProps = <>
            EditButtons = <>
            FieldName = 'FlowID'
            Footers = <>
          end
          item
            CellButtons = <>
            DynProps = <>
            EditButtons = <>
            FieldName = 'TypeLecture'
            Footers = <>
            LookupParams.KeyFieldNames      =
'TypeLecture'
            LookupParams.LookupDataSet      =
DM.dsTypeLecture

LookupParams.LookupDisplayFieldName = 'Name'
            LookupParams.LookupKeyFieldNames
= 'TypeLecture'
            Width = 87
          end
          item
            CellButtons = <>
            DynProps = <>
            EditButtons = <>
            FieldName = 'HoursCount'
            Footers = <>
            Width = 42
          end
          item
            CellButtons = <>
            DynProps = <>
            EditButtons = <>
            FieldName = 'CountElements'
            Footers = <>
            Width = 47
          end
          item
            CellButtons = <>
            DynProps = <>
            EditButtons = <>
            FieldName = 'Groups'
            Footers = <>
          end
          item
            CellButtons = <>
            DynProps = <>
            EditButtons = <>
            FieldName = 'Discipline'
            Footers = <>
            Width = 495
          end>
        object                  RowDetailData:
TRowDetailPanelControlEh
          object gpFlowInfo: TGridPanel
            Left = 0
            Top = 0
            Width = 904
            Height = 398
            Align = alClient
            ColumnCollection = <
              item
                Value                       =
100.000000000000000000
              end>
            ControlCollection = <
              item
                Column = 0
                Control = pnlFlowInfo
                Row = 1
              end
              item
                Column = 0
                Control = pnlTeacherFlow
                Row = 2
              end
              item
                Column = 0
                Control = pnlFlowAttr
                Row = 0
              end>
            RowCollection = <
              item
                Value = 33.523105440554910000
              end
              item
                Value = 33.317343205967180000
              end
              item
                Value = 33.159551353477900000
              end>
            TabOrder = 0
            object pnlFlowInfo: TPanel
              Left = 1
              Top = 133
              Width = 902
              Height = 131
              Align = alClient
              TabOrder = 0
              object l4: TLabel
                AlignWithMargins = True
                Left = 4
                Top = 4
                Width = 894
                Height = 13
                Align = alTop
                Alignment = taCenter
                Caption                     =
#1043#1088#1091#1087#1087#1099'
'#1087#1086#1090#1086#1082#1072
                ExplicitWidth = 77
              end
              object dbgFlowGroups: TDBGridEh
                Left = 1
                Top = 20
                Width = 900
                Height = 110
                Align = alClient
                Color = 13294591
                DataSource = dsFlowGroups
                DynProps = <>
                EvenRowColor = 16769734
                Flat = True
                IndicatorOptions            =
[gioShowRowIndicatorEh, gioShowRecNoEh]
                OddRowColor = 16777147
                OptionsEh    =    [dghFixed3D,
dghHighlightFocus,         dghClearSelection,
dghRowHighlight,               dghShowRecNo,
dghColumnResize,              dghColumnMove,
dghExtendVertLines]
                TabOrder = 0
                OnEnter = dbgGroupsEnter
                Columns = <
                  item
                    CellButtons = <>
                    DynProps = <>
                    EditButtons = <>
                    FieldName = 'GroupID'
                    Footers = <>

LookupParams.KeyFieldNames = 'GroupID'

LookupParams.LookupDataSet = DM.dsGroupList
```

```
LookupParams.LookupDisplayFieldName = 'Code'

LookupParams.LookupKeyFieldNames = 'GroupID'
                  Width = 136
                end
                item
                  CellButtons = <>
                  DynProps = <>
                  EditButtons = <>
                  FieldName               =
'DisciplineID'
                  Footers = <>

LookupParams.KeyFieldNames = 'DisciplineID'

LookupParams.LookupDataSet = DM.dsDiscList

LookupParams.LookupDisplayFieldName = 'Name'

LookupParams.LookupKeyFieldNames            =
'DisciplineID'
                  Width = 299
                end
                item
                  CellButtons = <>
                  DynProps = <>
                  EditButtons = <>
                  FieldName               =
'CountStudent'
                  Footers = <>
                end
                item
                  CellButtons = <>
                  DynProps = <>
                  EditButtons = <>
                  FieldName = 'TermNumber'
                  Footers = <>
                  Width = 50
                end
                item
                  CellButtons = <>
                  DynProps = <>
                  EditButtons = <>
                  FieldName = 'SubGroup'
                  Footers = <>
                  Width = 50
                end
                item
                  CellButtons = <>
                  DynProps = <>
                  EditButtons = <>
                  FieldName               =
'SubGroupMask'
                  Footers = <>
                  Width = 237
                end>
              object         RowDetailData:
TRowDetailPanelControlEh
              end
            end
          end
          object pnlTeacherFlow: TPanel
            Left = 1
            Top = 264
            Width = 902
            Height = 133
            Align = alClient
            TabOrder = 1
            object l5: TLabel
              AlignWithMargins = True
              Left = 4
              Top = 4
              Width = 894
              Height = 13
              Align = alTop
              Alignment = taCenter
              Caption                     =
#1055#1088#1077#1087#1086#1076#1072#1074#1072
#1090#1077#1083#1080'
'#1087#1086#1090#1086#1082#1072
              ExplicitWidth = 119
            end
            object          dbgFlowTeacher:
TDBGridEh
              Left = 1
              Top = 20
              Width = 900
              Height = 112
              Align = alClient
              Color = 13294591
              DataSource = dsFlowTeacher
              DynProps = <>
              EvenRowColor = 16769734
              Flat = True
              IndicatorOptions            =
[gioShowRowIndicatorEh, gioShowRecNoEh]
              OddRowColor = 16777147
              OptionsEh    =    [dghFixed3D,
dghHighlightFocus,         dghClearSelection,
dghRowHighlight,               dghShowRecNo,
dghColumnResize,               dghColumnMove,
dghExtendVertLines]
              TabOrder = 0
              OnEnter = dbgGroupsEnter
              Columns = <
                item
                  CellButtons = <>
                  DynProps = <>
                  EditButtons = <>
                  FieldName = 'TeacherID'
                  Footers = <>

LookupParams.KeyFieldNames = 'TeacherID'

LookupParams.LookupDataSet = DM.dsTeacherList

LookupParams.LookupDisplayFieldName = 'Name'

LookupParams.LookupKeyFieldNames            =
'TeacherID'
                  Width = 448
                end
                item
                  CellButtons = <>
                  DynProps = <>
                  EditButtons = <>
                  FieldName = 'SubGroup'
                  Footers = <>
                end>
              object         RowDetailData:
TRowDetailPanelControlEh
              end
            end
          end
          object pnlFlowAttr: TPanel
            Left = 1
            Top = 1
            Width = 902
            Height = 132
            Align = alClient
            TabOrder = 2
            object l10: TLabel
              AlignWithMargins = True
              Left = 4
              Top = 4
              Width = 894
              Height = 13
              Align = alTop
              Alignment = taCenter
              Caption                     =
#1040#1090#1088#1080#1073#1091#1090#1099'
'#1087#1086#1090#1086#1082#1072
              ExplicitWidth = 90
            end
            object dbgFlowAttr: TDBGridEh
              Left = 1
              Top = 20
```

```
                Width = 900
                Height = 111
                Align = alClient
                Color = 13294591
                DataSource = dsFlowAttr
                DynProps = <>
                EvenRowColor = 16769734
                Flat = True
                IndicatorOptions            =
[gioShowRowIndicatorEh, gioShowRecNoEh]
                OddRowColor = 16777147
                OptionsEh    =    [dghFixed3D,
dghHighlightFocus,         dghClearSelection,
dghRowHighlight,               dghShowRecNo,
dghColumnResize,               dghColumnMove,
dghExtendVertLines]
                TabOrder = 0
                OnEnter = dbgGroupsEnter
                Columns = <
                  item
                    CellButtons = <>
                    DynProps = <>
                    EditButtons = <>
                    FieldName = 'Name'
                    Footers = <>
                  end
                  item
                    CellButtons = <>
                    DynProps = <>
                    EditButtons = <>
                    FieldName = 'Value'
                    Footers = <>
                  end>
                object         RowDetailData:
TRowDetailPanelControlEh
                end
              end
            end
          end
        end
        object gp2: TGridPanel
          Left = 0
          Top = 0
          Width = 996
          Height = 70
          Align = alTop
          ColumnCollection = <
            item
              SizeStyle = ssAbsolute
              Value = 200.000000000000000000
            end
            item
              Value = 100.000000000000000000
            end>
          ControlCollection = <
            item
              Column = 0
              Control = l6
              Row = 0
            end
            item
              Column = 1
              Control = edtFilterFlowGroup
              Row = 0
            end
            item
              Column = 0
              Control = l61
              Row = 1
            end
            item
              Column = 1
              Control = edtFilterFlowDiscipline
              Row = 1
            end
            item
              Column = 0
              Control = l62
              Row = 2
            end
            item
              Column = 1
              Control = edtFilterFlowTeacher
              Row = 2
            end>
          RowCollection = <
            item
              SizeStyle = ssAbsolute
              Value = 23.000000000000000000
            end
            item
              SizeStyle = ssAbsolute
              Value = 23.000000000000000000
            end
            item
              SizeStyle = ssAbsolute
              Value = 23.000000000000000000
            end>
          TabOrder = 1
          DesignSize = (
            996
            70)
          object l6: TLabel
            Left = 49
            Top = 6
            Width = 104
            Height = 13
            Anchors = []
            Caption                          =
#1060#1080#1083#1100#1090#1088'  '#1087#1086'
'#1075#1088#1091#1087#1087#1077
            Font.Charset = DEFAULT_CHARSET
            Font.Color = clWindowText
            Font.Height = -11
            Font.Name = 'Tahoma'
            Font.Style = [fsBold]
            ParentFont = False
            ExplicitLeft = 45
          end
          object edtFilterFlowGroup: TEdit
            Left = 201
            Top = 1
            Width = 794
            Height = 23
            Align = alClient
            TabOrder = 0
            OnChange = edtFilterFlowChange
            ExplicitHeight = 21
          end
          object l61: TLabel
            Left = 35
            Top = 29
            Width = 132
            Height = 13
            Anchors = []
            Caption                          =
#1060#1080#1083#1100#1090#1088'  '#1087#1086'
'#1076#1080#1089#1094#1080#1087#1083#1080#108
5#1077
            Font.Charset = DEFAULT_CHARSET
            Font.Color = clWindowText
            Font.Height = -11
            Font.Name = 'Tahoma'
            Font.Style = [fsBold]
            ParentFont = False
            ExplicitLeft = 31
          end
          object edtFilterFlowDiscipline: TEdit
            Left = 201
            Top = 24
            Width = 794
            Height = 23
            Align = alClient
            TabOrder = 1
            OnChange = edtFilterFlowChange
            ExplicitHeight = 21
          end
```

```
        object l62: TLabel
          Left = 58
          Top = 52
          Width = 86
          Height = 13
          Anchors = []
          Caption                                =
#1060#1080#1083#1100#1090#1088'  '#1087#1086'
'#1055#1055#1057
          Font.Charset = DEFAULT_CHARSET
          Font.Color = clWindowText
          Font.Height = -11
          Font.Name = 'Tahoma'
          Font.Style = [fsBold]
          ParentFont = False
          ExplicitLeft = 61
        end
        object edtFilterFlowTeacher: TEdit
          Left = 201
          Top = 47
          Width = 794
          Height = 23
          Align = alClient
          TabOrder = 2
          OnChange = edtFilterFlowChange
          ExplicitHeight = 21
        end
      end
    end
  end
  object dbnvgrGroup: TDBNavigator
    Left = 0
    Top = 587
    Width = 1004
    Height = 18
    Align = alBottom
    Flat = True
    TabOrder = 2
  end
  object dsGroups: TDataSource
    DataSet = DM.dsGroups
    Left = 40
    Top = 152
  end
  object dsTeachers: TDataSource
    DataSet = DM.dsTeachers
    Left = 40
    Top = 208
  end
  object dsRooms: TDataSource
    DataSet = DM.dsRooms
    Left = 208
    Top = 152
  end
  object dsLectureTime: TDataSource
    DataSet = DM.dsLectureTime
    Left = 40
    Top = 272
  end
  object dsDisciplines: TDataSource
    DataSet = DM.dsDiscipline
    Left = 304
    Top = 440
  end
  object dsRoomDisc: TDataSource
    DataSet = DM.dsDiscRooms
    Left = 224
    Top = 440
  end
  object dsFlows: TDataSource
    DataSet = DM.dsFlows
    Left = 728
    Top = 408
  end
  object dsFlowGroups: TDataSource
    DataSet = DM.dsFlowGroup
    Left = 808
    Top = 408
  end
  object dsFlowTeacher: TDataSource
    DataSet = DM.dsFlowTeacher
    Left = 808
    Top = 464
  end
  object tmrFilterFlow: TTimer
    Enabled = False
    Interval = 500
    OnTimer = tmrFilterFlowTimer
    Left = 792
    Top = 152
  end
  object tmrFilterGroup: TTimer
    Enabled = False
    Interval = 500
    OnTimer = tmrFilterGroupTimer
    Left = 792
    Top = 104
  end
  object tmrFilterTeacher: TTimer
    Enabled = False
    Interval = 500
    OnTimer = tmrFilterTeacherTimer
    Left = 792
    Top = 200
  end
  object tmrFilterDisc: TTimer
    Enabled = False
    Interval = 500
    OnTimer = tmrFilterDiscTimer
    Left = 792
    Top = 248
  end
  object dsGroupAttr: TDataSource
    DataSet = DM.dsGroupsAttr
    Left = 128
    Top = 152
  end
  object dsTeacherAttr: TDataSource
    DataSet = DM.dsTeacherAttr
    Left = 128
    Top = 208
  end
  object dsRoomAttr: TDataSource
    DataSet = DM.dsRoomAttr
    Left = 208
    Top = 208
  end
  object dsDays: TDataSource
    DataSet = DM.dsDays
    Left = 40
    Top = 328
  end
  object dsFlowAttr: TDataSource
    DataSet = DM.dsFlowAttr
    Left = 728
    Top = 464
  end
end
```

Файл Un_TimetableEdit.pas

```
unit Un_TimetableEdit;

interface

uses Windows, Messages, SysUtils, Variants,
Classes, Graphics, Controls, Forms, ADODB,
Dialogs, ComCtrls, ToolWin, DBGridEhGrouping,
ToolCtrlsEh, DBGridEhToolCtrls, DynVarsEh,
DB, ExtCtrls, DBCtrls, EhLibVCL, GridsEh,
DBAxisGridsEh, DBGridEh, StdCtrls;

type
  TfrmTimetableEdit = class(TForm)
    tlb1: TToolBar;
    btnSelectManager: TToolButton;
    pgcData: TPageControl;
    tsGroups: TTabSheet;
```

```
    tsTeachers: TTabSheet;
    tsRooms: TTabSheet;
    dbnvgrGroup: TDBNavigator;
    dsGroups: TDataSource;
    dbgTeachers: TDBGridEh;
    dsTeachers: TDataSource;
    dsRooms: TDataSource;
    dbgRooms: TDBGridEh;
    tsLectureTime: TTabSheet;
    dbgLectureTime: TDBGridEh;
    dsLectureTime: TDataSource;
    tsDisciplines: TTabSheet;
    dbgDisciplines: TDBGridEh;
    dsDisciplines: TDataSource;
    dsRoomDisc: TDataSource;
    tsFlows: TTabSheet;
    dbgFlow: TDBGridEh;
    dsFlows: TDataSource;
    gpFlowInfo: TGridPanel;
    pnlFlowInfo: TPanel;
    pnlTeacherFlow: TPanel;
    l4: TLabel;
    l5: TLabel;
    dbgFlowGroups: TDBGridEh;
    dbgFlowTeacher: TDBGridEh;
    dsFlowGroups: TDataSource;
    dsFlowTeacher: TDataSource;
    tmrFilterFlow: TTimer;
    pnl6: TPanel;
    l7: TLabel;
    edtFindGroup: TEdit;
    tmrFilterGroup: TTimer;
    pnl7: TPanel;
    l8: TLabel;
    edtFilterTutor: TEdit;
    tmrFilterTeacher: TTimer;
    pnl8: TPanel;
    l9: TLabel;
    edtFilterDisc: TEdit;
    tmrFilterDisc: TTimer;
    gp2: TGridPanel;
    l6: TLabel;
    edtFilterFlowGroup: TEdit;
    l61: TLabel;
    edtFilterFlowDiscipline: TEdit;
    l62: TLabel;
    edtFilterFlowTeacher: TEdit;
    dbgGroups: TDBGridEh;
    dbgGroupAttr: TDBGridEh;
    dsGroupAttr: TDataSource;
    spGroups: TSplitter;
    spTeacher: TSplitter;
    dbgTeacherAttr: TDBGridEh;
    dsTeacherAttr: TDataSource;
    spGroups11: TSplitter;
    dbgGroupAttr11: TDBGridEh;
    dsRoomAttr: TDataSource;
    tsDays: TTabSheet;
    dbgDays: TDBGridEh;
    dsDays: TDataSource;
    spGroups12: TSplitter;
    pnl1: TPanel;
    lRoomDiscCap: TLabel;
    dbgDiscRooms: TDBGridEh;
    pnlFlowAttr: TPanel;
    l10: TLabel;
    dbgFlowAttr: TDBGridEh;
    dsFlowAttr: TDataSource;
    procedure   btnSelectManagerClick(Sender:
TObject);
    procedure          dbgGroupsEnter(Sender:
TObject);
    procedure     edtFilterFlowChange(Sender:
TObject);
    procedure      tmrFilterFlowTimer(Sender:
TObject);
    procedure     tmrFilterGroupTimer(Sender:
TObject);
    procedure      edtFindGroupChange(Sender:
TObject);
    procedure    edtFilterTutorChange(Sender:
TObject);
    procedure   tmrFilterTeacherTimer(Sender:
TObject);
    procedure     edtFilterDiscChange(Sender:
TObject);
    procedure      tmrFilterDiscTimer(Sender:
TObject);
  private
    FManagerID: Integer;
    procedure SetManagerID(Value: Integer);
    procedure SetID(ds: TADODataSet);
    procedure  SetFilter(timer:  TTimer;  ds:
TADODataSet; edit: TEdit; Field: string);
  public
    property    ManagerID:    Integer    read
FManagerID write SetManagerID;
  end;

var
  frmTimetableEdit: TfrmTimetableEdit;

implementation

uses Un_TimetableManager, Un_TimetableData;

{$R *.dfm}

procedure
TfrmTimetableEdit.btnSelectManagerClick(Sende
r: TObject);
begin
  TfrmTimetableManager.Create;
  if  not  DM.fManagerManagerID.IsNull  then
ManagerID := DM.fManagerManagerID.Value;
end;

procedure
TfrmTimetableEdit.dbgGroupsEnter(Sender:
TObject);
begin
  dbnvgrGroup.DataSource                    :=
TDBGridEh(Sender).DataSource;
end;

procedure
TfrmTimetableEdit.edtFilterDiscChange(Sender:
TObject);
begin
  tmrFilterDisc.Enabled := False;
  tmrFilterDisc.Enabled := True;
end;

procedure
TfrmTimetableEdit.edtFilterFlowChange(Sender:
TObject);
begin
  tmrFilterFlow.Enabled := False;
  tmrFilterFlow.Enabled := True;
end;

procedure
TfrmTimetableEdit.edtFilterTutorChange(Sender
: TObject);
begin
  tmrFilterTeacher.Enabled := False;
  tmrFilterTeacher.Enabled := True;
end;

procedure
TfrmTimetableEdit.edtFindGroupChange(Sender:
TObject);
begin
  tmrFilterGroup.Enabled := False;
  tmrFilterGroup.Enabled := True;
end;
```

```
procedure  TfrmTimetableEdit.SetFilter(timer:
TTimer; ds: TADODataSet; edit: TEdit; Field:
string);
begin
  timer.Enabled := False;
  ds.Filtered := False;
  if Length(Trim(edit.Text)) > 0 then
    begin
      ds.Filter  :=  Field  +  '  LIKE  '  +
QuotedStr('%' + Trim(edit.Text) + '%');
      ds.Filtered := True;
    end;
end;

procedure         TfrmTimetableEdit.SetID(ds:
TADODataSet);
begin
  DM.ReOpen(ds, 'ID', FManagerID);
end;

procedure
TfrmTimetableEdit.SetManagerID(Value:
Integer);
begin
  FManagerID := Value;
  btnSelectManager.Caption                  :=
DM.fManagerName.Value   +   '   (ID:   '   +
IntToStr(FManagerID) + ')';
  SetID(DM.dsGroups);
  SetID(DM.dsTeachers);
  SetID(DM.dsRooms);
  SetID(DM.dsLectureTime);
  SetID(DM.dsDays);
  SetID(DM.dsDiscipline);
  SetID(DM.dsFlows);
end;

procedure
TfrmTimetableEdit.tmrFilterDiscTimer(Sender:
TObject);
begin
  SetFilter(tmrFilterDisc,   DM.dsDiscipline,
edtFilterDisc, 'Name');
end;

procedure
TfrmTimetableEdit.tmrFilterFlowTimer(Sender:
TObject);
begin
  tmrFilterFlow.Enabled := False;
  if  Length(Trim(edtFilterFlowGroup.Text))  +
Length(Trim(edtFilterFlowDiscipline.Text)) +
     Length(Trim(edtFilterFlowTeacher.Text))
> 0 then
    DM.ReOpen(DM.dsFlows,     ['FilterGroup',
'FilterDisc', 'FilterTeacher'],
      [Trim(edtFilterFlowGroup.Text),
Trim(edtFilterFlowDiscipline.Text),
Trim(edtFilterFlowTeacher.Text)])
  else
    DM.ReOpen(DM.dsFlows,     ['FilterGroup',
'FilterDisc', 'FilterTeacher'], [Null, Null,
Null]);
end;

procedure
TfrmTimetableEdit.tmrFilterGroupTimer(Sender:
TObject);
begin
  SetFilter(tmrFilterGroup,      DM.dsGroups,
edtFindGroup, 'Code');
end;

procedure
TfrmTimetableEdit.tmrFilterTeacherTimer(Sende
r: TObject);
begin
  SetFilter(tmrFilterTeacher,  DM.dsTeachers,
edtFilterTutor, 'Name');
end;

end.
```

Приложение В – Исходный код динамический библиотеки

Файл Dict\Discipline.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Flow;

namespace ektu.Timetable.Dict
{
    /// <summary>
    /// Дисциплина
    /// </summary>
    public class Discipline : Item
    {
        /// <summary>
        /// Идентификатор дисциплины
        /// </summary>
        public int DisciplineID { get;
private set; }
        /// <summary>
        /// Аудитории для дисциплины
        /// </summary>
        public Dictionary<TypeFlowBase,
List<string>> Rooms { get; private set; }
        /// <summary>
        /// Конструктор
        /// </summary>
        public Discipline(): base()
        {
            Rooms = new
Dictionary<TypeFlowBase, List<string>>();
        }

        protected override void
SetValue(string name, object value)
        {
            if (name == "DisciplineID")
            {
                DisciplineID = (int)value;
            }
        }
    }
}
```

Файл Dict\Group.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ektu.Timetable.Dict
{
    /// <summary>
    /// Группа
    /// </summary>
    public class Group : Item
    {
        /// <summary>
        /// Идентификатор группы
        /// </summary>
        public int GroupID { get; private
set; }
        /// <summary>
        /// Строковое представление для
группы
        /// </summary>
        /// <returns>Строковое представление
для группы</returns>
        public override string ToString()
        {
            return "GroupID=" +
GroupID.ToString();
        }

        protected override void
SetValue(string name, object value)
        {
            if (name == "GroupID")
            {
                GroupID = (int)value;
            }
        }
    }
}
```

Файл Dict\Item.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ektu.Timetable.Dict
{
    /// <summary>
    /// Базовый элемент для элемента
    /// </summary>
    public abstract class Item
    {
        /// <summary>
        /// Метод обработки установки
атрибута
        /// </summary>
        /// <param name="name">Имя
атрибута</param>
        /// <param name="value">Значение
атрибута</param>
        protected abstract void
SetValue(string name, object value);
        /// <summary>
        /// Список атрибутов для элемента
        /// </summary>
        private Dictionary<string, object>
Attributes { get; set; }
        /// <summary>
        /// Получить значение элемента
        /// </summary>
        /// <typeparam name="T">Тип
атрибута</typeparam>
        /// <param name="name">Название
атрибута</param>
        /// <returns>Значение
атрибута</returns>
        public T Value<T>(string name)
        {
            return (T)Attributes[name];
        }
        /// <summary>
        /// Атрибут элемента
        /// </summary>
        /// <param name="name">Имя
атрибута</param>
        /// <returns>Значение
атрибута</returns>
        public virtual object this[string
name]
        {
            get { return Attributes[name]; }
            set
            {
                Attributes[name] = value;
                SetValue(name, value);
            }
        }
```

```
        /// <summary>
        /// Проверка наличия атрибута
        /// </summary>
        ///       <param       name="name">Имя
атрибута</param>
        /// <returns>Да/Нет</returns>
        public bool HasAttribute(string name)
        {
            return
Attributes.ContainsKey(name);
        }
        /// <summary>
        /// Конструктор
        /// </summary>
        public Item(): base()
        {
            Attributes          =          new
Dictionary<string, object>();
        }
    }
}
```

Файл Dict\LectureItem.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ektu.Timetable.Dict
{
    /// <summary>
    /// Время занятия
    /// </summary>
    public class LectureTime : Item
    {
        /// <summary>
        /// Номер занятия
        /// </summary>
        public   int   NumberLecture   {   get;
private set; }
        /// <summary>
        /// Идентфикатор времени занятия
        /// </summary>
        public   int   LectureTimeID   {   get;
private set; }
        /// <summary>
        /// Следующее по номер время занятия
        /// </summary>
        public LectureTime Next { get; set; }
        /// <summary>
        /// Начало занаятия
        /// </summary>
        public  TimeSpan  StartTime  {  get;
private set; }
        /// <summary>
        /// Завершение занятия
        /// </summary>
        public  TimeSpan  FinishTime  {  get;
private set; }
        /// <summary>
        ///   Строковое   представление   для
времени занятия
        /// </summary>
        /// <returns>Строковое представление
времени занятия</returns>
        public override string ToString()
        {
            return     "LectureTimeID="     +
LectureTimeID.ToString();
        }

        public List<LectureTime> Cross { get;
set; }

        public    LectureTime    NextHours(int
hours)
        {
            LectureTime lt = this;
            while(--hours > 0)
            {
                lt = lt.Next;
            }
            return lt;
        }

        protected        override        void
SetValue(string name, object value)
        {
            if (name == "NumberLecture")
            {
                NumberLecture = (int)value;
            }
            if (name == "LectureTimeID")
            {
                LectureTimeID = (int)value;
            }
            if (name == "StartTime")
            {
                StartTime = (TimeSpan)value;
            }
            if (name == "FinishTime")
            {
                FinishTime = (TimeSpan)value;
            }
        }
    }
}
```

Файл Dict\Room.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ektu.Timetable.Dict
{
    /// <summary>
    /// Аудитория
    /// </summary>
    public class Room : Item
    {
        /// <summary>
        /// Идентификатор аудитории
        /// </summary>
        public int RoomID { get; private set;
}
        /// <summary>
        /// Число мест в аудитории
        /// </summary>
        public int Places { get; private set;
}
        /// <summary>
        ///   Строковое   представление   для
аудитории
        /// </summary>
        ///  <returns>Строковое представление
аудитории</returns>
        public override string ToString()
        {
            return        "RoomID="        +
RoomID.ToString();
        }
        protected        override        void
SetValue(string name, object value)
        {
            switch(name)
            {
                case "RoomID":
                    RoomID = (int)value;
                    break;
                case "Places":
                    Places = (int)value;
                    break;
            }
```

```
        }
    }
}
```

Файл Dict\Teacher.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ektu.Timetable.Dict
{
    /// <summary>
    /// Преподаватель
    /// </summary>
    public class Teacher : Item
    {
        /// <summary>
        /// Идентфикатор преподавателя
        /// </summary>
        public int TeacherID { get; private
set; }
        /// <summary>
        ///   Строковое   представление   для
преподавателя
        /// </summary>
        /// <returns>Строковое предствление
преподвателя</returns>
        public override string ToString()
        {
            return       "TeacherID="       +
TeacherID.ToString();
        }

        protected        override        void
SetValue(string name, object value)
        {
            if (name == "TeacherID")
            {
                TeacherID = (int)value;
            }
        }
    }
}
```

Файл Dict\WeekDayItem.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Reflection;

namespace ektu.Timetable.Dict
{
    /// <summary>
    /// День
    /// </summary>
    public     abstract     class     DayItem:
IComparable<DayItem>
    {
        /// <summary>
        /// Идентификатор
        /// </summary>
        public abstract int ID { get; }
        /// <summary>
        /// Сравнение дней
        /// </summary>
        ///     <param     name="other">Другой
день</param>
        ///       <returns>Разница       между
днями</returns>
        public int CompareTo(DayItem other)
        {
            return ID - other.ID;
        }

        public  List<DayItem>  Cross  =  new
List<DayItem>();
    }

    /// <summary>
    /// День недели
    /// </summary>
    public sealed class WeekDayItem : DayItem
    {
        /// <summary>
        /// ИД дня
        /// </summary>
        private int WeekDayID;
        /// <summary>
        /// Идентификатор для недели
        /// </summary>
        public override int ID
        {
            get { return WeekDayID; }
        }
        /// <summary>
        ///   Текстовое   представление   дня
недели
        /// </summary>
        ///       <returns>Название       дня
недели</returns>
        public override string ToString()
        {
            switch(ID)
            {
                case 1:
                    return "Monday";
                case 2:
                    return "Tuesday";
                case 3:
                    return "Wednesday";
                case 4:
                    return "Thursday";
                case 5:
                    return "Friday";
                case 6:
                    return "SaturDay";
                case 7:
                    return "Sunday";
                default:
                    return "";
            }
        }

        /// <summary>
        /// Понедельник
        /// </summary>
        public  static  WeekDayItem  Monday  =
new WeekDayItem() { WeekDayID = 1 };
        /// <summary>
        /// Вторник
        /// </summary>
        public  static  WeekDayItem  Tuesday  =
new WeekDayItem() { WeekDayID = 2 };
        /// <summary>
        /// Среда
        /// </summary>
        public static WeekDayItem Wednesday =
new WeekDayItem() { WeekDayID = 3 };
        /// <summary>
        /// Четверг
        /// </summary>
        public static WeekDayItem Thursday =
new WeekDayItem() { WeekDayID = 4 };
        /// <summary>
        /// Пятница
        /// </summary>
        public  static  WeekDayItem  Friday  =
new WeekDayItem() { WeekDayID = 5 };
        /// <summary>
        /// Суббота
        /// </summary>
```

```
        public static WeekDayItem Saturday =
new WeekDayItem() { WeekDayID = 6 };
        /// <summary>
        /// Воскресение
        /// </summary>
        public static WeekDayItem Sunday =
new WeekDayItem() { WeekDayID = 7 };
    }

    /// <summary>
    /// Календарный день
    /// </summary>
    public sealed class DateDayItem : DayItem
    {
        /// <summary>
        /// Дата
        /// </summary>
        private DateTime date;
        /// <summary>
        /// Идентификатор
        /// </summary>
        private int id;
        /// <summary>
        /// Идентфикатор
        /// </summary>
        public override int ID
        {
            get { return id; }
        }
        /// <summary>
        /// Создать каледарный день из даты
        /// </summary>
        /// <param name="dt">Дата</param>
        ///    <returns>Объект    календарного
дня</returns>
        public static int DateTimeID(DateTime
dt)
        {
            return   dt.Year   *   100000   +
dt.Month * 100 + dt.Day;
        }
        /// <summary>
        ///  Конструктор календарного дня из
даты
        /// </summary>
        /// <param name="d">Дата</param>
        public    DateDayItem(DateTime    d):
base()
        {
            date = d;
            id = DateTimeID(d);
        }
        /// <summary>
        /// Текствое представление дня
        /// </summary>
        ///    <returns>Строка    в    формает
ДД.ММ.ГГГГ</returns>
        public override string ToString()
        {
            return
date.ToString("dd.MM.yyyy");
        }
    }

    /// <summary>
    /// День недели с разбивкой на 2 недели:
числитель, знаменатель, обе недели
    /// </summary>
    public   sealed   class   WeekDay2Item   :
DayItem
    {
        public enum WeekType { Both, Week1,
Week2  };
        private int WeekDayID;
        private WeekType WeekTypeID;
        public override int ID
        {
            get
            {
                return  WeekDayID  *  100  +
Convert.ToInt32(WeekTypeID);
            }
        }

        public int DayID
        {
            get { return WeekDayID; }
        }
        public WeekType WeekNumber
        {
            get { return WeekTypeID; }
        }

        public override string ToString()
        {
            switch (WeekDayID)
            {
                case 1:
                    return   "Monday,   "   +
WeekTypeID.ToString();
                case 2:
                    return   "Tuesday,   "   +
WeekTypeID.ToString();
                case 3:
                    return  "Wednesday,  "  +
WeekTypeID.ToString();
                case 4:
                    return   "Thursday,   "   +
WeekTypeID.ToString();
                case 5:
                    return   "Friday,   "   +
WeekTypeID.ToString();
                case 6:
                    return   "Saturday,   "   +
WeekTypeID.ToString();
                case 7:
                    return   "Sunday,   "   +
WeekTypeID.ToString();
                default:
                    return "";
            }
        }

        private   static   List<WeekDay2Item>
OneDayData(int DayID)
        {
            WeekDay2Item     d1     =     new
WeekDay2Item()    {    WeekDayID    =    DayID,
WeekTypeID = WeekType.Both };
            WeekDay2Item     d2     =     new
WeekDay2Item()    {    WeekDayID    =    DayID,
WeekTypeID = WeekType.Week1 };
            WeekDay2Item     d3     =     new
WeekDay2Item()    {    WeekDayID    =    DayID,
WeekTypeID = WeekType.Week2 };
            d1.Cross.Add(d2);
            d1.Cross.Add(d3);
            //d2.Cross.Add(d1);
            //d3.Cross.Add(d1);
            List<WeekDay2Item>  data  =  new
List<WeekDay2Item>();
            data.Add(d1);
            data.Add(d2);
            data.Add(d3);
            return data;
        }

        public    static    List<WeekDay2Item>
Monday = OneDayData(1);
        public    static    List<WeekDay2Item>
Tuesday = OneDayData(2);
        public    static    List<WeekDay2Item>
Wednesday = OneDayData(3);
        public    static    List<WeekDay2Item>
Thursday = OneDayData(4);
        public    static    List<WeekDay2Item>
Friday = OneDayData(5);
```

```
        public   static   List<WeekDay2Item>
Saturday = OneDayData(6);
        public   static   List<WeekDay2Item>
Sunday = OneDayData(7);
    }
}
```

Файл Flow\FlowData.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Dict;
using System.Security.Cryptography;

namespace ektu.Timetable.Flow
{

    /// <summary>
    /// Класс для вида занятия
    /// </summary>
    public abstract class TypeFlowBase
    {
        /// <summary>
        /// Идентификатор
        /// </summary>
        public abstract int ID { get; }
        /// <summary>
        /// Название
        /// </summary>
        public abstract string Name { get; }
    }

    /// <summary>
    /// Вид занятия - лекция
    /// </summary>
    public   sealed   class   TypeFlowLecture:
TypeFlowBase
    {
        public override int ID
        {
            get { return 0; }
        }
        public override string Name
        {
            get { return "Lecture"; }
        }
        private TypeFlowLecture() { }
        public static TypeFlowLecture Item {
get; } = new TypeFlowLecture();
    }

    /// <summary>
    /// Вид занятия - практика
    /// </summary>
    public   sealed   class   TypeFlowPractice:
TypeFlowBase
    {
        public override int ID
        {
            get { return 1; }
        }
        public override string Name
        {
            get { return "Practice"; }
        }
        private TypeFlowPractice() { }
        public static TypeFlowPractice Item {
get; } = new TypeFlowPractice();
    }

    /// <summary>
    /// Вид занятия - лабораторная
    /// </summary>
    public sealed class TypeFlowLaboratory :
TypeFlowBase
    {
        public override int ID
        {
            get { return 2; }
        }
        public override string Name
        {
            get { return "Laboratory"; }
        }
        private TypeFlowLaboratory() { }
        public static TypeFlowLaboratory Item
{ get; } = new TypeFlowLaboratory();
    }

    /// <summary>
    /// Аудитория для потока
    /// </summary>
    public sealed class RoomSelectInfo
    {
        private Tuple<int, Room> data;
        /// <summary>
        /// Конструктор
        /// </summary>
        ///                             <param
name="priority">Приоритет</param>
        ///                             <param
name="room">Аудитория</param>
        public  RoomSelectInfo(int  priority,
Room room)
        {
            data     =     new     Tuple<int,
Room>(priority, room);
        }
        /// <summary>
        /// Приоритет
        /// </summary>
        public int Priority => data.Item1;
        /// <summary>
        /// Аудитория
        /// </summary>
        public Room Room => data.Item2;
    }

    /// <summary>
    ///  День  недели  и  время  занятий  для
потока
    /// </summary>
    public class SelectDataWeekTime
    {
        private  Tuple<DayItem,  LectureTime,
int> data;
        /// <summary>
        /// Конструктор
        /// </summary>
        ///       <param      name="day">День
недели</param>
        ///       <param      name="lt">Время
занятия</param>
        ///                             <param
name="priority">Приоритет</param>
        public     SelectDataWeekTime(DayItem
day, LectureTime lt, int priority)
        {
            data    =    new    Tuple<DayItem,
LectureTime, int>(day, lt, priority);
        }
        /// <summary>
        /// День недели
        /// </summary>
        public DayItem Day => data.Item1;
        /// <summary>
        /// Время занятия
        /// </summary>
        public     LectureTime     Time     =>
data.Item2;
        /// <summary>
        /// Приоритет
        /// </summary>
        public int Priority => data.Item3;
    }
```

```
    /// <summary>
    /// Поток
    /// </summary>
    public class FlowData
    {
        /// <summary>
        /// Тип занятия
        /// </summary>
        public TypeFlowBase TypeFlow { get;
set; }
        /// <summary>
        /// Преподаватели
        /// </summary>
        public Dictionary<int, Teacher>
Teachers { get; private set; }
        /// <summary>
        /// Число часов
        /// </summary>
        public int CountHours { get; set; }
        /// <summary>
        /// Параллельные занятия
        /// </summary>
        public List<FlowData> Par { get;
private set; }
        /// <summary>
        /// Связанные гены
        /// </summary>
        public List<FlowData> Linked { get;
private set; }
        /// <summary>
        /// Первая дисциплина
        /// </summary>
        public Discipline FirstDiscipline
        {
            get { return Items[0].Discipline;
}
        }
        /// <summary>
        /// Элементы потока
        /// </summary>
        public List<FlowItem> Items { get;
private set; }
        public Dictionary<string, object>
FlowParam { get; private set; }
        /// <summary>
        /// Конструктор
        /// </summary>
        public FlowData()
        {
            Teachers = new Dictionary<int,
Teacher>();
            Items = new List<FlowItem>();
            Par = new List<FlowData>();
            Linked = new List<FlowData>();
            Rooms = new
List<RoomSelectInfo>();
            DayTimes = new
List<SelectDataWeekTime>();
            FlowParam = new
Dictionary<string, object>();
        }
        /// <summary>
        /// Число студентов в потоке
        /// </summary>
        public int CountStudent
        {
            get
            {
                int v = Items.Sum(f =>
f.CountStudent);
                if (Par.Count == 0)
                {
                    return v;
                }
                else
                {
                    double p = Items.Select(f
=> f.SubGroup).Distinct().Count();
                    return
Convert.ToInt32(Math.Ceiling(v / p));
                }
            }
        }
        /// <summary>
        /// Идентфикатор потока
        /// </summary>
        public string FlowId { get; set; }
        /// <summary>
        /// Аудитории для потока
        /// </summary>
        public List<RoomSelectInfo> Rooms {
get; private set; }
        /// <summary>
        /// Дни недели и время занятия для
потока
        /// </summary>
        public List<SelectDataWeekTime>
DayTimes { get; private set; }

    }
}
```

Файл Flow\FlowItem.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Dict;

namespace ektu.Timetable.Flow
{
    /// <summary>
    /// Элемент потока
    /// </summary>
    public class FlowItem
    {
        /// <summary>
        /// Дисциплина
        /// </summary>
        public Discipline Discipline {
get; set; }
        /// <summary>
        /// Группа
        /// </summary>
        public Group Group { get; set; }
        /// <summary>
        /// Подгруппа
        /// </summary>
        public int SubGroup { get; set; }
        /// <summary>
        /// Битовая маска для подгруппы
        /// </summary>
        public BitArray SubGroupMask {
get; set; }
        /// <summary>
        /// Число студентов
        /// </summary>
        public int CountStudent { get; set; }
    }
}
```

Файл Flow\ForelClassificator.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ektu.Timetable.Flow
{
    /// <summary>
    /// Кластеризация Forel
    /// </summary>
```

```
    public class ForelClassificator
    {
        private    Dictionary<Tuple<FlowData,
FlowData>,     double>     matrix     =     new
Dictionary<Tuple<FlowData,          FlowData>,
double>();
        private List<FlowData> flowList = new
List<FlowData>();

        /// <summary>
        /// Поиск центра кластера
        /// </summary>
        ///    <param    name="cluster">Список
учебных поток входящих в кластер</param>
        /// <returns>Центр кластера</returns>
        private                         FlowData
FindCenter(List<FlowData> cluster)
        {
            Dictionary<FlowData,      double>
distances    =    new    Dictionary<FlowData,
double>();
            foreach    (FlowData    flow1    in
cluster)
            {
                double ds = 0;
                foreach  (FlowData  flow2  in
cluster)
                {
                    if (flow1 == flow2)
                    {
                        continue;
                    }
                    Tuple<FlowData, FlowData>
d  =  new  Tuple<FlowData,  FlowData>(flow1,
flow2);
                    if
(matrix.ContainsKey(d))
                    {
                        ds     +=     1     -
Convert.ToDouble(matrix[d]);
                    }
                    else
                    {
                        d        =        new
Tuple<FlowData, FlowData>(flow2, flow1);
                        if
(matrix.ContainsKey(d))
                        {
                            ds    +=    1    -
Convert.ToDouble(matrix[d]);
                        }
                    }
                }
                distances.Add(flow1, ds);
            }
            double        minDistance        =
double.MaxValue;
            FlowData center = null;
            foreach   (KeyValuePair<FlowData,
double> d in distances)
            {
                if (d.Value < minDistance)
                {
                    minDistance = d.Value;
                    center = d.Key;
                }
            }
            return center;
        }

        /// <summary>
        /// Поиск кластера
        /// </summary>
        /// <param name="data">Список учебных
потоков</param>
        ///  <param  name="center_flow">Центр
кластера</param>
        ///       <param       name="R">Радиус
поиска</param>
        /// <returns>Кластер</returns>
        private                List<FlowData>
FindCluster(List<FlowData>   data,   FlowData
center_flow, double R)
        {
            List<FlowData> nearestFlow = new
List<FlowData>();
            foreach (FlowData flow in data)
            {
                if (flow == center_flow)
                {
                    nearestFlow.Add(flow);
                }
                else
                {
                    Tuple<FlowData, FlowData>
d        =        new        Tuple<FlowData,
FlowData>(center_flow, flow);
                    if
(matrix.ContainsKey(d))
                    {
                        if        (1        -
Convert.ToDouble(matrix[d]) <= R)
                        {

nearestFlow.Add(flow);
                        }
                    }
                    else
                    {
                        d        =        new
Tuple<FlowData, FlowData>(flow, center_flow);
                        if
(matrix.ContainsKey(d))
                        {
                            if      (1      -
Convert.ToDouble(matrix[d]) <= R)
                            {

nearestFlow.Add(flow);
                            }
                        }
                    }
                }
            }

            FlowData         center         =
FindCenter(nearestFlow);
            if  ((center  ==  center_flow)  &&
(nearestFlow.Count != 0))
            {
                return nearestFlow;
            }
            else
            {
                if (center == null)
                {
                    return null;
                }
                return      FindCluster(data,
center, R);
            }
        }

        /// <summary>
        /// Конструктор
        /// </summary>
        ///     <param     name="flows">Список
потоков</param>
        ///  <param name="groupValue">Весовой
коэффицент для связи групп</param>
        ///  <param name="tutorValue">Весовой
коэффицент для связи ППС</param>
        ///  <param  name="roomValue">Весовой
коэффициент для связи аудиторий</param>
        public
ForelClassificator(List<FlowData>      flows,
double groupValue, double tutorValue, double
roomValue)
```

```
        {
            for (int i = 0; i < flows.Count;
i++)
            {
                for (int j = i; j <
flows.Count; j++)
                {
                    Tuple<FlowData, FlowData>
d = new Tuple<FlowData, FlowData>(flows[i],
flows[j]);

                    if ((i == j) &&
(flows[i].Par.Contains(flows[j])))
                    {
                        matrix[d] = 1;
                    }
                    else
                    {

                        int groupShared =
flows[i].Items.Count(f =>
flows[j].Items.Exists(fx => fx.Group ==
f.Group));
                        int groupAll =
flows[i].Items.Count + flows[j].Items.Count -
groupShared;

                        int tutorShared =
flows[i].Teachers.Count(f =>
flows[j].Teachers.ContainsKey(f.Key));
                        int tutorAll =
flows[i].Teachers.Count +
flows[j].Teachers.Count - tutorShared;

                        int roomShared =
flows[i].Rooms.Count(f =>
flows[j].Rooms.Exists(fx => fx.Room ==
f.Room));
                        int roomAll =
flows[i].Rooms.Count + flows[j].Rooms.Count -
roomShared;

                        matrix[d] =
groupShared / Convert.ToDouble(groupAll) *
groupValue +
                            (tutorAll == 0 ?
1 : tutorShared / Convert.ToDouble(tutorAll))
* tutorValue +
                            roomShared /
Convert.ToDouble(roomAll) * roomValue;
                    }
                }
            }

            flowList = new
List<FlowData>(flows);
        }

        /// <summary>
        /// Получение кластеров
        /// </summary>
        ///       <param       name="R">Радиус
поиска</param>
        ///                   <returns>Список
кластеров</returns>
        public           List<List<FlowData>>
GetClusters(double R)
        {
            List<FlowData>    fl    =    new
List<FlowData>(flowList);
            List<List<FlowData>>  clusters  =
new List<List<FlowData>>();
            int cc = 0;
            while (fl.Count > 0)
            {
                Random random = new Random();
                FlowData   center_cluster   =
fl[random.Next(fl.Count - 1)];
                List<FlowData>   cluster   =
FindCluster(fl, center_cluster, R);
                if (cluster == null)
                {
                    cc++;
                }
                else
                {
                    clusters.Add(cluster);
                    foreach (FlowData flow in
cluster)
                    {
                        fl.Remove(flow);
                    }
                }
                if (cc == 50)
                {
                    break;
                }
            }
            return clusters;
        }
    }
}
```

Файл Restrict\Restrict.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Timetable;
using ektu.Timetable.Dict;

namespace ektu.Timetable.Restrict
{
    public           delegate           bool
FilterRestrictDelegate<T>(T item);

    /// <summary>
    /// Интерфейс для классов ограничений
    /// </summary>
    public interface IRestrict
    {
        /// <summary>
        ///     Проверка    расписания    на
ограничение
        /// </summary>
        ///                             <param
name="timetable">Расписание</param>
        /// <returns>Значение от 0 (полностью
не     выполнено)     до     1     (полностью
выполнено)</returns>
        double Check(TimetableData timetable,
double hard, double soft);
        /// <summary>
        /// Коэффициент для ограничения
        /// </summary>
        double Factor { get; }
        string Name { get; }
    }
}
```

Файл Restrict\RestrictDayDiscipline.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Timetable;
using ektu.Timetable.Dict;
using ektu.Timetable.Flow;

namespace ektu.Timetable.Restrict
{
```

```
    using ResultData = Tuple<int, int>;
    using GroupTimetable = KeyValuePair<Group, TimetableSubject>;
    using DefectItem = Tuple<TimetableItem, DayItem>;

    /// <summary>
    /// Ограничение для числа дисциплин в течении дня для группы
    /// </summary>
    public class RestrictDayDiscipline : IRestrict
    {
        /// <summary>
        /// Коэффициент для ограничения
        /// </summary>
        public double Factor { get; private set; }

        /// <summary>
        /// Название ограничения
        /// </summary>
        public string Name { get; private set; }

        /// <summary>
        /// Конструктор
        /// </summary>
        ///                                 <param name="Factor">Коэффициент</param>
        ///    <param name="Name">Название ограничения</param>
        public RestrictDayDiscipline(double Factor = 1, string Name = "RestrictDayDiscipline")
        {
            this.Factor = Factor;
            this.Name = Name;
        }

        /// <summary>
        /// Фильтр для ограничения
        /// </summary>
        public FilterRestrictDelegate<Group> Filter { get; set; }

        /// <summary>
        /// Фильтр для потоков
        /// </summary>
        public FilterRestrictDelegate<FlowData> FilterFlow { get; set; }

        /// <summary>
        /// Пометка нарушения ограничения
        /// </summary>
        /// <param name="g">Группа</param>
        /// <param name="d">День</param>
        /// <returns></returns>
        private string DefectName(Group g, DayItem d)
        {
            return Name + "=" + g.GroupID.ToString() + "," + d.ID.ToString();
        }

        /// <summary>
        /// Вычисление ограничения по группе
        /// </summary>
        ///     <param name="s">Расписание группы</param>
        ///     <param name="hard">жесткое ограничение</param>
        ///     <param name="soft">мягкое ограничение</param>
        ///   <returns>Значение ограничения (число нарушений, всего элементов)</returns>
        private ResultData calc(GroupTimetable s, double hard, double soft)
        {
            int cv = 0;
            int ccv = 0;
            foreach (KeyValuePair<DayItem, Dictionary<LectureTime, List<TimetableItemProxy>>> wd in s.Value.AllData)
            {
                ResultData cacheDay = s.Value.GetCacheWeekDay<ResultData>(wd.Key, this);
                if (cacheDay == null)
                {
                    HashSet<DefectItem> defList = new HashSet<DefectItem>();


List<Tuple<HashSet<Discipline>, HashSet<Discipline>, HashSet<DefectItem>, HashSet<DefectItem>>> st =
                        new List<Tuple<HashSet<Discipline>, HashSet<Discipline>, HashSet<DefectItem>, HashSet<DefectItem>>>();

                    foreach (KeyValuePair<LectureTime, List<TimetableItemProxy>> lt in wd.Value)
                    {
                        foreach (TimetableItemProxy i in lt.Value)
                        {
                            if (i.IsBlank) continue;

i.data.SetDefected(DefectName(s.Key, wd.Key), false, hard, soft);
                            FlowItem fd = i.data.Flow.Items.Where(f => f.Group == s.Key).First();
                            if (st.Count == 0)
                            {
                                for(int k = 0; k <= fd.SubGroupMask.Count; k++)
                                {

st.Add(new Tuple<HashSet<Discipline>, HashSet<Discipline>, HashSet<DefectItem>, HashSet<DefectItem>>(
                                        new HashSet<Discipline>(), new HashSet<Discipline>(), new HashSet<DefectItem>(), new HashSet<DefectItem>()));
                                }
                            }
                            if ((i.data.Flow.TypeFlow is TypeFlowLecture) && ((FilterFlow == null) || (FilterFlow(i.data.Flow))))
                            {
                                for (int k = 0; k < fd.SubGroupMask.Count; k++)
                                {
                                    if (fd.SubGroupMask[k])
                                    {

st[k].Item1.Add(fd.Discipline);

st[k].Item3.Add(new DefectItem(i.data, wd.Key));
                                    }
                                }
                            }
```

```
                            else
                            {
                                if
((FilterFlow        ==        null)        ||
(FilterFlow(i.data.Flow)))
                                {
                                    for  (int
k = 0; k < fd.SubGroupMask.Count; k++)
                                    {
                                        if
(fd.SubGroupMask[k])
                                        {

st[k].Item2.Add(fd.Discipline);

st[k].Item4.Add(new        DefectItem(i.data,
wd.Key));
                                        }
                                    }
                                }
                            }
                        }

                        int dv = 0;
                        int ddv = 0;
                        for  (int  k  =  0;  k  <
st.Count; k++)
                        {
                            int     ddv_st     =
st[k].Item1.Count + st[k].Item2.Count;
                            bool    isLec_st    =
st[k].Item1.Count > 3;
                            bool   isNonLec_st   =
st[k].Item2.Count > 3;
                            int dv_st = (isLec_st
?  st[k].Item3.Count  :  0)  +  (isNonLec_st  ?
st[k].Item4.Count : 0);
                            dv += dv_st;
                            ddv += ddv_st;
                            if (isLec_st)
                            {

foreach(DefectItem di in st[k].Item3)
                                {

defList.Add(di);
                                }
                            }
                            if (isNonLec_st)
                            {
                                foreach
(DefectItem di in st[k].Item4)
                                {

defList.Add(di);
                                }
                            }
                        }

                        cacheDay = new Tuple<int,
int>(dv, ddv);

s.Value.SetCacheWeekDay(wd.Key,         this,
cacheDay);
                        foreach  (DefectItem  t  in
defList)
                        {

t.Item1.SetDefected(DefectName(s.Key,
t.Item2), true, hard, soft);
                        }
                    }
                    cv += cacheDay.Item1;
                    ccv += cacheDay.Item2;
                }
                ResultData r = new ResultData(cv,
ccv);
                s.Value.SetCacheRestrict(this,
r);
                return r;
            }

            /// <summary>
            /// Функция проверки ограничения
            /// </summary>
            ///                                <param
name="timetable">Расписание</param>
            ///                 <returns>Значение
проверки</returns>
            public   double   Check(TimetableData
timetable, double hard, double soft)
            {
                int c = 0;
                int cc = 0;
                foreach   (GroupTimetable   s   in
timetable.ScheduleByGroups)
                {
                    if   ((Filter   !=   null)   &&
(!Filter(s.Key)))
                    {
                        continue;
                    }
                    ResultData     cacheObj     =
s.Value.GetCacheRestrict<ResultData>(this);
                    if (cacheObj == null)
                    {
                        cacheObj = calc(s, hard,
soft);
                    }
                    c += cacheObj.Item1;
                    cc += cacheObj.Item2;
                }
                if (cc == 0) return 1;
                return 1 - Convert.ToDouble(c) /
cc;
            }

        }
    }
```

Файл Restrict\RestrictEmployee.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Timetable;
using ektu.Timetable.Dict;

namespace ektu.Timetable.Restrict
{

    using ResultData = Tuple<int, int>;
    using          TeacherTimetable          =
KeyValuePair<Teacher, TimetableSubject>;
    using  DefectItem  =  Tuple<TimetableItem,
DayItem>;

    /// <summary>
    /// Класс проверки на накладки по ППС
    /// </summary>
    public class RestrictEmployee : IRestrict
    {
        /// <summary>
        /// Название ограничения
        /// </summary>
        public  string  Name  {  get;  private
set; }

        /// <summary>
        /// Коэффициент для ограничения
        /// </summary>
        public double Factor { get; private
set; }
```

```
        /// <summary>
        /// Фильтр для ограничения
        /// </summary>
        public
FilterRestrictDelegate<Teacher> Filter { get;
set; }

        /// <summary>
        /// Пометка нарушения ограничения
        /// </summary>
        ///                             <param
name="t">Преподаватель</param>
        /// <param name="d">День</param>
        /// <returns></returns>
        private string DefectName(Teacher t,
DayItem d)
        {
            return    Name    +    "="    +
t.TeacherID.ToString()      +      ","      +
d.ID.ToString();
        }

        /// <summary>
        ///    Проверка    ограничения    для
преподавателя
        /// </summary>
        ///     <param     name="s">Расписание
преподавателя</param>
        ///     <param     name="hard">жесткое
ограничение</param>
        ///     <param     name="soft">мягкое
ограничение</param>
        ///   <returns>Значение   ограничения
(число нарушений, всего элементов)</returns>
        private                     ResultData
calc(TeacherTimetable s, double hard, double
soft)
        {
            int ct = 0;
            int cct = 0;
            HashSet<DefectItem>  def  =  new
HashSet<DefectItem>();
            foreach    (KeyValuePair<DayItem,
Dictionary<LectureTime,
List<TimetableItemProxy>>>       wd       in
s.Value.AllData)
            {
                ResultData     cacheDay     =
s.Value.GetCacheWeekDay<ResultData>(wd.Key,
this);
                if (cacheDay == null)
                {
                    int dv = 0;
                    int ddv = 0;
                    foreach
(KeyValuePair<LectureTime,
List<TimetableItemProxy>> lt in wd.Value)
                    {
                        if  (lt.Value.Count(f
=> f.IsBlank) == lt.Value.Count)
                        {
                            continue;
                        }
                        List<TimetableItem> t
= lt.Value.Select(f => f.data).ToList();
                        t.ForEach(f        =>
f.SetDefected(DefectName(s.Key,      wd.Key),
false, hard, soft));
                        if (t.Count > 1)
                        {
                            t.ForEach(
                                f =>
                                {
                                    if
(!f.NotModified)   def.Add(new   DefectItem(f,
wd.Key));
                                });
                                dv  +=  t.Where(fx
=>         !fx.NotModified).Sum(f          =>
f.Flow.Items.Count);
                                ddv += t.Sum(f =>
f.Flow.Items.Count);
                            }
                            if (t.Count == 1)
                            {
                                ddv += 1;
                            }
                        }
                        cacheDay      =      new
ResultData(dv, ddv);

s.Value.SetCacheWeekDay(wd.Key,         this,
cacheDay);
                }
                ct += cacheDay.Item1;
                cct += cacheDay.Item2;
            }
            foreach (DefectItem t in def)
            {

t.Item1.SetDefected(DefectName(s.Key,
t.Item2), true, hard, soft);
            }
            ResultData r = new ResultData(ct,
cct);
            s.Value.SetCacheRestrict(this,
r);
            return r;
        }

        /// <summary>
        /// Функция провекри ограничения
        /// </summary>
        ///                                 <param
name="timetable">Расписание</param>
        ///                 <returns>Результат
проверки</returns>
        public   double   Check(TimetableData
timetable, double hard, double soft)
        {
            int c = 0;
            int cc = 0;
            foreach  (TeacherTimetable  s  in
timetable.ScheduleByTeacher)
            {
                if   ((Filter   !=   null)   &&
(!Filter(s.Key)))
                {
                    continue;
                }
                ResultData     cacheObj     =
s.Value.GetCacheRestrict<ResultData>(this);
                if (cacheObj == null)
                {
                    cacheObj = calc(s, hard,
soft);
                }
                c += cacheObj.Item1;
                cc += cacheObj.Item2;
            }
            if (cc == 0) return 1;
            return 1 - Convert.ToDouble(c) /
cc;
        }

        /// <summary>
        /// Конструктор
        /// </summary>
        ///                                 <param
name="Factor">Коэффициент</param>
        ///    <param    name="Name">Название
ограничения</param>
        public RestrictEmployee(double Factor
=  1,  string  Name  =  "RestrictEmployee"):
base()
        {
```

```
            this.Factor = Factor;
            this.Name = Name;
        }
    }
}
```

Файл Restrict\RestrictEmployeeFill.cs

```
using System;
using System.Threading.Tasks;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using ektu.Timetable.Timetable;
using ektu.Timetable.Dict;
using ektu.Timetable.Flow;

namespace ektu.Timetable.Restrict
{
    using ResultData = Tuple<int, int>;
    using TeacherSchedule = KeyValuePair<Teacher, TimetableSubject>;
    using DefectItem = Tuple<TimetableItem, DayItem>;

    /// <summary>
    /// Проверка на разрывы в расписании преподавателя
    /// </summary>
    public class RestrictEmployeeFill : IRestrict
    {
        /// <summary>
        /// Название ограничения
        /// </summary>
        public string Name { get; private set; }

        /// <summary>
        /// Коэффициент для ограничения
        /// </summary>
        public double Factor { get; private set; }

        /// <summary>
        /// Фильтр по потоку
        /// </summary>
        public FilterRestrictDelegate<FlowData> FilterItem { get; set; }

        /// <summary>
        /// Конструктор
        /// </summary>
        /// <param name="Factor">Коэффициент</param>
        /// <param name="Name">Название ограничения</param>
        public RestrictEmployeeFill(double Factor = 1, string Name = "RestrictEmployeeFill")
        {
            this.Factor = Factor;
            this.Name = Name;
        }

        /// <summary>
        /// Пометка нарушения ограничения в расписании
        /// </summary>
        /// <param name="t">Преподаватель</param>
        /// <param name="d">День</param>
        /// <returns></returns>
        private string DefectName(Teacher t, DayItem d)
        {
            return Name + "=" + t.TeacherID.ToString() + "," + d.ID.ToString();
        }

        /// <summary>
        /// Расчет ограничения по расписанию преподавателя
        /// </summary>
        /// <param name="s">Расписание преподвателя</param>
        /// <param name="hard">Жесткое ограничение</param>
        /// <param name="soft">Мягкое ограничение</param>
        /// <returns>Значение ограничения (число нарушений, всего элементов)</returns>
        private ResultData calc(TeacherSchedule s, double hard, double soft)
        {
            int ct = 0;
            int cct = 0;
            HashSet<DefectItem> def = new HashSet<DefectItem>();
            foreach (KeyValuePair<DayItem, Dictionary<LectureTime, List<TimetableItemProxy>>> v in s.Value.AllData)
            {
                Tuple<int, int> cacheDay = s.Value.GetCacheWeekDay<Tuple<int, int>>(v.Key, this);
                if (cacheDay == null)
                {
                    int dt = 0;
                    Dictionary<int, List<TimetableItem>> lst = new Dictionary<int, List<TimetableItem>>();
                    HashSet<int> nums = new HashSet<int>();
                    foreach (KeyValuePair<LectureTime, List<TimetableItemProxy>> v2 in v.Value)
                    {
                        if (v2.Value.Count(f => f.IsBlank) == v2.Value.Count)
                        {
                            continue;
                        }
                        if (v2.Value.Count(f => (FilterItem == null) || FilterItem(f.data.Flow)) > 0)
                        {

                            lst[v2.Key.NumberLecture] = new List<TimetableItem>();
                            v2.Value.Where(f => (FilterItem == null) || FilterItem(f.data.Flow))
                                    .ToList()
                                    .ForEach(f => {

                                f.data.SetDefected(DefectName(s.Key, v.Key), false, hard, soft);

                                lst[v2.Key.NumberLecture].Add(f.data);

                                nums.Add(v2.Key.NumberLecture);
                                    });
                        }
                    }

                    if (nums.Count > 0)
                    {
                        dt = nums.Max() - nums.Min() + 1 - nums.Count();
                        if (dt > 0)
```

```
                        {
                            List<List<int>>
vs = new List<List<int>>();
                            List<int>   t   =
nums.ToList();
                            t.Sort();
                            List<int>  cur  =
null;
                            for (int i = 0; i
< t.Count; i++)
                            {
                                if ((i == 0)
|| (t[i] != t[i - 1] + 1))
                                {
                                    cur = new
List<int>();

cur.Add(t[i]);

vs.Add(cur);
                                }
                                else
                                {

cur.Add(t[i]);
                                }
                            }
                            vs.Sort((f1,  f2)
=> f2.Count - f1.Count);
                            vs.RemoveAt(0);
                            vs.ForEach(f   =>
f.ForEach(fx  =>  {  if  (lst.ContainsKey(fx))
lst[fx].ForEach(fy       =>       def.Add(new
DefectItem(fy, v.Key))); }));
                        }
                    }
                    cacheDay       =       new
ResultData(dt, nums.Count);

s.Value.SetCacheWeekDay(v.Key,          this,
cacheDay);
                }
                ct += cacheDay.Item1;
                cct += cacheDay.Item2;
            }
            foreach (DefectItem t in def)
            {

t.Item1.SetDefected(DefectName(s.Key,
t.Item2), true, hard, soft);
            }
            ResultData r = new ResultData(ct,
cct);
            s.Value.SetCacheRestrict(this,
r);
            return r;
        }

        /// <summary>
        /// Функция проверки ограничения
        /// </summary>
        ///                                <param
name="timetable">Расписание</param>
        ///                 <returns>Значение
проверки</returns>
        public   double   Check(TimetableData
timetable, double hard, double soft)
        {
            int c = 0;
            int cc = 0;
            foreach   (TeacherSchedule   s   in
timetable.ScheduleByTeacher)
            {
                ResultData     cacheObj     =
s.Value.GetCacheRestrict<ResultData>(this);
                if (cacheObj == null)
                {
                    cacheObj = calc(s, hard,
soft);
                }
                c    +=    cacheObj.Item1    >
cacheObj.Item2     ?     cacheObj.Item2     :
cacheObj.Item1;
                cc += cacheObj.Item2;
            }
            if (cc == 0)
            {
                return 1;
            }
            return 1 - Convert.ToDouble(c) /
cc;
        }
    }
}
```

Файл Restrict\RestrictEmployeeHours.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Timetable;
using ektu.Timetable.Flow;
using ektu.Timetable.Dict;

namespace ektu.Timetable.Restrict
{
    /// <summary>
    /// Проверка ограничения на число занятий
для группы в течении дня
    /// </summary>
    public   class   RestrictEmployeeHours   :
IRestrict
    {
        public  string  Name  {  get;  private
set; }
        /// <summary>
        ///  Максимальное  число  часов  для
группы в течении дня
        /// </summary>
        private int mMaxHours;
        /// <summary>
        ///  Минимальное  число  часов  для
группы в течении дня
        /// </summary>
        private int mMinHours;
        /// <summary>
        /// Коэффициент для ограничения
        /// </summary>
        public double Factor { get; private
set; }
        public
FilterRestrictDelegate<Teacher> Filter { get;
set; }
        private        Tuple<int,        int>
calc(KeyValuePair<Teacher,  TimetableSubject>
s, double hard, double soft)
        {
            int ct = 0;
            int cct = 0;
            HashSet<TimetableItem> def = new
HashSet<TimetableItem>();
            HashSet<TimetableItem> ndef = new
HashSet<TimetableItem>();
            foreach    (KeyValuePair<DayItem,
Dictionary<LectureTime,
List<TimetableItemProxy>>>       wd       in
s.Value.AllData)
            {
                Tuple<int,  int>  cacheDay  =
s.Value.GetCacheWeekDay<Tuple<int,
int>>(wd.Key, this);
                if (cacheDay == null)
                {
                    int n = 0;
                    int dt = 0;
                    int ddt = 0;
```

```
                    foreach
(KeyValuePair<LectureTime,
List<TimetableItemProxy>> lt in wd.Value)
                    {
                        if  (lt.Value.Count(f
=> !f.IsBlank) != 0)
                        {
                            n++;
                        }
                    }
                    if (n > 0)
                    {
                        ddt++;
                        if  ((n  >  mMaxHours)
|| (n < mMinHours))
                        {
                            dt++;
                        }
                        foreach
(KeyValuePair<LectureTime,
List<TimetableItemProxy>> lt in wd.Value)
                        {
                            foreach
(TimetableItemProxy i in lt.Value)
                            {
                                if    ((n    >
mMaxHours) || (n < mMinHours))
                                {
                                    if
(!i.IsBlank)
                                    {

def.Add(i.data);
                                    }
                                }
                                else
                                {
                                    if
(!i.IsBlank)
                                    {

ndef.Add(i.data);
                                    }
                                }
                            }
                        }
                    }
                    cacheDay = new Tuple<int,
int>(dt, ddt);

s.Value.SetCacheWeekDay(wd.Key,         this,
cacheDay);
                }
                ct += cacheDay.Item1;
                cct += cacheDay.Item2;
            }
            foreach (TimetableItem i in def)
            {
                i.SetDefected(Name  +  "="  +
s.Key.TeacherID.ToString(),    true,    hard,
soft);
            }
            ndef.RemoveWhere(f             =>
def.Contains(f));
            foreach (TimetableItem i in ndef)
            {
                i.SetDefected(Name  +  "="  +
s.Key.TeacherID.ToString(),    false,    hard,
soft);
            }
            Tuple<int,    int>    r    =    new
Tuple<int, int>(ct, cct);
            s.Value.SetCacheRestrict(this,
r);
            return r;
        }
        /// <summary>
        /// Функция провекри ограничения
        /// </summary>
        ///                               <param
name="timetable">Расписание</param>
        ///                <returns>Результат
проверки</returns>
        public   double   Check(TimetableData
timetable, double hard, double soft)
        {
            int c = 0;
            int cc = 0;
            foreach    (KeyValuePair<Teacher,
TimetableSubject>            s            in
timetable.ScheduleByTeacher)
            {
                if   ((Filter   !=   null)   &&
(!Filter(s.Key)))
                {
                    continue;
                }
                Tuple<int,  int>  cacheObj  =
s.Value.GetCacheRestrict<Tuple<int,
int>>(this);
                if (cacheObj == null)
                {
                    cacheObj = calc(s, hard,
soft);
                }
                c += cacheObj.Item1;
                cc += cacheObj.Item2;
            }
            if (cc == 0) return 1;
            return 1 - Convert.ToDouble(c) /
cc;
        }
        /// <summary>
        /// Конструктор
        /// </summary>
        ///                                <param
name="Factor">Коэффициент</param>
        ///                                <param
name="maxHours">Максимальное  число  часов  в
день</param>
        public   RestrictEmployeeHours(double
Factor     =     1,     string     Name     =
"RestrictEmployeeHours",  int  maxHours  =  8,
int minHours = 0)
        {
            this.Factor = Factor;
            this.Name = Name;
            mMaxHours = maxHours;
            mMinHours = minHours;
        }

        public  override  bool  Equals(object
obj)
        {
            return (obj != null) && (obj ==
this);
        }

        public override int GetHashCode()
        {
            return Name.GetHashCode();
        }
    }
}
```

Файл Restrict\RestrictGroup.cs

```
using System;
using System.Collections.Generic;
using System.Collections;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Timetable;
using ektu.Timetable.Dict;
using ektu.Timetable.Flow;

namespace ektu.Timetable.Restrict
```

```
{

    using ResultData = Tuple<int, int>;
    using GroupTimetable = KeyValuePair<Group, TimetableSubject>;
    using DefectItem = Tuple<TimetableItem, DayItem>;

    /// <summary>
    /// Класс для проверки накладок для группы
    /// </summary>
    public class RestrictGroup : IRestrict
    {
        /// <summary>
        /// Название ограничения
        /// </summary>
        public string Name { get; private set; }

        /// <summary>
        /// Коэффициент для ограничения
        /// </summary>
        public double Factor { get; private set; }

        /// <summary>
        /// Пометка нарушения ограничения
        /// </summary>
        /// <param name="g">Группа</param>
        /// <param name="d">День</param>
        /// <returns></returns>
        private string DefectName(Group g, DayItem d)
        {
            return Name + "=" + g.GroupID.ToString() + "," + d.ID.ToString();
        }

        /// <summary>
        /// Проверка ограничения для группы
        /// </summary>
        /// <param name="s">Расписание группы</param>
        /// <param name="hard">жесткое ограничение</param>
        /// <param name="soft">мягкое ограничение</param>
        /// <returns>Значение ограничения (число нарушений, всего элементов)</returns>
        private Tuple<int, int> calc(GroupTimetable s, double hard, double soft)
        {
            int ct = 0;
            int cct = 0;
            HashSet<DefectItem> def = new HashSet<DefectItem>();
            HashSet<TimetableItem> ci = new HashSet<TimetableItem>();
            foreach (KeyValuePair<DayItem, Dictionary<LectureTime, List<TimetableItemProxy>>> wd in s.Value.AllData)
            {
                foreach (KeyValuePair<LectureTime, List<TimetableItemProxy>> lt in wd.Value)
                {
                    if (lt.Value.Count(f => f.IsBlank) == lt.Value.Count)
                    {
                        continue;
                    }

                    List<TimetableItem> d = lt.Value.Select(f => f.data).ToList();
                    d.ForEach(
                        f =>
                        {
                            f.SetDefected(DefectName(s.Key, wd.Key), false, hard, soft);
                            ci.Add(f);
                        });

                    if (d.Count <= 1)
                    {
                        continue;
                    }

                    BitArray mask = null;
                    int[] arr_check = null;

                    foreach (TimetableItem l in d)
                    {
                        FlowItem fi = l.Flow.Items.Where(f => f.Group == s.Key).First();
                        if (mask == null)
                        {
                            mask = new BitArray(fi.SubGroupMask);
                            arr_check = new int[Convert.ToInt32(Math.Ceiling(mask.Length / 32.0f))];
                        }
                        else
                        {
                            if (d[0].GetPar.Contains(l))
                            {
                                continue;
                            }
                            BitArray arr = new BitArray(mask);

                            arr.And(fi.SubGroupMask);

                            arr.CopyTo(arr_check, 0);
                            if (arr_check.Sum() > 0)
                            {
                                d.ForEach(f => def.Add(new DefectItem(f, wd.Key)));
                                break;
                            }

                            mask.Or(fi.SubGroupMask);
                        }
                    }
                }
            }
            foreach (DefectItem i in def)
            {

                i.Item1.SetDefected(DefectName(s.Key, i.Item2), true, hard, soft);
            }
            ct = def.Count;
            cct = ci.Count;
            ResultData r = new ResultData(ct, cct);
            s.Value.SetCacheRestrict(this, r);
            return r;
        }

        /// <summary>
        /// Функция провекри ограничения
        /// </summary>
        /// <param name="timetable">Расписание</param>
        /// <returns>Результат проверки</returns>
        public double Check(TimetableData timetable, double hard, double soft)
```

```
        {
            int c = 0;
            int cc = 0;
            foreach (GroupTimetable s in
timetable.ScheduleByGroups)
            {
                ResultData cacheObj =
s.Value.GetCacheRestrict<ResultData>(this);
                if (cacheObj == null)
                {
                    cacheObj = calc(s, hard,
soft);
                }
                c += cacheObj.Item1;
                cc += cacheObj.Item2;
            }
            if (cc == 0) return 1;
            return 1 - Convert.ToDouble(c) /
cc;
        }

        /// <summary>
        /// Конструктор
        /// </summary>
        ///                             <param
name="Factor">Коэффициент</param>
        ///    <param    name="Name">Название
ограничения</param>
        public RestrictGroup(double Factor =
1, string Name = "RestrictGroup")
        {
            this.Factor = Factor;
            this.Name = Name;
        }
    }
}
```

Файл Restrict\RestrictGroupFill.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Timetable;
using ektu.Timetable.Flow;
using ektu.Timetable.Dict;

namespace ektu.Timetable.Restrict
{
    using ResultData = Tuple<int, int>;
    using          GroupTimetable          =
KeyValuePair<Group, TimetableSubject>;
    using DefectItem = Tuple<TimetableItem,
DayItem>;

    /// <summary>
    /// Проверка на разрывы в расписании
группы
    /// </summary>
    public    class    RestrictGroupFill    :
IRestrict
    {
        /// <summary>
        /// Коэффициент
        /// </summary>
        public double Factor { get; private
set; }

        /// <summary>
        /// Название ограничения
        /// </summary>
        public string Name { get; private
set; }

        /// <summary>
        /// Конструктор
        /// </summary>
        ///                             <param
name="Factor">Коэффициент</param>
        ///    <param    name="Name">Название
ограничения</param>
        public       RestrictGroupFill(double
Factor     =     1,     string     Name     =
"RestrictGroupFill")
        {
            this.Factor = Factor;
            this.Name = Name;
        }

        /// <summary>
        /// Фильтр по группе
        /// </summary>
        public  FilterRestrictDelegate<Group>
Filter { get; set; }

        /// <summary>
        /// Фильтр по потокам
        /// </summary>
        public
FilterRestrictDelegate<FlowData> FilterItem {
get; set; }

        /// <summary>
        /// Пометка нарушения ограничения
        /// </summary>
        /// <param name="g">Группа</param>
        /// <param name="d">День</param>
        /// <returns></returns>
        private  string  DefectName(Group  g,
DayItem d)
        {
            return     Name     +     "="     +
g.GroupID.ToString() + "," + d.ID.ToString();
        }

        /// <summary>
        /// Вычисление ограничения по группе
        /// </summary>
        ///     <param     name="s">Расписание
группы</param>
        ///     <param     name="hard">жесткое
ограничение</param>
        ///     <param     name="soft">мягкое
ограничение</param>
        ///   <returns>Значение   ограничения
(число нарушений, всего элементов)</returns>
        private                    ResultData
calc(GroupTimetable  s,  double  hard,  double
soft)
        {
            int ct = 0;
            int cct = 0;
            HashSet<DefectItem>  def  =  new
HashSet<DefectItem>();
            foreach    (KeyValuePair<DayItem,
Dictionary<LectureTime,
List<TimetableItemProxy>>>        v        in
s.Value.AllData)
            {
                ResultData     cacheDay     =
s.Value.GetCacheWeekDay<ResultData>(v.Key,
this);
                if (cacheDay == null)
                {
                    int dt = 0;
                    int dtt = 0;
                    List<HashSet<int>>  st  =
new List<HashSet<int>>();
                    HashSet<TimetableItem>
items = new HashSet<TimetableItem>();

                    foreach
(KeyValuePair<LectureTime,
List<TimetableItemProxy>> v2 in v.Value)
                    {
```

```
                        if  (v2.Value.Count(f
=> f.IsBlank) == v2.Value.Count)
                        {
                            continue;
                        }
                        if  (v2.Value.Count(f
=>      (FilterItem      ==      null)      ||
FilterItem(f.data.Flow)) > 0)
                        {
                            v2.Value.Where(f
=>      (FilterItem      ==      null)      ||
FilterItem(f.data.Flow))
                                .ToList()
                                .ForEach(f =>
{

f.data.SetDefected(DefectName(s.Key,  v.Key),
false, hard, soft);

f.data.Flow.Items.ForEach(
                                            ff =>
                                            {

if (ff.Group == s.Key)
                                                {

if (st.Count == 0)

{

for (int k = 0; k < ff.SubGroupMask.Count;
k++)

{

st.Add(new HashSet<int>());

}

}

for(int  k  =  0;  k  <  ff.SubGroupMask.Count;
k++)

{

st[k].Add(v2.Key.NumberLecture);

items.Add(f.data);

}
                                                }
                                            });
                                });
                        }
                    }

                    if (st.Count > 0)
                    {
                        foreach(HashSet<int>
stl in st)
                        {
                            int    min_l    =
stl.Min();
                            int    max_l    =
stl.Max();
                            int rs = max_l -
min_l + 1 - stl.Count;
                            dtt += stl.Count;
                            dt += rs;
                            if (rs > 0)
                            {

foreach(TimetableItem item in items)
                                {

def.Add(new DefectItem(item, v.Key));
                                }
                            }
                        }
                    }
                    cacheDay      =      new
ResultData(dt, dtt);

s.Value.SetCacheWeekDay(v.Key,          this,
cacheDay);
                }
                ct += cacheDay.Item1;
                cct += cacheDay.Item2;
            }
            foreach(DefectItem t in def)
            {

t.Item1.SetDefected(DefectName(s.Key,
t.Item2), true, hard, soft);
            }
            ResultData r = new ResultData(ct,
cct);
            s.Value.SetCacheRestrict(this,
r);
            return r;
        }

        /// <summary>
        /// Функция проверки ограничения
        /// </summary>
        ///                                <param
name="timetable">Расписание</param>
        ///                 <returns>Значение
проверки</returns>
        public   double   Check(TimetableData
timetable, double hard, double soft)
        {
            int c = 0;
            int cc = 0;
            foreach   (GroupTimetable   s   in
timetable.ScheduleByGroups)
            {
                if   ((Filter   !=   null)   &&
(!Filter(s.Key)))
                {
                    continue;
                }
                ResultData     cacheObj     =
s.Value.GetCacheRestrict<ResultData>(this);
                if (cacheObj == null)
                {
                    cacheObj = calc(s, hard,
soft);
                }

                c    +=    cacheObj.Item1    >
cacheObj.Item2     ?     cacheObj.Item2     :
cacheObj.Item1;
                cc += cacheObj.Item2;
            }
            if (cc == 0) return 1;
            return 1 - Convert.ToDouble(c) /
cc;
        }
    }
}
```

Файл Restrict\RestrictGroupHours.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Timetable;
using ektu.Timetable.Flow;
using ektu.Timetable.Dict;

namespace ektu.Timetable.Restrict
{
    using ResultData = Tuple<int, int>;
```

```
    using          GroupTimetable          =
KeyValuePair<Group, TimetableSubject>;
    using  DefectItem  =  Tuple<TimetableItem,
DayItem>;

    /// <summary>
    /// Проверка ограничения на число занятий
для группы в течении дня
    /// </summary>
    public    class    RestrictGroupHours    :
IRestrict
    {
        /// <summary>
        /// Название ограничения
        /// </summary>
        public  string  Name  {  get;  private
set; }

        /// <summary>
        ///  Максимальное  число  часов  для
группы в течении дня
        /// </summary>
        private int MaxHours;

        /// <summary>
        ///  Минимальное  число  часов  для
группы в течении дня
        /// </summary>
        private int MinHours;

        /// <summary>
        /// Коэффициент для ограничения
        /// </summary>
        public double Factor { get; private
set; }

        /// <summary>
        /// Фильтр по группе
        /// </summary>
        public  FilterRestrictDelegate<Group>
Filter { get; set; }

        /// <summary>
        /// Фильтр по потокам
        /// </summary>
        public
FilterRestrictDelegate<FlowData> FilterItem {
get; set; }

        /// <summary>
        /// Пометка нарушения ограничения
        /// </summary>
        /// <param name="g">Группа</param>
        /// <param name="d">День</param>
        /// <returns></returns>
        private  string  DefectName(Group  g,
DayItem d)
        {
            return    Name    +    "="    +
g.GroupID.ToString() + "," + d.ID.ToString();
        }

        /// <summary>
        /// Расчет ограничения по расписанию
преподавателя
        /// </summary>
        ///    <param    name="s">Расписание
преподвателя</param>
        ///    <param    name="hard">Жесткое
ограничение</param>
        ///     <param     name="soft">Мягкое
ограничение</param>
        ///   <returns>Значение   ограничения
(число нарушений, всего элементов)</returns>
        private                    ResultData
calc(GroupTimetable  s,  double  hard,  double
soft)
        {
            int ct = 0;
            int cct = 0;
            HashSet<DefectItem>  def  =  new
HashSet<DefectItem>();
            foreach    (KeyValuePair<DayItem,
Dictionary<LectureTime,
List<TimetableItemProxy>>>       wd       in
s.Value.AllData)
            {
                Tuple<int,  int>  cacheDay  =
s.Value.GetCacheWeekDay<Tuple<int,
int>>(wd.Key, this);
                if (cacheDay == null)
                {
                    int dt = 0;
                    int ddt = 0;


                    int[] sums = null;

                    foreach
(KeyValuePair<LectureTime,
List<TimetableItemProxy>> lt in wd.Value)
                    {
                        BitArray mask = null;
                        if  (lt.Value.Count(f
=> (!f.IsBlank) &&

((FilterItem        ==        null)        ||
(FilterItem(f.data.Flow)))) != 0)
                        {
                            FlowItem   fi   =
lt.Value[0].data.Flow.Items.Where(f        =>
f.Group == s.Key).First();
                            if (sums == null)
                            {
                                sums   =   new
int[fi.SubGroupMask.Length];
                            }
                        }

                        lt.Value.ForEach(
                            f =>
                            {

f.data.SetDefected(DefectName(s.Key, wd.Key),
false, hard, soft);
                                FlowItem fi =
f.data.Flow.Items.Where(fg   =>   fg.Group   ==
s.Key).First();
                                if  (mask  ==
null)
                                {
                                    mask    =
new BitArray(fi.SubGroupMask);
                                }
                                else
                                {

mask.Or(fi.SubGroupMask);
                                }
                            });
                        if (mask != null)
                        {
                            for (int i = 0; i
< sums.Length; i++)
                            {
                                sums[i]    +=
mask[i] ? 1 : 0;
                            }
                        }
                    }
                    if (sums != null)
                    {
                        for  (int  i  =  0;  i  <
sums.Length; i++)
                        {
                            if (sums[i] > 0)
                            {
```

```
                                ddt        +=
sums[i];
                                if  ((sums[i]
> MaxHours) || (sums[i] < MinHours))
                                {
                                    dt     +=
sums[i];
                                    foreach
(KeyValuePair<LectureTime,
List<TimetableItemProxy>> lt in wd.Value)
                                    {

foreach     (TimetableItemProxy     it     in
lt.Value.Where(f => !f.IsBlank))
                                        {

def.Add(new DefectItem(it.data, wd.Key));
                                        }
                                    }
                                }
                            }
                        }
                        cacheDay       =       new
ResultData(dt, ddt);

s.Value.SetCacheWeekDay(wd.Key,         this,
cacheDay);
                }
                ct += cacheDay.Item1;
                cct += cacheDay.Item2;
            }
            foreach(DefectItem i in def)
            {

i.Item1.SetDefected(DefectName(s.Key,
i.Item2), true, hard, soft);
            }
            ResultData r = new ResultData(ct,
cct);
            s.Value.SetCacheRestrict(this,
r);
            return r;
        }

        /// <summary>
        /// Функция провекри ограничения
        /// </summary>
        ///                                <param
name="timetable">Расписание</param>
        ///                <returns>Результат
проверки</returns>
        public   double   Check(TimetableData
timetable, double hard, double soft)
        {
            int c = 0;
            int cc = 0;
            foreach   (GroupTimetable   s   in
timetable.ScheduleByGroups)
            {
                if   ((Filter   !=   null)   &&
(!Filter(s.Key)))
                {
                    continue;
                }
                ResultData     cacheObj     =
s.Value.GetCacheRestrict<ResultData>(this);
                if (cacheObj == null)
                {
                    cacheObj = calc(s, hard,
soft);
                }
                c += cacheObj.Item1;
                cc += cacheObj.Item2;
            }
            if (cc == 0) return 1;
            return 1 - Convert.ToDouble(c) /
cc;
        }
        /// <summary>
        /// Конструктор
        /// </summary>
        ///                                <param
name="Factor">Коэффициент</param>
        ///                                <param
name="maxHours">Максимальное  число  часов  в
день</param>
        ///                                <param
name="minHours">Минимальное  число  часов  в
день</param>
        ///    <param    name="Name">Название
ограничения</param>
        public      RestrictGroupHours(double
Factor = 1, int maxHours = 8, int minHours =
2, string Name = "RestrictGroupHours")
        {
            this.Factor = Factor;
            this.Name = Name;
            MaxHours = maxHours;
            MinHours = minHours;
        }
    }
}
```

Файл Restrict\RestrictOneDayDisciplines.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Timetable;
using ektu.Timetable.Dict;
using ektu.Timetable.Flow;

namespace ektu.Timetable.Restrict
{

    using ResultData = Tuple<int, int>;
    using          GroupTimetable          =
KeyValuePair<Group, TimetableSubject>;
    using TypeID = Tuple<Discipline, DayItem,
TypeFlowBase, int>;
    using SetType = HashSet<TimetableItem>;

    /// <summary>
    /// Ограничение на повторение дисциплины
в течении дня
    /// </summary>
    public class RestrictOneDayDisciplines :
IRestrict
    {
        /// <summary>
        /// Коэффициент для ограничения
        /// </summary>
        public double Factor { get; private
set; }

        /// <summary>
        /// Название ограничения
        /// </summary>
        public  string  Name  {  get;  private
set; }

        /// <summary>
        /// Конструктор
        /// </summary>
        /// <param name="Factor">Коэффициент
для ограничения</param>
        ///    <param    name="Name">Название
ограничения</param>
        public
RestrictOneDayDisciplines(double Factor = 1,
string Name = "RestrictOneDayDisciplines")
        {
            this.Factor = Factor;
            this.Name = Name;
```

```
        }

        /// <summary>
        /// Фильтр для ограничения группы
        /// </summary>
        public  FilterRestrictDelegate<Group>
Filter { get; set; }

        /// <summary>
        /// Фильтр для ограничения потока
        /// </summary>
        public
FilterRestrictDelegate<FlowData> FilterFlow {
get; set; }

        /// <summary>
        /// Пометка нарушения ограничения
        /// </summary>
        /// <param name="g">Группа</param>
        /// <param name="d">День</param>
        /// <returns></returns>
        private  string  DefectName(Group  g,
DayItem d)
        {
            return     Name     +     "="     +
g.GroupID.ToString() + "," + d.ID.ToString();
        }

        /// <summary>
        /// Проверка ограничения для группы
        /// </summary>
        ///     <param     name="s">Расписание
группы</param>
        ///     <param     name="hard">жесткое
ограничение</param>
        ///     <param     name="soft">мягкое
ограничение</param>
        ///   <returns>Значение   ограничения
(число нарушений, всего элементов)</returns>
        private                      ResultData
calc(GroupTimetable  s,  double  hard,  double
soft)
        {
            int ct = 0;
            int cct = 0;
            Dictionary<TypeID, SetType> disc
= new Dictionary<TypeID, SetType>();
            foreach    (KeyValuePair<DayItem,
Dictionary<LectureTime,
List<TimetableItemProxy>>>       wd       in
s.Value.AllData)
            {
                foreach
(KeyValuePair<LectureTime,
List<TimetableItemProxy>> lt in wd.Value)
                {
                    foreach
(TimetableItemProxy i in lt.Value)
                    {
                        if
(i.data.NotModified) continue;
                        if        (i.IsBlank)
continue;
                        if   ((FilterFlow   !=
null)      &&      (!FilterFlow(i.data.Flow)))
continue;

                        FlowItem  flowItem  =
i.data.Flow.Items.First(f   =>   f.Group   ==
s.Key);
                        TypeID  tuple  =  new
TypeID(flowItem.Discipline,           wd.Key,
i.data.Flow.TypeFlow, flowItem.SubGroup);
                        if
(!disc.ContainsKey(tuple))
                        {
                            disc.Add(tuple,
new SetType());
                        }
disc[tuple].Add(i.data);
                    }
                }
            }
            foreach     (KeyValuePair<TypeID,
SetType> i in disc)
            {
                cct += i.Value.Count;
                bool        isDefect        =
(i.Value.Count > 1) /*&&

(i.Value.First().Flow.DayTimes.GroupBy(f   =>
f.Day).Count() > 1)*/;
                if (isDefect)
                {
                    ct += i.Value.Count;
                }
                foreach (TimetableItem it in
i.Value)
                {

it.SetDefected(DefectName(s.Key,
i.Key.Item2), isDefect, hard, soft);
                }
            }
            ResultData r = new ResultData(ct,
cct);
            s.Value.SetCacheRestrict(this,
r);
            return r;
        }

        /// <summary>
        /// Функция проверки ограничения
        /// </summary>
        ///                                <param
name="timetable">Расписание</param>
        ///                 <returns>Значение
проверки</returns>
        public   double   Check(TimetableData
timetable, double hard, double soft)
        {
            int c = 0;
            int cc = 0;
            foreach   (GroupTimetable   s   in
timetable.ScheduleByGroups)
            {
                if   ((Filter   !=   null)   &&
(!Filter(s.Key)))
                {
                    continue;
                }
                ResultData     cacheObj     =
s.Value.GetCacheRestrict<ResultData>(this);
                if (cacheObj == null)
                {
                    cacheObj = calc(s, hard,
soft);
                }
                c += cacheObj.Item1;
                cc += cacheObj.Item2;
            }
            if (cc == 0) return 1;
            return 1 - Convert.ToDouble(c) /
cc;
        }
    }
}
```

Файл Restrict\RestrictRoom.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Timetable;
using ektu.Timetable.Dict;
```

```
using System.Diagnostics;

namespace ektu.Timetable.Restrict
{
    using ResultData = Tuple<int, int>;
    using RoomTimetable = KeyValuePair<Room,
TimetableSubject>;
    using DefectItem = Tuple<TimetableItem,
DayItem>;

    /// <summary>
    /// Ограничение для аудитории
    /// </summary>
    public class RestrictRoom : IRestrict
    {
        /// <summary>
        /// Коэффициент
        /// </summary>
        public double Factor { get; private
set; }

        /// <summary>
        /// Название ограничения
        /// </summary>
        public string Name { get; private
set; }

        /// <summary>
        /// Фильтр для ограничения
        /// </summary>
        public FilterRestrictDelegate<Room>
Filter { get; set; }

        /// <summary>
        /// Пометка нарушения ограничения
        /// </summary>
        /// <param name="r">Аудитория</param>
        /// <param name="d">День</param>
        /// <returns></returns>
        private string DefectName(Room r,
DayItem d)
        {
            return Name + "=" +
r.RoomID.ToString() + "," + d.ID.ToString();
        }

        /// <summary>
        /// Вычисление ограничения по
аудитории (только 1 занятие)
        /// </summary>
        /// <param name="s">Расписание
аудитории</param>
        /// <param name="hard">жесткое
ограничение</param>
        /// <param name="soft">мягкое
ограничение</param>
        /// <returns>Значение ограничения
(число нарушений, всего элементов)</returns>
        private Tuple<int, int>
calcOneLesson(RoomTimetable s, double hard,
double soft)
        {
            int ct = 0;
            int cct = 0;

            HashSet<DefectItem> def = new
HashSet<DefectItem>();
            foreach (KeyValuePair<DayItem,
Dictionary<LectureTime,
List<TimetableItemProxy>>> v in
s.Value.AllData)
            {
                Tuple<int, int> cacheDay =
s.Value.GetCacheWeekDay<Tuple<int,
int>>(v.Key, this);
                if (cacheDay == null)
                {
                    int dt = 0;
                    int ddt = 0;
                    foreach
(KeyValuePair<LectureTime,
List<TimetableItemProxy>> v2 in v.Value)
                    {
                        if (v2.Value.Count(f
=> f.IsBlank) == v2.Value.Count)
                        {
                            continue;
                        }
                        v2.Value.ForEach(f =>
f.data.SetDefected(DefectName(s.Key, v.Key),
false, hard, soft));

                        if (v2.Value.Count >
1)
                        {
                            dt             +=
v2.Value.Where(fx                          =>
!fx.data.NotModified).Sum(f                =>
f.data.Flow.Items.Sum(fx                   =>
fx.CountStudent));
                            foreach
(TimetableItemProxy i in v2.Value)
                            {
                                if
(!i.data.NotModified)             def.Add(new
DefectItem(i.data, v.Key));
                            }
                        }
                        ddt                +=
v2.Value.Where(fx                          =>
!fx.data.NotModified).Sum(f                =>
f.data.Flow.Items.Sum(fx                   =>
fx.CountStudent));
                    }
                    cacheDay       =       new
ResultData(dt, ddt);

s.Value.SetCacheWeekDay(v.Key,          this,
cacheDay);
                }
                ct += cacheDay.Item1;
                cct += cacheDay.Item2;
            }
            foreach (DefectItem i in def)
            {

i.Item1.SetDefected(DefectName(s.Key,
i.Item2), true, hard, soft);
            }
            ResultData r = new ResultData(ct,
cct);
            s.Value.SetCacheRestrict(this,
r);
            return r;
        }

        /// <summary>
        /// Вычисление ограничения по
аудитории (несколько занятий одновременное с
заданной вместимостью)
        /// </summary>
        /// <param name="s">Расписание
аудитории</param>
        /// <param name="hard">жесткое
ограничение</param>
        /// <param name="soft">мягкое
ограничение</param>
        /// <returns>Значение ограничения
(число нарушений, всего элементов)</returns>
        private Tuple<int, int>
calcPlaceLesson(RoomTimetable s, double hard,
double soft)
        {
            int ct = 0;
            int cct = 0;
            HashSet<DefectItem> def = new
HashSet<DefectItem>();
```

```
            foreach    (KeyValuePair<DayItem,
Dictionary<LectureTime,
List<TimetableItemProxy>>>        v        in
s.Value.AllData)
            {
                ResultData     cacheDay     =
s.Value.GetCacheWeekDay<ResultData>(v.Key,
this);
                if (cacheDay == null)
                {
                    int dt = 0;
                    int ddt = 0;
                    foreach
(KeyValuePair<LectureTime,
List<TimetableItemProxy>> pair in v.Value)
                    {
                        if
(pair.Value.Count(f    =>    f.IsBlank)    ==
pair.Value.Count)
                        {
                            continue;
                        }
                        pair.Value.ForEach(f
=>       f.data.SetDefected(DefectName(s.Key,
v.Key), false, hard, soft));

                        int       ss       =
pair.Value.Sum(f   =>   f.IsBlank   ?   0   :
f.data.Flow.CountStudent);
                        if (ss > 0)
                        {
                            ddt            +=
pair.Value.Count;
                            if      (ss      >
s.Key.Places)
                            {
                                dt         +=
pair.Value.Count;

pair.Value.ForEach(f      =>      def.Add(new
DefectItem(f.data, v.Key)));
                            }
                        }
                    }
                    cacheDay       =       new
ResultData(dt, ddt);

s.Value.SetCacheWeekDay(v.Key,          this,
cacheDay);
                }
                ct += cacheDay.Item1;
                cct += cacheDay.Item2;
            }
            foreach (DefectItem i in def)
            {

i.Item1.SetDefected(DefectName(s.Key,
i.Item2), true, hard, soft);
            }
            ResultData r = new ResultData(ct,
cct);
            s.Value.SetCacheRestrict(this,
r);
            return r;
        }

        /// <summary>
        /// Функция провекри ограничения
        /// </summary>
        ///                                <param
name="timetable">Расписание</param>
        ///                <returns>Результат
проверки</returns>
        public   double   Check(TimetableData
timetable, double hard, double soft)
        {
            int c = 0;
            int cc = 0;
            foreach   (RoomTimetable   s   in
timetable.ScheduleByRoom)
            {
                if   ((Filter   !=   null)   &&
(!Filter(s.Key)))
                {
                    continue;
                }
                ResultData     cacheObj     =
s.Value.GetCacheRestrict<ResultData>(this);
                if (cacheObj == null)
                {
                    if
(s.Key.HasAttribute("PlaceCheck"))
                    {
                        cacheObj            =
calcPlaceLesson(s, hard, soft);
                    }
                    else
                    {
                        cacheObj            =
calcOneLesson(s, hard, soft);
                    }
                }
                c += cacheObj.Item1;
                cc += cacheObj.Item2;
            }
            if (cc == 0) return 1;
            return 1 - Convert.ToDouble(c) /
cc;
        }

        /// <summary>
        /// Конструктор
        /// </summary>
        ///                              <param
name="Factor">Коэффициент</param>
        ///    <param    name="Name">Название
ограничения</param>
        public  RestrictRoom(double  Factor  =
1, string Name = "RestrictRoom")
        {
            this.Factor = Factor;
            this.Name = Name;
        }
    }
}
```

Файл Timetable\Excluder.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Dict;
using ektu.Timetable.Timetable;

namespace ektu.Timetable.Timetable
{
    /// <summary>
    ///  Исключение  возможных  значений  по
потоку
    /// </summary>
    public class Excluder
    {
        /// <summary>
        /// Проверка значений на исключение
        /// </summary>
        /// <param name="flow">Поток</param>
        ///                              <param
name="room">Аудитория</param>
        ///      <param      name="day">День
недели</param>
        ///      <param      name="lt">Время
занятий</param>
        ///   <returns>Исключено  занятие  в
данное  время  и  день  в  аудитории  или
нет</returns>
```

```
        public          virtual          bool
Exclude(TimetableItem    flow,    Room    room,
DayItem day, LectureTime lt)
        {
            return false;
        }
    }
}
```

Файл Timetable\TimetableData.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml;
using ektu.Timetable.Restrict;
using ektu.Timetable.Dict;
using ektu.Timetable.Flow;
using System.Collections;

namespace ektu.Timetable.Timetable
{
    /// <summary>
    /// Тип получения расписания
    /// </summary>
    public enum TypeGenTimetable
    {
        /// <summary>
        /// Начальное расписание
        /// </summary>
        Initial,
        /// <summary>
        /// Мутация
        /// </summary>
        Mutating,
        /// <summary>
        /// Скрещивание
        /// </summary>
        Crossing
    }
    /// <summary>
    /// Расписание
    /// </summary>
    public class TimetableData : ICloneable,
IEnumerable<TimetableItem>
    {
        private          Dictionary<FlowData,
TimetableItem>      flowItems      =      new
Dictionary<FlowData, TimetableItem>();
        /// <summary>
        /// Расписание по группам
        /// </summary>
        private             Dictionary<Group,
TimetableSubject> mGroupTimetable;
        /// <summary>
        /// Расписания по преподвателям
        /// </summary>
        private           Dictionary<Teacher,
TimetableSubject> mTeacherTimetable;
        /// <summary>
        /// Расписания по аудиториям
        /// </summary>
        private              Dictionary<Room,
TimetableSubject> mRoomTimetable;
        /// <summary>
        /// Элементы расписания
        /// </summary>
        private List<TimetableItem> mItems;
        /// <summary>
        /// Генератор
        /// </summary>
        internal Generator mGenerator;
        /// <summary>
        /// Тип получения расписания
        /// </summary>
        public TypeGenTimetable Type { get;
internal set; }
        /// <summary>
        /// Значение "жестких" ограничений
        /// </summary>
        public  double  HardRestrictValue  {
get; private set; }
        /// <summary>
        /// Значение "мягких" ограничений
        /// </summary>
        public  double  SoftRestrictValue  {
get; private set; }
        /// <summary>
        /// Значение целевой функции
        /// </summary>
        public double RestrictValue
        {
            get { return HardRestrictValue +
SoftRestrictValue;  }
        }
        /// <summary>
        /// Поколение
        /// </summary>
        public int Generation { get; internal
set; }
        /// <summary>
        /// Элемент расписания
        /// </summary>
        /// <param name="i">Позиция</param>
        ///                  <returns>Элемент
расписания</returns>
        public TimetableItem this[int i]
        {
            get { return mItems[i]; }
        }
        public TimetableItem this[FlowData f]
        {
            get { return flowItems[f]; }
        }

        /// <summary>
        /// Дефективные элементы
        /// </summary>
        ///    <returns>Список    дефективных
элементов</returns>
        public List<TimetableItem> Defected()
        {
            return     mItems.Where(f     =>
f.IsDefected).ToList();
        }
        /// <summary>
        /// Недефективные элементы
        /// </summary>
        ///   <returns>Список   недефективных
элементов</returns>
        public            List<TimetableItem>
NoDefected()
        {
            return     mItems.Where(f     =>
!f.IsDefected && !f.NotModified).ToList();
        }
        /// <summary>
        /// Добавление элемента в расписание
        /// </summary>
        /// <param name="item"></param>
        public   void   AddItem(TimetableItem
item)
        {
            mItems.Add(item);
            flowItems.Add(item.Flow, item);
            item.ChangeItem                +=
Item_ChangeItem;
            item.ClearItem += Item_ClearItem;
            foreach(FlowItem       i       in
item.Flow.Items)
            {
                if
(!mGroupTimetable.ContainsKey(i.Group))
                {
```

```
mGroupTimetable.Add(i.Group,              new
TimetableSubject(mGenerator));
                }
            }
            foreach(Teacher       i       in
item.Flow.Teachers.Values)
            {
                if
(!mTeacherTimetable.ContainsKey(i))
                {
                    mTeacherTimetable.Add(i,
new TimetableSubject(mGenerator));
                }
            }
            if (item.Room != null)
            {
                if
(!mRoomTimetable.ContainsKey(item.Room))
                {

mRoomTimetable.Add(item.Room,             new
TimetableSubject(mGenerator));
                }
            }
            Item_ChangeItem(item);
        }

        private                           void
Item_ClearItem(TimetableItem item)
        {
            if   ((item.Day   !=   null)   &&
(item.Time != null) && (item.Room != null))
            {
                foreach   (FlowItem   i   in
item.Flow.Items)
                {

mGroupTimetable[i.Group].ClearItem(item);
                }
                foreach   (Teacher   t   in
item.Flow.Teachers.Values)
                {

mTeacherTimetable[t].ClearItem(item);
                }
                if
(mRoomTimetable.ContainsKey(item.Room))
                {

mRoomTimetable[item.Room].ClearItem(item);
                }
            }
        }

        private                           void
Item_ChangeItem(TimetableItem item)
        {
            if ((item != null) && (item.Time
!= null) && (item.Room != null))
            {
                foreach   (FlowItem   i   in
item.Flow.Items)
                {

mGroupTimetable[i.Group].AddItem(item);
                }
                foreach   (Teacher   t   in
item.Flow.Teachers.Values)
                {

mTeacherTimetable[t].AddItem(item);
                }
                if
(!mRoomTimetable.ContainsKey(item.Room))
                {

mRoomTimetable.Add(item.Room,             new
TimetableSubject(mGenerator));
                }

mRoomTimetable[item.Room].AddItem(item);
            }
        }

        /// <summary>
        /// Число элементов в расписании
        /// </summary>
        public int Count
        {
            get { return mItems.Count; }
        }
        /// <summary>
        /// Сравнение расписаний
        /// </summary>
        ///   <param   name="table">Расписание
для сравнения</param>
        ///       <returns>Одниковое       или
нет</returns>
        public   bool   IsEqual(TimetableData
table)
        {
            for(int i = 0; i < Count; i++)
            {
                if      (this[i].Day      !=
table[i].Day) return false;
                if      (this[i].Room      !=
table[i].Room) return false;
                if      (this[i].Time      !=
table[i].Time) return false;
                if      (this[i].Flow      !=
table[i].Flow) return false;
            }
            return true;
        }
        /// <summary>
        /// Конструктор
        /// </summary>
        ///                               <param
name="generator">Генератор</param>
        public        TimetableData(Generator
generator)
        {
            HardRestrictValue = 0;
            SoftRestrictValue = 0;
            Generation = 0;
            mGenerator = generator;
            mItems            =            new
List<TimetableItem>();
            mGroupTimetable       =       new
Dictionary<Group, TimetableSubject>();
            mTeacherTimetable      =      new
Dictionary<Teacher, TimetableSubject>();
            mRoomTimetable        =        new
Dictionary<Room, TimetableSubject>();
            Type = TypeGenTimetable.Initial;
        }
        /// <summary>
        /// Конструктор для загрузки из файла
        /// </summary>
        ///                               <param
name="generator">Генератор</param>
        ///    <param    name="fileName">Файл
данных</param>
        public        TimetableData(Generator
generator, string fileName) : this(generator)
        {
            XmlDocument     data     =     new
XmlDocument();
            data.Load(fileName);
            foreach(XmlNode        n        in
data.DocumentElement.ChildNodes)
            {
                if (n.Name != "s")
                {
                    continue;
                }
```

```
                string          fid          =
n.Attributes["flowid"].Value;
                FlowData         fd          =
generator.Flows.First(f => f.FlowId == fid);
                TimetableItem    ti    =    new
TimetableItem(fd, this);
                AddItem(ti);
                ti.Day                       =
generator.WeekDays.First(f    =>    f.ID    ==
Convert.ToInt32(n.Attributes["day"].Value));
                ti.Room                      =
generator.Rooms[Convert.ToInt32(n.Attributes[
"room"].Value)];
                foreach(XmlNode     lt     in
n.ChildNodes)
                {
                    if (lt.Name == "lt")
                    {
                        ti.Time             =
generator.Times[Convert.ToInt32(lt.Attributes
["id"].Value)];
                        break;
                    }
                }
            }
        }
        /// <summary>
        /// Обновить значение целевой функции
        /// </summary>
        ///     <param     name="hard">Жесткие
условия</param>
        ///     <param     name="soft">Мягкие
условия</param>
        internal                            void
UpdateRestrictValues(IEnumerable<IRestrict>
hard, IEnumerable<IRestrict> soft)
        {
            HardRestrictValue               =
CheckRestrinct(hard, true, false);
            SoftRestrictValue               =
CheckRestrinct(soft, false, true);
        }
        internal  Dictionary<string,  double>
vals = new Dictionary<string, double>();
        public                              double
GetRestrictItemValue(IRestrict restrict)
        {
            return vals[restrict.Name];
        }
        /// <summary>
        /// Расчет списка ограничений
        /// </summary>
        ///      <param      name="rs">Список
ограничений</param>
        /// <returns>Оценка</returns>
        private                             double
CheckRestrinct(IEnumerable<IRestrict>     rs,
bool hard, bool soft)
        {
            double sum = 0;
            if (rs != null)
            {
                foreach (IRestrict r in rs)
                {
                    double  v  =  r.Check(this,
hard ? r.Factor : 0, soft ? r.Factor : 0) *
r.Factor;
                    vals[r.Name] = v;
                    sum += v;
                }
            }
            return sum;
        }
        /// <summary>
        /// Получить расписание в формате xml
        /// </summary>
        /// <returns>Xml-документ</returns>
        public XmlDocument GetXml()
        {
            StringBuilder    docb    =    new
StringBuilder();
            XmlWriterSettings   st   =   new
XmlWriterSettings();
            st.Encoding = Encoding.UTF8;
            st.OmitXmlDeclaration = false;
            using     (XmlWriter     w     =
XmlWriter.Create(docb, st))
            {
                w.WriteStartElement("root");
                foreach(KeyValuePair<string,
double> k in vals)
                {

w.WriteStartElement("val");

w.WriteAttributeString("name", k.Key);

w.WriteAttributeString("value",
k.Value.ToString());
                    w.WriteEndElement();

                }
                foreach (TimetableItem tu in
this)
                {
                    w.WriteStartElement("s");

w.WriteAttributeString("day",
tu.Day.ID.ToString());

w.WriteAttributeString("dayName",
tu.Day.ToString());

w.WriteAttributeString("room",   tu.Room   ==
null ? "" : tu.Room.RoomID.ToString());

w.WriteAttributeString("roomNumber",  tu.Room
==         null         ?         ""         :
tu.Room.Value<string>("Name"));

w.WriteAttributeString("type",
tu.Flow.TypeFlow.Name);

w.WriteAttributeString("flowid",
tu.Flow.FlowId.ToString());

w.WriteAttributeString("const",
tu.NotModified.ToString());

                    #region defects

foreach(KeyValuePair<string,      Tuple<bool,
double>> s in tu.mDefected)
                    {

w.WriteStartElement("defect");

w.WriteAttributeString("name", s.Key);

w.WriteAttributeString("hard",
s.Value.Item1.ToString());
                        w.WriteEndElement();
                    }
                    #endregion

                    #region LectureTime
                    if (tu.Time != null)
                    {
                        List<LectureTime> lst
= new List<LectureTime>();
                        lst.Add(tu.Time);

lst.AddRange(tu.mAddTimes);
                        foreach  (LectureTime
lt in lst)
                        {

w.WriteStartElement("lt");
```

```
w.WriteAttributeString("id",
lt.LectureTimeID.ToString());

w.WriteAttributeString("num",
lt.NumberLecture.ToString());

w.WriteAttributeString("start",
lt.Value<TimeSpan>("StartTime").ToString(@"hh
\:mm"));

w.WriteAttributeString("finish",
lt.Value<TimeSpan>("FinishTime").ToString(@"h
h\:mm"));

w.WriteEndElement();
                        }
                    }
                    #endregion

                    #region teachers
                    foreach (Teacher tc in
tu.Flow.Teachers.Values)
                    {

w.WriteStartElement("t");

w.WriteAttributeString("id",
tc.TeacherID.ToString());

w.WriteAttributeString("name",
tc.Value<string>("Name"));
                        w.WriteEndElement();
                    }
                    #endregion

                    #region flows
                    foreach (FlowItem flow in
tu.Flow.Items)
                    {

w.WriteStartElement("g");

w.WriteAttributeString("id",
flow.Group.GroupID.ToString());

w.WriteAttributeString("code",
flow.Group.Value<string>("Code"));

w.WriteAttributeString("sub",
flow.SubGroup.ToString());

w.WriteAttributeString("disc",
flow.Discipline.DisciplineID.ToString());

w.WriteAttributeString("discname",
flow.Discipline.Value<string>("Name"));
                        w.WriteEndElement();
                    }
                    #endregion

                    w.WriteEndElement();
                }
                w.WriteEndElement();
            }

            XmlDocument doc = new
XmlDocument();
            doc.LoadXml(docb.ToString());
            return doc;
        }
        /// <summary>
        /// Расписание по группам
        /// </summary>
        public              Dictionary<Group,
TimetableSubject> ScheduleByGroups
        {
            get { return mGroupTimetable; }
        }
        /// <summary>
        /// Расписание по преподавателям
        /// </summary>
        public            Dictionary<Teacher,
TimetableSubject> ScheduleByTeacher
        {
            get { return mTeacherTimetable; }
        }
        /// <summary>
        /// Расписание по аудиториям
        /// </summary>
        public               Dictionary<Room,
TimetableSubject> ScheduleByRoom
        {
            get { return mRoomTimetable; }
        }
        /// <summary>
        /// Проверка занятости аудитории
        /// </summary>
        ///                             <param
name="rm">Аудитория</param>
        ///       <param       name="wd">День
недели</param>
        ///    <param    name="start">Начало
занятия</param>
        /// <param name="ft">Поток</param>
        ///  <returns>Занята  аудитория  или
нет</returns>
        public   bool   IsBusyRoom(Room   rm,
DayItem wd, LectureTime start, FlowData ft)
        {
            if (rm.RoomID == 1057)
            {
                return false;
            }
            if
(mRoomTimetable.ContainsKey(rm))
            {
                LectureTime lt = start;
                for(int   i   =   0;   i   <
ft.CountHours; i++)
                {
                    if (lt == null)
                    {
                        return true;
                    }
                    if
(mRoomTimetable[rm].ByDayTime(wd,   lt).Count
!= 0)
                    {
                        return true;
                    }
                    foreach(DayItem   wdc   in
wd.Cross)
                    {
                        if
(mRoomTimetable[rm].ByDayTime(wdc,  lt).Count
!= 0)
                        {
                            return true;
                        }
                    }
                    lt = lt.Next;
                }

            }
            return false;
        }
        public bool IsBusyTeacher(Teacher t,
DayItem wd, LectureTime start, FlowData ft)
        {
            if
(mTeacherTimetable.ContainsKey(t))
            {
                LectureTime lt = start;
                for(int   i   =   0;   i   <
ft.CountHours; i++)
                {
                    if (lt == null)
```

```
                    {
                        return true;
                    }
                    if
(mTeacherTimetable[t].ByDayTime(wd, lt).Count
!= 0)
                    {
                        return true;
                    }
                    foreach (DayItem wdc in
wd.Cross)
                    {
                        if
(mTeacherTimetable[t].ByDayTime(wdc,
lt).Count != 0)
                        {
                            return true;
                        }
                    }
                    lt = lt.Next;
                }
            }
            return false;
        }
        /// <summary>
        /// Клонирование расписания
        /// </summary>
        ///                     <returns>Новое
расписание</returns>
        public object Clone()
        {
            TimetableData newTimetable = new
TimetableData(mGenerator);
            foreach(TimetableItem i in this)
            {
                TimetableItem   ni   =   new
TimetableItem(i.Flow, newTimetable);
                newTimetable.AddItem(ni);
            }
            for(int i = 0; i < Count; i++)
            {
                newTimetable[i].Day         =
this[i].Day;
                newTimetable[i].Time        =
this[i].Time;
                newTimetable[i].Room        =
this[i].Room;
                newTimetable[i].NotModified =
this[i].NotModified;
            }
            return newTimetable;
        }
        /// <summary>
        /// Перечислитель
        /// </summary>
        /// <returns>Перечислитель</returns>
        public     IEnumerator<TimetableItem>
GetEnumerator()
        {
            return
((IEnumerable<TimetableItem>)mItems).GetEnume
rator();
        }
        /// <summary>
        /// Перечислитель
        /// </summary>
        /// <returns>Перечислитель</returns>
        IEnumerator
IEnumerable.GetEnumerator()
        {
            return
((IEnumerable<TimetableItem>)mItems).GetEnume
rator();
        }
        public   int   IndexOf(TimetableItem
item)
        {
            return mItems.IndexOf(item);
        }
    }
}
```

Файл Timetable\TimetableDataCrosser.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Flow;

namespace ektu.Timetable.Timetable
{
    /// <summary>
    ///   Интефейс   для   класса,   который
производит скрешивание
    /// </summary>
    public interface ITimetableDataCrosser
    {
        /// <summary>
        /// Скрещивание двух расписаний
        /// </summary>
        ///     <param     name="Table1">1-е
расписание</param>
        ///     <param     name="Table2">2-е
расписание</param>
        ///                <returns>Результат
скрещивания</returns>
        TimetableData   Execute(TimetableData
Table1,  TimetableData  Table2,  out  string
info);
    }

    /// <summary>
    /// Класс для выполнения скрешивания
    /// </summary>
    internal  class  TimetableDataCrosser  :
ITimetableDataCrosser
    {
        /// <summary>
        /// Выполнение скрещивания
        /// </summary>
        ///                <returns>Результат
скрешивания</returns>
        public                  TimetableData
Execute(TimetableData  Table1,  TimetableData
Table2, out string info)
        {
            TimetableData     newTable     =
(TimetableData)Table1.Clone();
            double          maxHard         =
newTable.mGenerator.Hard.Sum(f => f.Factor);
            newTable.Type                   =
TypeGenTimetable.Crossing;

newTable.UpdateRestrictValues(newTable.mGener
ator.Hard, newTable.mGenerator.Soft);
            double            h             =
Math.Min(newTable.HardRestrictValue,
Table2.HardRestrictValue);
            double            s             =
Math.Min(newTable.SoftRestrictValue,
Table2.SoftRestrictValue);
            Random rd = new Random();
            int rdg = rd.Next(Table1.Count /
10) + Table1.Count / 10;
            bool isMutate = false;
            for (int i = 0; i < rdg; i++)
            {
                List<Tuple<int,
Generator.DayTimeRoom,
Generator.DayTimeRoom>>    changes    =    new
List<Tuple<int,        Generator.DayTimeRoom,
Generator.DayTimeRoom>>();
                int mi = rd.Next(1, 1);
                for (int j = 0; j < 1; j++)
                {
```

```
                    int         gm         =
rd.Next(Table1.Count);
                    if
(newTable[gm].NotModified)
                    {
                        continue;
                    }
                    Generator.DayTimeRoom it1
= new Generator.DayTimeRoom(newTable[gm].Day,
newTable[gm].Time, newTable[gm].Room);
                    Generator.DayTimeRoom it2
=  new  Generator.DayTimeRoom(Table2[gm].Day,
Table2[gm].Time, Table2[gm].Room);
                    changes.Add(new
Tuple<int,             Generator.DayTimeRoom,
Generator.DayTimeRoom>(gm, it1, it2));

                    foreach(TimetableItem
item in newTable[gm].GetLinked)
                    {
                        it1       =       new
Generator.DayTimeRoom(newTable[item.Flow].Day
,                   newTable[item.Flow].Time,
newTable[item.Flow].Room);
                        it2       =       new
Generator.DayTimeRoom(Table2[item.Flow].Day,
Table2[item.Flow].Time,
Table2[item.Flow].Room);
                        changes.Add(new
Tuple<int,             Generator.DayTimeRoom,
Generator.DayTimeRoom>(newTable.IndexOf(newTa
ble[item.Flow]), it1, it2));
                    }

                    foreach (TimetableItem it
in newTable[gm].GetPar)
                    {
                        it1       =       new
Generator.DayTimeRoom(it.Day,        it.Time,
it.Room);
                        int       ind       =
newTable.IndexOf(it);
                        it2       =       new
Generator.DayTimeRoom(Table2[ind].Day,
Table2[ind].Time, Table2[ind].Room);
                        changes.Add(new
Tuple<int,             Generator.DayTimeRoom,
Generator.DayTimeRoom>(ind, it1, it2));
                    }
                }

                if (changes.Count > 0)
                {
                    changes.ForEach(f      =>
newTable[f.Item1].Assign(f.Item3));

newTable.UpdateRestrictValues(newTable.mGener
ator.Hard, newTable.mGenerator.Soft);
                    if
((newTable.HardRestrictValue > h) ||

((newTable.HardRestrictValue    ==    h)    &&
(newTable.HardRestrictValue != maxHard)) ||

((newTable.HardRestrictValue    ==    h)    &&
(newTable.SoftRestrictValue > s)))
                    {
                        h                   =
newTable.HardRestrictValue;
                        s                   =
newTable.SoftRestrictValue;
                        isMutate = true;
                    }
                    else
                    {
                        changes.ForEach(f  =>
newTable[f.Item1].Assign(f.Item2));
newTable.UpdateRestrictValues(newTable.mGener
ator.Hard, newTable.mGenerator.Soft);
                    }
                }
            }
            info                            =
String.Format("Скрещивание:    [{0}/{1}    #
{2}/{3} -> {4}/{5}]",
                Table1.RestrictValue,
Table1.HardRestrictValue,
                Table2.RestrictValue,
Table2.HardRestrictValue,
                newTable.RestrictValue,
newTable.HardRestrictValue);
            return  isMutate  ?  newTable  :
null;
        }
    }
}
```

Файл Timetable\TimetableDataInitial.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Flow;
using ektu.Timetable.Dict;

namespace ektu.Timetable.Timetable
{
    public interface ITimetableDataInitial
    {
        TimetableData      Generate(Generator
generator,      string      TimetableNumber,
TimetableData parent = null);
    }

    internal  class  TimetableDataInitial  :
ITimetableDataInitial
    {
        public                  TimetableData
Generate(Generator     generator,     string
TimetableNumber, TimetableData parent = null)
        {
            generator.SendInfoAction(" Начало
генерации расписания №" + TimetableNumber);
            TimetableData  timetable  =  new
TimetableData(generator);
            double          mvalRes          =
generator.Soft.Sum(f => f.Factor);
            double        mhardValRed        =
generator.Hard.Sum(f => f.Factor);
            Random rd = new Random();

            generator.Flows.ForEach(f      =>
timetable.AddItem(new        TimetableItem(f,
timetable)));

            int numAdd = 0;
            List<TimetableItem>  items  =  new
List<TimetableItem>();
            foreach  (TimetableItem  item  in
timetable)
            {
                items.Add(item);
            }
            if (parent != null)
            {
                foreach  (TimetableItem  i  in
timetable)
                {
                    TimetableItem    it    =
parent.Where(fl      =>      fl.Flow      ==
i.Flow).FirstOrDefault();
                    if  ((it  !=  null)  &&
(it.Room != null))
```

```
                    {
                        i.Assign(it);
                        items.Remove(i);
                        numAdd++;
                        continue;
                    }
                }
            }

            List<TimetableItem> eitems = new
List<TimetableItem>();
            while (items.Count > 0)
            {
                TimetableItem   getItems   =
items[0];
                items.RemoveAt(0);
                if (getItems.NotModified)
                {
                    continue;
                }
                foreach (TimetableItem t in
getItems.GetPar)
                {
                    items.Remove(t);
                }

                try
                {

TimetableOptimalFinder.FindOptimal(getItems,
false);
                    if
(timetable.HardRestrictValue                <
generator.Hard.Sum(f => f.Factor))
                    {

TimetableOptimalFinder.FindOptimal(getItems,
true);
                    }
                    foreach(TimetableItem   a
in getItems.GetLinked)
                    {

TimetableOptimalFinder.FindOptimal(a, true);
                        items.Remove(a);
                    }
                }
                catch
                {

TimetableOptimalFinder.FindOptimal(getItems,
true);
                    foreach (TimetableItem a
in getItems.GetLinked)
                    {

TimetableOptimalFinder.FindOptimal(a, true);
                        items.Remove(a);
                    }
                }

                if (getItems.Room != null)
                {

generator.SendInfoAction("   Расписание №" +
TimetableNumber +
                        ": добавлен элемент "
+   (++numAdd).ToString()   +   "   из   "   +
generator.Flows.Count.ToString() + " [" +

timetable.RestrictValue.ToString()  +  "/"  +
timetable.HardRestrictValue.ToString()  +  "]"
+

getItems.Flow.Items[0].Discipline.Value<strin
g>("Name") + ", " +

getItems.Flow.Items.Aggregate("", (k1, k2) =>
k1 + " " + k2.Group.Value<string>("Code"), v
=> v)
                        );
                    foreach    (TimetableItem
item2 in getItems.GetPar)
                    {

generator.SendInfoAction("   Расписание №" +
TimetableNumber +
                            ":       добавлен
элемент " + (++numAdd).ToString() + " из " +
generator.Flows.Count.ToString() + " [" +

timetable.RestrictValue.ToString()  +  "/"  +
timetable.HardRestrictValue.ToString()  +  "]"
+

item2.Flow.Items[0].Discipline.Value<string>(
"Name") + ", " +

item2.Flow.Items.Aggregate("", (k1, k2) => k1
+ " " + k2.Group.Value<string>("Code"), v =>
v)
                            );
                    }
                }

                if (eitems.Count > 0)
                {
                    bool fnd = true;
                    while (fnd)
                    {
                        foreach
(TimetableItem  it  in  eitems.OrderBy(f  =>
rd.NextDouble()).Take(25))
                        {

TimetableOptimalFinder.FindOptimal(it, true);
                            if
(timetable.HardRestrictValue                <
generator.Hard.Sum(f => f.Factor))
                            {

it.Assign(null, null, null);
                                foreach
(TimetableItem t in it.GetPar)
                                {

t.Assign(null, null, null);
                                }
                            }
                            else
                            {

generator.SendInfoAction("   Расписание №" +
TimetableNumber +
                                    ":
добавлен      элемент      (доп      "      +
eitems.Count.ToString()    +    ")    "    +
(++numAdd).ToString() + " из " +

generator.Flows.Count.ToString()  +  "  ["  +
timetable.RestrictValue.ToString()  +  "/"  +
timetable.HardRestrictValue.ToString()  +  "]"
+

it.Flow.Items[0].Discipline.Value<string>("Na
me") + ", " +

it.Flow.Items.Aggregate("", (k1, k2) => k1 +
" " + k2.Group.Value<string>("Code"), v => v)
                                    );
                                foreach
(TimetableItem item2 in it.GetPar)
                                {

generator.SendInfoAction("   Расписание №" +
TimetableNumber +
```

```
                                        ":
добавлен      элемент      (доп)      "      +
(++numAdd).ToString()    +    "    из    "    +
generator.Flows.Count.ToString() + " [" +

timetable.RestrictValue.ToString()  +  "/"  +
timetable.HardRestrictValue.ToString()  +  "]"
+

item2.Flow.Items[0].Discipline.Value<string>(
"Name") + ", " +

item2.Flow.Items.Aggregate("", (k1, k2) => k1
+ " " + k2.Group.Value<string>("Code"), v =>
v)
                                            );
                                    }
                                }
                            }
                            fnd                 =
eitems.RemoveAll(f => f.Room != null) > 0;
                        }
                    }
                }

                while (eitems.Count > 0)
                {
                    List<Tuple<double,    double,
List<Room>,       DayItem,       LectureTime,
TimetableItem>> maxV =
                        new    List<Tuple<double,
double,  List<Room>,  DayItem,  LectureTime,
TimetableItem>>();
                    foreach (TimetableItem it in
eitems.Take(1))
                    {

TimetableOptimalFinder.FindOptimal(it, true);
                        maxV.Add(new
Tuple<double,  double,  List<Room>,  DayItem,
LectureTime, TimetableItem>(

timetable.HardRestrictValue,
timetable.SoftRestrictValue,              new
List<Room>(), it.Day, it.Time, it));

maxV.Last().Item3.Add(it.Room);
                        it.Assign(null,     null,
null);
                        foreach  (TimetableItem  t
in it.GetPar)
                        {

maxV.Last().Item3.Add(t.Room);
                            t.Assign(null,  null,
null);
                        }
                    }
                    double   h   =   maxV.Max(f   =>
f.Item1);
                    maxV    =    maxV.Where(f    =>
f.Item1 == h).ToList();
                    double   s   =   maxV.Max(f   =>
f.Item2);
                    Tuple<double,          double,
List<Room>,       DayItem,       LectureTime,
TimetableItem>  fi  =  maxV.Where(f  =>  f.Item2
== s).First();
                    fi.Item6.Assign(fi.Item4,
fi.Item5, fi.Item3.First());
                    int r = 1;
                    foreach  (TimetableItem  t  in
fi.Item6.GetPar)
                    {
                        t.Assign(fi.Item4,
fi.Item5, fi.Item3[r++]);
                    }
                    eitems.Remove(fi.Item6);
                    generator.SendInfoAction("
Расписание №" + TimetableNumber +
                                            ":       добавлен
элемент (доп) " + (++numAdd).ToString() + "
из " + generator.Flows.Count.ToString() + "
[" +

timetable.RestrictValue.ToString()  +  "/"  +
timetable.HardRestrictValue.ToString()  +  "]"
+

fi.Item6.Flow.Items[0].Discipline.Value<strin
g>("Name") + ", " +

fi.Item6.Flow.Items.Aggregate("", (k1, k2) =>
k1 + " " + k2.Group.Value<string>("Code"), v
=> v)
                                );
                    foreach  (TimetableItem  item2
in fi.Item6.GetPar)
                    {

generator.SendInfoAction("   Расписание №" +
TimetableNumber +
                                        ": добавлен элемент
(доп) " + (++numAdd).ToString() + " из " +
generator.Flows.Count.ToString() + " [" +

timetable.RestrictValue.ToString()  +  "/"  +
timetable.HardRestrictValue.ToString()  +  "]"
+

item2.Flow.Items[0].Discipline.Value<string>(
"Name") + ", " +

item2.Flow.Items.Aggregate("", (k1, k2) => k1
+ " " + k2.Group.Value<string>("Code"), v =>
v)
                                );
                    }
                }

timetable.UpdateRestrictValues(generator.Hard
, generator.Soft);

                generator.SendInfoAction("
Завершение   генерации   расписания   №"   +
TimetableNumber);
                return timetable;
            }
        }
}
```

Файл Timetable\TimetableDataMutator.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Flow;
using ektu.Timetable.Dict;
using ektu.Timetable.Restrict;

namespace ektu.Timetable.Timetable
{
    /// <summary>
    ///   Интефейс   для   класса,   который
производит мутацию
    /// </summary>
    public interface ITimetableDataMutator
    {
        /// <summary>
        /// Мутация расписания
        /// </summary>
        ///                             <param
name="Table">Расписание</param>
        ///                 <returns>Результат
мутации</returns>
```

```
        TimetableData Execute(TimetableData
Table, out string InfoAction);
    }

    /// <summary>
    /// Класс для проведения мутации
расписания
    /// </summary>
    internal class TimetableDataMutator :
ITimetableDataMutator
    {
        /// <summary>
        /// Мутация расписания
        /// </summary>
        /// <param
name="Table">Расписание</param>
        /// <param name="InfoAction">Данные о
мутации</param>
        /// <returns>Результат
мутации</returns>
        public TimetableData
Execute(TimetableData Table, out string
InfoAction)
        {
            // создание новой вариации
расписания
            TimetableData newTable =
(TimetableData)Table.Clone();
            newTable.Type =
TypeGenTimetable.Mutating;

newTable.UpdateRestrictValues(newTable.mGener
ator.Hard, newTable.mGenerator.Soft);
            Tuple<double, double> old = new
Tuple<double, double>(newTable.RestrictValue,
newTable.HardRestrictValue);
            InfoAction = "";
            Random rd = new Random();

            // случайная мутация генов по
определенному нарушению
            if (rd.NextDouble() > 0.8)
            {
                RandomMutation(newTable, 50);
                InfoAction =
String.Format("Мутация случайных генов:
[{0}/{1} -> {2}/{3}]",
                    old.Item1, old.Item2,
newTable.RestrictValue,
newTable.HardRestrictValue);
            }
            // мутация случайных пар генов
            else if (rd.NextDouble() > 0.7)
            {
                RandomGroupMutation(newTable,
50);
                InfoAction =
String.Format("Мутация группы случайных
генов: [{0}/{1} -> {2}/{3}]",
                    old.Item1, old.Item2,
newTable.RestrictValue,
newTable.HardRestrictValue);
            }
            //
            else if ((rd.NextDouble() > 0.5)
&& (newTable.Defected().Count > 0))
            {
                DefectMutation(newTable);
                InfoAction =
String.Format("Мутация дефектного гена:
[{0}/{1} -> {2}/{3}]",
                    old.Item1, old.Item2,
newTable.RestrictValue,
newTable.HardRestrictValue);
            }
            // мутация с выборкой генов со
одинаковыми значениями переменных (ппс,
аудитория, группа)
            else
            {
                MutationSelect(newTable,
0.1);
                InfoAction =
String.Format("Мутация с выборкой генов:
[{0}/{1} -> {2}/{3}]",
                    old.Item1, old.Item2,
newTable.RestrictValue,
newTable.HardRestrictValue);
            }

            foreach (var e in newTable)
            {
                if (e.Day == null)
                {
                    InfoAction += ", Неверный
день";
                    return null;
                }
                if (e.Time == null)
                {
                    InfoAction += ", Неверное
время";
                    return null;
                }
                if (e.Room == null)
                {
                    InfoAction += ", Неверная
аудитория";
                    return null;
                }

            }

            return newTable;
        }

        /// <summary>
        /// Мутация случайных генов, которые
нарушают случайное ограничение
        /// </summary>
        /// <param
name="newTable">Расписание</param>
        /// <param name="count">Кол-во генов
для мутации</param>
        private void
RandomMutation(TimetableData newTable, int
count)
        {
            Random rd = new Random();
            IRestrict restrict = null;
            if (rd.NextDouble() > 0.7)
            {
                restrict =
newTable.mGenerator.Soft.OrderBy(f =>
rd.NextDouble()).First();
            }
            else
            {
                restrict =
newTable.mGenerator.Hard.OrderBy(f =>
rd.NextDouble()).First();
            }
            //restrict =
newTable.mGenerator.Hard.Where(f => f.Name ==
"RestrictGroupFill").First();
            List<TimetableItem> lst =
newTable
                .ToList()
                .Where(f =>
f.mDefected.Keys.ToList().Exists(fx =>
fx.StartsWith(restrict.Name + "=")))
                .OrderByDescending(f =>
rd.NextDouble() * f.HardDefectEval +
rd.NextDouble() * f.SoftDefectEval)
                .Take(count)
                .ToList();
            for(int i = 0; i < lst.Count;
i++)
```

```
            {
                foreach(TimetableItem e in
lst[i].GetLinked)
                {
                    lst.Remove(e);
                }
                foreach (TimetableItem e in
lst[i].GetPar)
                {
                    lst.Remove(e);
                }
            }
            lst.ForEach(f => f.Assign(null,
null, null));
            foreach (TimetableItem i in lst)
            {

TimetableOptimalFinder.FindOptimal(i, true,
restrict);
                foreach(TimetableItem j in
i.GetLinked)
                {
                    j.Assign(null, null,
null);

TimetableOptimalFinder.FindOptimal(j, true,
restrict);
                }
            }

newTable.UpdateRestrictValues(newTable.mGener
ator.Hard, newTable.mGenerator.Soft);
        }

        /// <summary>
        /// Мутация случайных пар генов
        /// </summary>
        ///                                <param
name="newTable">Расписание</param>
        ///     <param     name="Count">Кол-во
генов</param>
        private                             void
RandomGroupMutation(TimetableData newTable,
int Count)
        {
            Random rd = new Random();
            List<TimetableItem>     it     =
newTable.Defected()
                    .OrderByDescending(f =>
f.HardDefectEval * rd.NextDouble())
                    .ThenByDescending(f =>
f.SoftDefectEval * rd.NextDouble())
                    .Take(Count)
                    .ToList();
            for (int i = 0; i < it.Count;
i++)
            {
                foreach (TimetableItem e in
it[i].GetLinked)
                {
                    it.Remove(e);
                }
                foreach (TimetableItem e in
it[i].GetPar)
                {
                    it.Remove(e);
                }
            }
            it.ForEach(f => f.Assign(null,
null, null));
            while (it.Count > 0)
            {
                if (it[0].GetLinked.Count >
0)
                {

TimetableOptimalFinder.FindOptimal2(it[0],
it[0].GetLinked[0], true);
                    for(int i = 1; i <
it[0].GetLinked.Count; i += 2)
                    {
                        if (i + 1 <
it[0].GetLinked.Count)
                        {

TimetableOptimalFinder.FindOptimal2(it[0].Get
Linked[i], it[0].GetLinked[i + 1], true);
                        }
                        else
                        {

TimetableOptimalFinder.FindOptimal(it[0].GetL
inked[i], true);
                        }
                    }
                    it.RemoveAt(0);
                }
                else
                {
                    if (it.Count > 1)
                    {

TimetableOptimalFinder.FindOptimal2(it[0],
it[1], true);
                        it.RemoveAt(0);
                        it.RemoveAt(0);
                    }
                    else
                    {

TimetableOptimalFinder.FindOptimal(it[0],
true);
                        it.RemoveAt(0);
                    }
                }
            }

newTable.UpdateRestrictValues(newTable.mGener
ator.Hard, newTable.mGenerator.Soft);
        }

        /// <summary>
        /// Мутация с выборкой по группе,
ППС, аудитории
        /// </summary>
        ///                                <param
name="newTable">Расписание</param>
        /// <param name="percent">% генов для
мутации</param>
        private                             void
MutationSelect(TimetableData newTable, double
percent)
        {
            List<TimetableItem> def = new
List<TimetableItem>();
            int             dsC             =
Convert.ToInt32(newTable.Count(fx          =>
!fx.NotModified) * percent);
            Random rd = new Random();

            while ((def.Count < dsC) &&
(newTable.Defected().Count > 0))
            {
                TimetableItem di = null;
                if (rd.NextDouble() < 0.3)
                {
                    di = newTable
                        .Where(f          =>
!def.Contains(f))
                        .OrderBy(f        =>
rd.NextDouble())
                        .First();
                }
                else
                {
                    di = newTable.Defected()
```

```
                        .Where(f =>
!def.Contains(f))
                        .OrderByDescending(f
=> f.HardDefectEval * rd.NextDouble())
                        .ThenByDescending(f
=> f.SoftDefectEval * rd.NextDouble())
                        .First();
                }

                if (rd.NextDouble() > 0.8)
                {
                    foreach    (TimetableItem
itm in newTable)
                    {
                        if    (itm.Room    ==
di.Room)
                        {
                            if
((!def.Contains(itm)) && (!itm.NotModified))
                            {
                                def.Add(itm);
                            }
                        }
                    }
                }
                else  if  (rd.NextDouble()  >
0.7)
                {
                    foreach    (TimetableItem
itm in newTable)
                    {
                        foreach (int teacher
in di.Flow.Teachers.Keys)
                        {
                            if
(itm.Flow.Teachers.ContainsKey(teacher))
                            {
                                if
((!def.Contains(itm)) && (!itm.NotModified))
                                {

def.Add(itm);
                                }
                            }
                        }
                    }
                }
                else
                {
                    List<Group>     grs     =
di.Flow.Items.Select(f => f.Group).ToList();
                    foreach    (TimetableItem
itm in newTable)
                    {
                        if
(itm.Flow.Items.Exists(f                    =>
grs.Contains(f.Group)))
                        {
                            if
((!def.Contains(itm)) && (!itm.NotModified))
                            {
                                def.Add(itm);
                            }
                        }
                    }
                }

                def.ForEach(f               =>
f.Assign(null, null, null));

newTable.UpdateRestrictValues(newTable.mGener
ator.Hard, newTable.mGenerator.Soft);
            }

            def     =     def.OrderBy(f     =>
rd.NextDouble()).ToList();

            for (int i = 0; i < def.Count;
i++)
            {
                foreach (TimetableItem e in
def[i].GetLinked)
                {
                    def.Remove(e);
                }
                foreach (TimetableItem e in
def[i].GetPar)
                {
                    def.Remove(e);
                }
            }

            while (def.Count > 0)
            {

TimetableOptimalFinder.FindOptimal(def[0],
true);
                foreach(TimetableItem   e   in
def[0].GetLinked)
                {

TimetableOptimalFinder.FindOptimal(e, true);
                }
                def.RemoveAt(0);
            }
        }

        /// <summary>
        /// Мутация случайно выбранного гена
с дефектом и связанных с ним генов по
изменяемым    переменных    (ппс,    группа,
аудитория)
        /// </summary>
        ///                             <param
name="newTable">Расписание</param>
        private                             void
DefectMutation(TimetableData newTable)
        {
            Random rd = new Random();
            TimetableItem       item       =
newTable.Defected().OrderBy(f               =>
rd.NextDouble()).First();

            List<TimetableItem> items = new
List<TimetableItem>();
            items.Add(item);

            foreach(TimetableItem    i    in
newTable)
            {
                if (i.Room == item.Room)
                {
                    items.Add(i);
                    continue;
                }
                foreach   (int   teacher   in
item.Flow.Teachers.Keys)
                {
                    if
(i.Flow.Teachers.ContainsKey(teacher))
                    {
                        items.Add(i);
                        continue;
                    }
                }
                foreach(FlowItem    fi    in
item.Flow.Items)
                {
                    if (i.Flow.Items.Exists(f
=> f.Group == fi.Group))
                    {
                        items.Add(i);
                        continue;
                    }
                }
            }
```

```
            items                               =
items.Distinct().ToList();
            for (int i = 0; i < items.Count;
i++)
            {
                foreach (TimetableItem e in
items[i].GetLinked)
                {
                    items.Remove(e);
                }
                foreach (TimetableItem e in
items[i].GetPar)
                {
                    items.Remove(e);
                }
            }

            items.ForEach(f => f.Assign(null,
null, null));

newTable.UpdateRestrictValues(newTable.mGener
ator.Hard, newTable.mGenerator.Soft);

            while (items.Count != 0)
            {

TimetableOptimalFinder.FindOptimal(items[0],
true);
                foreach(TimetableItem   e   in
items[0].GetLinked)
                {

TimetableOptimalFinder.FindOptimal(e, true);
                }
                items.RemoveAt(0);
            }


newTable.UpdateRestrictValues(newTable.mGener
ator.Hard, newTable.mGenerator.Soft);

        }
    }
}
```

Файл Timetable\TimetableItem.cs

```
using System;
using System.Collections.Generic;
using System.Collections.Concurrent;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Dict;
using ektu.Timetable.Flow;

namespace ektu.Timetable.Timetable
{
    /// <summary>
    /// Элемент расписания
    /// </summary>
    public class TimetableItem
    {
        public bool NotModified { get; set; }
        public   ConcurrentDictionary<string,
Tuple<bool,   double>>   mDefected   =   new
ConcurrentDictionary<string,      Tuple<bool,
double>>();
        public double HardDefectEval
        {
            get
            {
                return   mDefected.Sum(f   =>
f.Value.Item1 ? f.Value.Item2 : 0);
            }
        }
        public double SoftDefectEval
        {
            get
            {
                return   mDefected.Sum(f   =>
f.Value.Item1 ? 0 : f.Value.Item2);
            }
        }
        /// <summary>
        /// День недели
        /// </summary>
        private DayItem mDay;
        /// <summary>
        /// Время занятия
        /// </summary>
        private LectureTime mTime;
        /// <summary>
        /// Дополнительное время для элемента
        /// </summary>
        internal LectureTime[] mAddTimes;
        /// <summary>
        /// Аудитории
        /// </summary>
        private Room mRoom;
        /// <summary>
        /// Поток
        /// </summary>
        public FlowData Flow { get; private
set; }
        /// <summary>
        /// Владелец элемента
        /// </summary>
        public  TimetableData  Owner  {  get;
private set; }
        private  List<TimetableItem>  mPar  =
null;
        /// <summary>
        /// Список параллельных элементов
        /// </summary>
        internal List<TimetableItem> GetPar
        {
            get
            {
                if (mPar == null)
                {
                    List<TimetableItem>  r  =
new List<TimetableItem>();
                    if (Flow.Par.Count > 0)
                    {
                        foreach
(TimetableItem t in Owner)
                        {
                            if
(Flow.Par.Contains(t.Flow))
                            {
                                r.Add(t);
                            }
                        }
                    }
                    mPar = r;
                }
                return mPar;
            }
        }
        private List<TimetableItem> mLinked =
null;
        /// <summary>
        /// Список связанных генов
        /// </summary>
        internal          List<TimetableItem>
GetLinked
        {
            get
            {
                if (mLinked == null)
                {
                    List<TimetableItem>  r  =
new List<TimetableItem>();
                    if  (Flow.Linked.Count  >
0)
                    {
```

```
                        foreach(TimetableItem
t in Owner)
                        {
                            if
(Flow.Linked.Contains(t.Flow))
                            {
                                r.Add(t);
                            }
                        }
                    }
                    mLinked = r;
                }
                return mLinked;
            }
        }
        /// <summary>
        /// Пометка дефективности элемента
        /// </summary>
        /// <param name="Code">Код</param>
        /// <param name="IsDefected">Статус -
дефективный или нет</param>
        public void SetDefected(string Code,
bool IsDefected, double hard, double soft)
        {
            if (IsDefected)
            {
                mDefected[Code]     =     new
Tuple<bool, double>(hard > 0, hard + soft);
            }
            else
            {
                Tuple<bool, double> v;
                mDefected.TryRemove(Code, out
v);
            }
        }
        /// <summary>
        /// Дефективный или нет
        /// </summary>
        public bool IsDefected
        {
            get { return mDefected.Count > 0;
}
        }
        /// <summary>
        /// Число дефектов
        /// </summary>
        public int DefectedCount
        {
            get { return mDefected.Count; }
        }
        public          delegate          void
ChangeItemDelegate(TimetableItem item);
        public    event    ChangeItemDelegate
ChangeItem;
        public    event    ChangeItemDelegate
ClearItem;
        /// <summary>
        /// День недели
        /// </summary>
        public DayItem Day
        {
            get { return mDay; }
            set
            {
                if (mDay == value) return;
                if (NotModified) return;
                ClearItem?.Invoke(this);
                mDay = value;
                ChangeItem?.Invoke(this);
                mDefected.Clear();
                foreach  (TimetableItem  i  in
GetPar)
                {
                    i.Day = value;
                }
            }
        }
        /// <summary>
        /// Время занятий
        /// </summary>
        public LectureTime Time
        {
            get { return mTime; }
            set
            {
                if (mTime == value) return;
                if (NotModified) return;
                ClearItem?.Invoke(this);
                mTime = value;
                if (mTime == null)
                {
                    for  (int  i  =  0;  i  <
Flow.CountHours - 1; i++)
                    {
                        mAddTimes[i] = null;
                    }
                }
                else
                {
                    LectureTime     lt     =
mTime.Next;
                    for  (int  i  =  0;  i  <
Flow.CountHours - 1; i++)
                    {
                        mAddTimes[i] = lt;
                        if (lt == null)
                        {
                            throw         new
Exception("errr");
                        }
                        lt = lt.Next;
                    }
                }
                ChangeItem?.Invoke(this);
                mDefected.Clear();
                foreach  (TimetableItem  i  in
GetPar)
                {
                    i.Time = value;
                }
            }
        }
        /// <summary>
        /// Аудитория
        /// </summary>
        public Room Room
        {
            get { return mRoom; }
            set
            {
                if (mRoom == value) return;
                if (NotModified) return;
                ClearItem?.Invoke(this);
                mRoom = value;
                ChangeItem?.Invoke(this);
                mDefected.Clear();
            }
        }
        /// <summary>
        ///  Копировать  данные  из  другого
элемента
        /// </summary>
        ///  <param  name="item">Элемент  из
которого копируются данные</param>
        public   void   Assign(TimetableItem
item)
        {
            Assign(item.Day,       item.Time,
item.Room);
        }
        internal                          void
Assign(Generator.DayTimeRoom values)
        {
            Assign(values.Day,   values.Time,
values.Room);
        }
        /// <summary>
```

```
        /// Назначить все элементы сразу
        /// </summary>
        /// <param name="wd">День</param>
        /// <param name="lt">Время</param>
        ///                                 <param
name="room">Аудитория</param>
        public   void   Assign(DayItem   wd,
LectureTime lt, Room room)
        {
            Day = wd;
            Time = lt;
            Room = room;
        }

        /// <summary>
        /// Конструктор
        /// </summary>
        /// <param name="flow">Поток</param>
        ///                                 <param
name="owner">Расписание</param>
        public  TimetableItem(FlowData  flow,
TimetableData owner)
        {
            Flow = flow;
            Owner = owner;
            mAddTimes          =          new
LectureTime[flow.CountHours - 1];
            NotModified = false;
        }

        /// <summary>
        /// Сохраненные состояния
        /// </summary>
        //private List<Generator.DayTimeRoom>
mStates = new List<Generator.DayTimeRoom>();
        /// <summary>
        /// Сохранить состояние
        /// </summary>
        //internal void PushState()
        //{
        //                    mStates.Add(new
Generator.DayTimeRoom(Day, Time, Room));
        //      foreach(TimetableItem item in
GetPar)
        //    {
        //        item.PushState();
        //    }
        //}
        /// <summary>
        /// Восстановить состояние
        /// </summary>
        //internal void PopState()
        //{
        //    if (mStates.Count > 0)
        //    {
        //
Assign(mStates[mStates.Count - 1]);
        //          foreach(TimetableItem item
in GetPar)
        //        {
        //            item.PopState();
        //        }
        //
mStates.RemoveAt(mStates.Count - 1);
        //    }
        //}

        private TimetableItem()
        {
            NotModified = true;
            Flow = new FlowData();
            Flow.CountHours = 0;
            mAddTimes = new LectureTime[0];
        }

    }

    public class TimetableItemProxy
    {
        public  TimetableItem  data  {  get;
private set; }
        public bool IsBlank { get; private
set; }
        public
TimetableItemProxy(TimetableItem  item,  bool
blank = false)
        {
            data = item;
            IsBlank = blank;
        }
    }
}
```

Файл Timetable\TimetableOptimalFinder.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Dict;
using ektu.Timetable.Restrict;
using ektu.Timetable.Flow;

namespace ektu.Timetable.Timetable
{
    using             SavedData             =
List<Tuple<TimetableItem,            DayItem,
LectureTime, Room>>;

    internal           static           class
TimetableOptimalFinder
    {
        /// <summary>
        ///  Поиск  свободной  аудитории  для
элемента расписания
        /// </summary>
        ///     <param     name="item">Элемент
расписания</param>
        /// <param name="day">День</param>
        /// <param name="time">Время</param>
        /// <returns>Свободная аудитория или
null, если аудитория не найдена</returns>
        private          static          Room
FindNonBusyRoom(TimetableItem  item,  DayItem
day, LectureTime time)
        {
            Random rd = new Random();
            RoomSelectInfo       room       =
item.Flow.Rooms
                .Where(f                    =>
!item.Owner.mGenerator.Excluder.Exclude(item,
f.Room, day, time) &&

!item.Owner.IsBusyRoom(f.Room,   day,   time,
item.Flow))
                .OrderBy(f => f.Priority)
                .ThenBy(f => rd.NextDouble())
                .FirstOrDefault();
            if (room == null)
            {
                return null;
            }
            return room.Room;
        }

        /// <summary>
        ///  Поиск  аудитории  для  элемента
расписания
        /// </summary>
        ///     <param     name="item">Элемент
расписания</param>
        /// <param name="day">День</param>
        /// <param name="time">Время</param>
        /// <returns>Аудитория или null, если
аудитория не найдена</returns>
```

```
        private          static          Room
FindAnyRoom(TimetableItem item, DayItem day,
LectureTime time)
        {
            Random rd = new Random();
            RoomSelectInfo       room       =
item.Flow.Rooms
                .Where(f                    =>
!item.Owner.mGenerator.Excluder.Exclude(item,
f.Room, day, time))
                .OrderBy(f => f.Priority)
                .ThenBy(f => rd.NextDouble())
                .FirstOrDefault();
            if (room == null)
            {
                return null;
            }
            return room.Room;
        }

        /// <summary>
        /// Очиста элемента расписания
        /// </summary>
        ///     <param     name="item">Элемент
расписания</param>
        private          static          void
ClearData(TimetableItem item)
        {
            item.Assign(null, null, null);
            foreach   (TimetableItem   par   in
item.GetPar)
            {
                par.Assign(null, null, null);
            }
        }

        /// <summary>
        /// Сохранение значений для элемента
расписаний
        /// </summary>
        ///     <param     name="item">Элемент
расписания</param>
        ///               <returns>Сохраненные
значения</returns>
        private       static       SavedData
SaveItemValues(TimetableItem item)
        {
            SavedData     maxData     =     new
SavedData();
            maxData.Add(new
Tuple<TimetableItem,   DayItem,   LectureTime,
Room>(item, item.Day, item.Time, item.Room));
            foreach   (TimetableItem   par   in
item.GetPar)
            {
                maxData.Add(new
Tuple<TimetableItem,   DayItem,   LectureTime,
Room>(par, item.Day, item.Time, par.Room));
            }
            return maxData;
        }

        internal          static          void
FindOptimal(TimetableItem item, bool timeAll,
IRestrict restrict = null)
        {
            // задаем начальные значения для
ограничений
            double hard = 0;
            double soft = 0;
            double rest = 0;

            // сохраняем предыдущие значения
элемента расписания
            SavedData maxData = null;
            if   ((item.Day   !=   null)   &&
(item.Time != null) && (item.Room != null))
            {
                maxData                        =
SaveItemValues(item);
                hard                           =
item.Owner.HardRestrictValue;
                soft                           =
item.Owner.SoftRestrictValue;
                if (restrict != null)
                {
                    rest                       =
item.Owner.GetRestrictItemValue(restrict);
                }
            }

            //   очищаем   значения   элемента
расписания
            //    если    есть    параллельные
занятия, то ищем элемент с минимальным числом
аудиторий
            item.Assign(null, null, null);
            TimetableItem titem = item;
            foreach   (TimetableItem   par   in
item.GetPar)
            {
                par.Assign(null, null, null);
                if   (par.Flow.Rooms.Count   <
titem.Flow.Rooms.Count)
                {
                    titem = par;
                }
            }
            item = titem;
            Random rd = new Random();

            //   выбираем   возможные   день   и
время для занятия
            IEnumerable<SelectDataWeekTime>
sdt = item.Flow.DayTimes
                        .Where(f   =>   timeAll
|| (f.Priority == 0))
                        .OrderBy(f          =>
f.Priority)
                        .ThenBy(f           =>
rd.NextDouble());
            if (sdt.Count() == 0)
            {
                sdt = item.Flow.DayTimes
                        .OrderBy(f          =>
f.Priority)
                        .ThenBy(f           =>
rd.NextDouble());
            }

            //   получаем   список   ППС   для
занятий
            List<Teacher>   teachers   =   new
List<Teacher>();

teachers.AddRange(item.Flow.Teachers.Values);
            foreach   (TimetableItem   it   in
item.GetPar)
            {

teachers.AddRange(it.Flow.Teachers.Values);
            }
            teachers                           =
teachers.Distinct().ToList();

            // перебор возможных значений для
дня и времени
            foreach (SelectDataWeekTime dt in
sdt)
            {
                // если в указанное время
какой-либо ППС занят
                if     (teachers.Any(fx     =>
item.Owner.IsBusyTeacher(fx, dt.Day, dt.Time,
item.Flow)))
                {
                    if (item.Time != null)
```

```
                    {
                        continue;
                    }
                }

                // ищем свободные аудитории
                Room          srm          =
FindNonBusyRoom(item, dt.Day, dt.Time);
                bool IsFindNonBusyRooms = srm
!= null;
                if (IsFindNonBusyRooms)
                {
                    item.Assign(dt.Day,
dt.Time, srm);
                    foreach    (TimetableItem
par in item.GetPar)
                    {
                        srm                =
FindNonBusyRoom(par, dt.Day, dt.Time);
                        if (srm != null)
                        {
                            par.Room = srm;
                        }
                        else
                        {

IsFindNonBusyRooms = false;
                            break;
                        }
                    }
                }

                // если свободные аудитории
не найдены
                if (!IsFindNonBusyRooms)
                {
                    ClearData(item);
                    srm  =  FindAnyRoom(item,
dt.Day, dt.Time);
                    if (srm == null)
                    {
                        continue;
                    }
                    item.Assign(dt.Day,
dt.Time, srm);
                    foreach    (TimetableItem
par in item.GetPar)
                    {
                        srm                =
FindAnyRoom(par, dt.Day, dt.Time);
                        if (srm != null)
                        {
                            par.Room = srm;
                        }
                        else
                        {
                            break;
                        }
                    }
                    if (srm == null)
                    {
                        ClearData(item);
                        continue;
                    }
                }

                //    определяем    значение
ограничений

item.Owner.UpdateRestrictValues(item.Owner.mG
enerator.Hard, item.Owner.mGenerator.Soft);

                // определяем если полученное
значение оптимальное, то сохраняем его
                // если оптимизируем общее
значение ограничений
                if (restrict == null)
                {
                    if
((item.Owner.HardRestrictValue > hard) ||
                        (maxData == null) ||

((item.Owner.HardRestrictValue  ==  hard)  &&
(item.Owner.SoftRestrictValue > soft)))
                    {
                        maxData             =
SaveItemValues(item);
                        hard                =
item.Owner.HardRestrictValue;
                        soft                =
item.Owner.SoftRestrictValue;
                    }
                }
                //     если     оптимизируем
конкретное значение
                else
                {
                    if
((item.Owner.GetRestrictItemValue(restrict) >
rest) ||
                        (maxData == null))
                    {
                        maxData             =
SaveItemValues(item);
                        rest                =
item.Owner.GetRestrictItemValue(restrict);
                    }
                }
                ClearData(item);
            }

            // Назначаем полученные значения
для элементов расписания
            foreach(var s in maxData)
            {
                s.Item1.Assign(s.Item2,
s.Item3, s.Item4);
            }

item.Owner.UpdateRestrictValues(item.Owner.mG
enerator.Hard, item.Owner.mGenerator.Soft);
        }

        internal          static          void
FindOptimal2(TimetableItem            item1,
TimetableItem item2, bool timeAll, IRestrict
restrict = null)
        {
            Dictionary<TimetableItem,
Tuple<DayItem, LectureTime, Room>> m0 = new
Dictionary<TimetableItem,      Tuple<DayItem,
LectureTime, Room>>();
            m0.Add(item1, new Tuple<DayItem,
LectureTime,   Room>(item1.Day,   item1.Time,
item1.Room));
            m0.Add(item2, new Tuple<DayItem,
LectureTime,   Room>(item2.Day,   item2.Time,
item2.Room));
            foreach   (TimetableItem   i   in
item1.GetPar)
            {
                m0.Add(i, new Tuple<DayItem,
LectureTime, Room>(i.Day, i.Time, i.Room));
            }
            foreach   (TimetableItem   i   in
item2.GetPar)
            {
                m0.Add(i, new Tuple<DayItem,
LectureTime, Room>(i.Day, i.Time, i.Room));
            }


            Dictionary<TimetableItem,
Tuple<DayItem, LectureTime, Room>> m1 = new
Dictionary<TimetableItem,      Tuple<DayItem,
LectureTime, Room>>();
```

```
            Dictionary<TimetableItem,
Tuple<DayItem, LectureTime, Room>> m2 = new
Dictionary<TimetableItem,      Tuple<DayItem,
LectureTime, Room>>();
            Dictionary<TimetableItem,
Tuple<DayItem,  LectureTime,  Room>>  max  =
null;
            double hard1 = 0;
            double hard2 = 0;
            double soft1 = 0;
            double soft2 = 0;
            double rest1 = 0;
            double rest2 = 0;

            foreach   (TimetableItem   i   in
item1.GetPar)
            {
                i.Assign(null, null, null);
            }
            foreach   (TimetableItem   i   in
item2.GetPar)
            {
                i.Assign(null, null, null);
            }
            item1.Assign(null, null, null);
            item2.Assign(null, null, null);
            FindOptimal(item1,       timeAll,
restrict);
            FindOptimal(item2,       timeAll,
restrict);
            m1.Add(item1, new Tuple<DayItem,
LectureTime,   Room>(item1.Day,   item1.Time,
item1.Room));
            m1.Add(item2, new Tuple<DayItem,
LectureTime,   Room>(item2.Day,   item2.Time,
item2.Room));
            foreach   (TimetableItem   i   in
item1.GetPar)
            {
                m1.Add(i, new Tuple<DayItem,
LectureTime, Room>(i.Day, i.Time, i.Room));
            }
            foreach   (TimetableItem   i   in
item2.GetPar)
            {
                m1.Add(i, new Tuple<DayItem,
LectureTime, Room>(i.Day, i.Time, i.Room));
            }

item1.Owner.UpdateRestrictValues(item1.Owner.
mGenerator.Hard,
item1.Owner.mGenerator.Soft);
            hard1                          =
item1.Owner.HardRestrictValue;
            soft1                          =
item1.Owner.HardRestrictValue;
            if (restrict != null)
            {
                rest1                      =
item1.Owner.GetRestrictItemValue(restrict);
            }

            foreach   (TimetableItem   i   in
item1.GetPar)
            {
                i.Assign(null, null, null);
            }
            foreach   (TimetableItem   i   in
item2.GetPar)
            {
                i.Assign(null, null, null);
            }
            item1.Assign(null, null, null);
            item2.Assign(null, null, null);
            FindOptimal(item2,       timeAll,
restrict);
            FindOptimal(item1,       timeAll,
restrict);
            m2.Add(item1, new Tuple<DayItem,
LectureTime,   Room>(item1.Day,   item1.Time,
item1.Room));
            m2.Add(item2, new Tuple<DayItem,
LectureTime,   Room>(item2.Day,   item2.Time,
item2.Room));
            foreach   (TimetableItem   i   in
item1.GetPar)
            {
                m2.Add(i, new Tuple<DayItem,
LectureTime, Room>(i.Day, i.Time, i.Room));
            }
            foreach   (TimetableItem   i   in
item2.GetPar)
            {
                m2.Add(i, new Tuple<DayItem,
LectureTime, Room>(i.Day, i.Time, i.Room));
            }

item1.Owner.UpdateRestrictValues(item1.Owner.
mGenerator.Hard,
item1.Owner.mGenerator.Soft);
            hard2                          =
item1.Owner.HardRestrictValue;
            soft2                          =
item1.Owner.HardRestrictValue;
            if (restrict != null)
            {
                rest2                      =
item1.Owner.GetRestrictItemValue(restrict);
            }

            if (restrict == null)
            {
                if (hard1 > hard2)
                {
                    max = m1;
                }
                if (hard2 > hard1)
                {
                    max = m2;
                }
                if   ((hard1   ==   hard2)   &&
(soft1 > soft2))
                {
                    max = m1;
                }
                if   ((hard1   ==   hard2)   &&
(soft2 > soft1))
                {
                    max = m2;
                }
            }
            else
            {
                if (rest1 > rest2)
                {
                    max = m1;
                }
                else
                {
                    max = m2;
                }
            }

            max = max ?? m1;
            if (max.Where(f => (f.Value.Item1
==  null)  ||  (f.Value.Item2  ==  null)  ||
(f.Value.Item3 == null)).Count() != 0)
            {
                max = m0;
            }

            item1.Assign(max[item1].Item1,
max[item1].Item2, max[item1].Item3);
            item2.Assign(max[item2].Item1,
max[item2].Item2, max[item2].Item3);
            foreach   (TimetableItem   i   in
item1.GetPar)
```

```
            {
                i.Assign(max[i].Item1,
max[i].Item2, max[i].Item3);
            }
            foreach   (TimetableItem   i   in
item2.GetPar)
            {
                i.Assign(max[i].Item1,
max[i].Item2, max[i].Item3);
            }

item1.Owner.UpdateRestrictValues(item1.Owner.
mGenerator.Hard,
item1.Owner.mGenerator.Soft);

        }
    }
}
```

Файл Timetable\TimetableSubject.cs

```
using System;
using System.Collections.Generic;
using System.Collections.Concurrent;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Dict;

namespace ektu.Timetable.Timetable
{
    /// <summary>
    ///   Расписание   по   объекту   (группа,
аудитория, преподаватель)
    /// </summary>
    public class TimetableSubject
    {
        private Generator gn;
        private      List<TimetableItemProxy>
emptyList = new List<TimetableItemProxy>();
        private           Dictionary<DayItem,
Dictionary<LectureTime,
List<TimetableItemProxy>>> tMainData = null;
        /// <summary>
        /// Элементы расписания
        /// </summary>
        private           Dictionary<DayItem,
Dictionary<LectureTime,
List<TimetableItemProxy>>> MainData
        {
            get
            {
                if (tMainData == null)
                {
                    tMainData      =      new
Dictionary<DayItem,   Dictionary<LectureTime,
List<TimetableItemProxy>>>();
                    foreach  (DayItem  wd  in
gn.WeekDays.Where(f => f.Cross.Count == 0))
                    {
                        tMainData.Add(wd, new
Dictionary<LectureTime,
List<TimetableItemProxy>>());
                        foreach  (LectureTime
lt in mTimesAll[gn])
                        {

tMainData[wd].Add(lt,                        new
List<TimetableItemProxy>());
                        }
                    }

                    mFullList.Sort((f1,   f2)
=> f1.Day.CompareTo(f2.Day));
                    foreach    (var    e    in
mFullList)
                    {
                        if (e.Day.Cross.Count
== 0)
                        {

tMainData[e.Day][e.Time].Add(new
TimetableItemProxy(e));
                            foreach
(LectureTime lt in e.Time.Cross)
                            {
                                if
(!tMainData[e.Day][lt].Exists(f  =>  f.data  ==
e))
                                {

tMainData[e.Day][lt].Add(new
TimetableItemProxy(e, true));
                                }
                            }

                            foreach
(LectureTime lt in e.mAddTimes)
                            {
                                if   (lt   !=
null)
                                {
                                    if
(!tMainData[e.Day][lt].Exists(f  =>  f.data  ==
e))
                                    {

tMainData[e.Day][lt].Add(new
TimetableItemProxy(e));
                                    }
                                    foreach
(LectureTime ltt in lt.Cross)
                                    {
                                        if
(!tMainData[e.Day][ltt].Exists(f => f.data ==
e))
                                        {

tMainData[e.Day][ltt].Add(new
TimetableItemProxy(e, true));
                                        }
                                    }
                                }
                            }

                            foreach  (DayItem  de
in e.Day.Cross)
                            {

tMainData[de][e.Time].Add(new
TimetableItemProxy(e));
                                foreach
(LectureTime lt in e.Time.Cross)
                                {
                                    if
(!tMainData[de][lt].Exists(f => f.data == e))
                                    {

tMainData[de][lt].Add(new
TimetableItemProxy(e, true));
                                    }
                                }

                                foreach
(LectureTime lt in e.mAddTimes)
                                {
                                    if   (lt   !=
null)
                                    {
                                        if
(!tMainData[de][lt].Exists(f => f.data == e))
                                        {

tMainData[de][lt].Add(new
TimetableItemProxy(e));
                                        }
```

```
                                    foreach
(LectureTime ltt in lt.Cross)
                                    {
                                        if
(!tMainData[de][ltt].Exists(f  =>  f.data  ==
e))
                                        {

tMainData[de][ltt].Add(new
TimetableItemProxy(e, true));
                                        }
                                    }
                                }
                            }
                        }
                    }
                    return tMainData;
                }
            }
            private           List<TimetableItem>
mFullList;
            /// <summary>
            /// Число элементов
            /// </summary>
            private int mCount;
            /// <summary>
            /// Время занятий
            /// </summary>
            private  static  Dictionary<Generator,
List<LectureTime>>     mTimesAll     =     new
Dictionary<Generator, List<LectureTime>>();
            /// <summary>
            /// Кэш расчетов ограничений
            /// </summary>
            private   Dictionary<object,   object>
mCacheRestrict;
            private           Dictionary<DayItem,
Dictionary<object,                   object>>
mCacheRestrictWeekDay;
            /// <summary>
            /// Получить кэш огранчения
            /// </summary>
            /// <typeparam name="T">Тип данных в
кэше</typeparam>
            /// <param name="type">Тип</param>
            /// <returns>Значение кэша или null
если кэша нет</returns>
            public  T  GetCacheRestrict<T>(object
type) where T: class
            {
                return
!mCacheRestrict.ContainsKey(type)  ?  null  :
(T)mCacheRestrict[type];
            }
            /// <summary>
            /// Установка кэша
            /// </summary>
            /// <param name="type">Тип</param>
            ///    <param    name="value">Значение
кэша</param>
            public  void  SetCacheRestrict(object
type, object value)
            {
                mCacheRestrict[type] = value;
            }
            public  T  GetCacheWeekDay<T>(DayItem
wd, object type) where T: class
            {
                return
!mCacheRestrictWeekDay[wd].ContainsKey(type)
? null : (T)mCacheRestrictWeekDay[wd][type];
            }
            public  void  SetCacheWeekDay(DayItem
wd, object type, object value)
            {
                mCacheRestrictWeekDay[wd][type] =
value;
            }
            /// <summary>
            /// Конструктор
            /// </summary>
            ///                             <param
name="generator">Генератор</param>
            public     TimetableSubject(Generator
generator)
            {
                gn = generator;
                mFullList          =          new
List<TimetableItem>();
                mCount = 0;
                lock (mTimesAll)
                {
                    if
(!mTimesAll.ContainsKey(generator))
                    {
                        mTimesAll.Add(generator,
generator.Times.Values.OrderBy(f           =>
f.NumberLecture).ToList());
                    }
                }

                mCacheRestrict       =       new
Dictionary<object, object>();
                mCacheRestrictWeekDay    =    new
Dictionary<DayItem,        Dictionary<object,
object>>();
                foreach(DayItem       wd       in
generator.WeekDays)
                {
                    mCacheRestrictWeekDay.Add(wd,
new Dictionary<object, object>());
                }
            }
            /// <summary>
            /// Получить расписание в указанный
день недели и время
            /// </summary>
            ///    <param    name="weekDay">День
недели</param>
            /// <param name="lt">Время</param>
            /// <returns>Расписание</returns>
            public       List<TimetableItemProxy>
ByDayTime(DayItem weekDay, LectureTime lt)
            {
                if
(MainData.ContainsKey(weekDay))
                {
                    return MainData[weekDay][lt];
                }
                return emptyList;
            }
            /// <summary>
            /// Имеется ли занятие в указанный
день недели и время
            /// </summary>
            ///    <param    name="weekDay">День
недели</param>
            ///      <param      name="lt">Время
занятия</param>
            ///    <returns>Имеется    или    нет
занятие</returns>
            public     bool     IsDayTime(DayItem
weekDay, LectureTime lt)
            {
                return
MainData[weekDay][lt].Count > 0;
            }
            /// <summary>
            /// Пустое расписание
            /// </summary>
            public bool IsEmpty
            {
                get { return mCount == 0; }
            }
            /// <summary>
            /// Все элементы расписания
            /// </summary>
```

```
        public            Dictionary<DayItem,
Dictionary<LectureTime,
List<TimetableItemProxy>>> AllData
        {
            get { return MainData; }
        }
        public List<TimetableItem> AsList
        {
            get
            {
                return mFullList;
            }
        }
        internal void ClearItem(TimetableItem
item)
        {
            mFullList.Remove(item);
            tMainData = null;
            mCacheRestrict.Clear();

mCacheRestrictWeekDay[item.Day].Clear();
            foreach(DayItem       d       in
item.Day.Cross)
            {

mCacheRestrictWeekDay[d].Clear();
            }
        }
        internal  void  AddItem(TimetableItem
item)
        {
            if   ((item.Day   ==   null)   ||
(item.Time == null)) return;
            mFullList.Add(item);
            tMainData = null;
            mCacheRestrict.Clear();

mCacheRestrictWeekDay[item.Day].Clear();
            foreach     (DayItem     d     in
item.Day.Cross)
            {

mCacheRestrictWeekDay[d].Clear();
            }
        }
    }
}
```

Файл Generator.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using ektu.Timetable.Dict;
using ektu.Timetable.Restrict;
using ektu.Timetable.Flow;
using ektu.Timetable.Timetable;

namespace ektu.Timetable
{
    /// <summary>
    /// Генератор расписания
    /// </summary>
    public abstract class Generator
    {
        /// <summary>
        /// Хранение данных по назначению для
элемента расписания
        /// </summary>
        internal class DayTimeRoom
        {
            /// <summary>
            /// Данные
            /// </summary>
            private            Tuple<DayItem,
LectureTime, Room> data;
            /// <summary>
            /// Конструктор
            /// </summary>
            ///                          <param
name="day">День</param>
            ///                          <param
name="time">Время</param>
            ///                          <param
name="room">Аудитория</param>
            internal DayTimeRoom(DayItem day,
LectureTime time, Room room)
            {
                data   =   new   Tuple<DayItem,
LectureTime, Room>(day, time, room);
            }
            /// <summary>
            /// День
            /// </summary>
            internal    DayItem    Day    =>
data.Item1;
            /// <summary>
            /// Время
            /// </summary>
            internal   LectureTime   Time   =>
data.Item2;
            /// <summary>
            /// Аудитория
            /// </summary>
            internal Room Room => data.Item3;
        }

        public          delegate          void
InfoActionHandler(string message);
        public     event     InfoActionHandler
InfoAction;
        public   void   SendInfoAction(string
info)
        {
            InfoAction?.Invoke(info);
        }

        public          delegate          void
StepActionHandler(int step);
        public     event     StepActionHandler
StartStep;

        public          delegate          bool
InfoTimetable(List<TimetableData>
timetables);
        public       event       InfoTimetable
InitialInfo;
        public       event       InfoTimetable
FinishExecute;
        public       event       InfoTimetable
FinishAlive;

        /// <summary>
        /// Дни недели
        /// </summary>
        public     abstract     List<DayItem>
WeekDays { get; }

        /// <summary>
        /// Список дисциплин
        /// </summary>
        public                          abstract
IReadOnlyDictionary<int,           Discipline>
Disciplines { get; }

        /// <summary>
        /// Список групп
        /// </summary>
        public                          abstract
IReadOnlyDictionary<int, Group> Groups { get;
}

        /// <summary>
        /// Список времени занятий
```

```
        /// </summary>
        public                          abstract
IReadOnlyDictionary<int, LectureTime> Times {
get; }

        /// <summary>
        /// Список аудиторий
        /// </summary>
        public                          abstract
IReadOnlyDictionary<int, Room> Rooms { get; }

        /// <summary>
        /// Список ППС
        /// </summary>
        public                          abstract
IReadOnlyDictionary<int, Teacher> Teachers {
get; }

        /// <summary>
        /// Список потоков
        /// </summary>
        public abstract List<FlowData> Flows
{ get; }

        /// <summary>
        /// Список начальных расписаний
        /// </summary>
        public                    List<string>
StartTimetablesFiles { get; private set; }

        /// <summary>
        /// Список "мягких" ограничений
        /// </summary>
        public                          abstract
IReadOnlyList<IRestrict> Soft { get; }

        /// <summary>
        /// Список "Жестких" ограничений
        /// </summary>
        public                          abstract
IReadOnlyList<IRestrict> Hard { get; }

        /// <summary>
        /// Функция сортироки расписаний
        /// </summary>
        ///    <param    name="t1">Расписание
1</param>
        ///    <param    name="t2">Расписание
2</param>
        /// <returns>Порядок</returns>
        private                               int
SortTimetables(TimetableData                t1,
TimetableData t2)
        {
            double v = t2.HardRestrictValue -
t1.HardRestrictValue;
            if (v > 0) return 1;
            if (v < 0) return -1;
            v     =     t2.RestrictValue     -
t1.RestrictValue;
            if (v > 0) return 1;
            if (v < 0) return -1;
            v    =    t1.Defected().Sum(f    =>
f.DefectedCount)   -   t2.Defected().Sum(f   =>
f.DefectedCount);
            if (v > 0) return 1;
            if (v < 0) return -1;
            return       t1.Generation       -
t2.Generation;
        }

        /// <summary>
        /// Вероятность выбора расписания из
списка для мутации
        /// </summary>
        public double MutationSelectPercent {
get; set; }

        /// <summary>
        ///   Вероятность   скрешивания   двух
расписаний
        /// </summary>
        public double CrossingSelectPercent {
get; set; }

        /// <summary>
        /// Начальное число расписаний
        /// </summary>
        public int CountTimetable { get; set;
}

        /// <summary>
        ///   Количество   расписаний   после
каждого шага
        /// </summary>
        public int CountTimetableCalc { get;
set; }

        /// <summary>
        /// Значение целевой функции
        /// </summary>
        public  double  FitnessValue  {  get;
set; }

        /// <summary>
        /// Исключение для элементов
        /// </summary>
        public Excluder Excluder { get; set;
}

        /// <summary>
        /// Конструктор
        /// </summary>
        public Generator() : base()
        {
            StartTimetablesFiles     =     new
List<string>();
            MutationSelectPercent = 0.8;
            CrossingSelectPercent = 0.1;
            CountTimetable = 10;
            FitnessValue = 0;
            Excluder = new Excluder();
            Crosser           =           new
TimetableDataCrosser();
            Mutator           =           new
TimetableDataMutator();
            Initiator          =          new
TimetableDataInitial();
        }

        /// <summary>
        /// Генератор начального расписания
        /// </summary>
        public          ITimetableDataInitial
Initiator { get; set; }

        /// <summary>
        ///   Объект   реализующий   мутации
расписания
        /// </summary>
        public ITimetableDataMutator Mutator
{ get; set; }

        /// <summary>
        /// Выполнение мутаций
        /// </summary>
        ///  <param  name="timetables">Список
расписаний</param>
        private                             void
MutateHardExec(List<TimetableData>
timetables)
        {
            Random rd = new Random();
            List<Task>     tsks     =     new
List<Task>();
            foreach (TimetableData table in
timetables.ToArray())
            {
```

```
                if     (rd.NextDouble()     <=
MutationSelectPercent)
                {
                    Task tsk = new Task(
                        tb =>
                        {
                            try
                            {
                                string info;
                                TimetableData
n  =  Mutator.Execute((TimetableData)tb,  out
info);
                                if    (n    !=
null)
                                {

InfoAction?.Invoke(" " + info);
                                    bool isEq
= false;
                                    lock
(timetables)
                                    {

foreach (TimetableData t in timetables)
                                        {

isEq = isEq || t.IsEqual(n);
                                        }
                                        if
(!isEq) timetables.Add(n);
                                    }
                                }
                            }
                            catch(Exception
ex)
                            {

Console.WriteLine(ex.Message    +    "\n"    +
ex.StackTrace);
                            }
                        }, table);
                    tsks.Add(tsk);
                    tsk.Start();
                }
            }
            Task.WaitAll(tsks.ToArray());
        }

        /// <summary>
        ///   Объект   реализующий   операцию
скрещивания
        /// </summary>
        public ITimetableDataCrosser Crosser
{ get; set; }

        /// <summary>
        /// Выполнение скрещиваний
        /// </summary>
        ///  <param  name="timetables">Список
расписаний</param>
        private                          void
CrossingExec(List<TimetableData> timetables)
        {
            Random rd = new Random();
            List<Task>     tsks     =     new
List<Task>();
            int nx = 1;
            for   (int   i   =   0;   i   <
timetables.Count - 1; i++)
            {
                for   (int   j   =   0;   j   <
timetables.Count; j++)
                {
                    if  ((rd.NextDouble()  <=
CrossingSelectPercent) && (i != j)/* &&

(timetables[i].RestrictValue                >
timetables[j].RestrictValue)*/)
                    {
                        Task tsk = new Task(
                            f =>
                            {
                                try
                                {

Tuple<TimetableData, TimetableData, int> d =
(Tuple<TimetableData, TimetableData, int>)f;
                                    string
info;

TimetableData  n  =  Crosser.Execute(d.Item1,
d.Item2, out info);

InfoAction?.Invoke(" " + info);
                                    if  (n  !=
null)
                                    {
                                        bool
isEq = false;
                                        lock
(timetables)
                                        {

foreach (TimetableData t in timetables)
                                            {

isEq = isEq || t.IsEqual(n);
                                            }

if (!isEq) timetables.Add(n);
                                        }
                                    }
                                }

catch(Exception ex)
                                {

Console.WriteLine(ex.Message);
                                }
                            },
                            new
Tuple<TimetableData,           TimetableData,
int>(timetables[i], timetables[j], nx++));
                        tsks.Add(tsk);
                    }
                }
            }
            tsks.ForEach(f => f.Start());
            Task.WaitAll(tsks.ToArray());
            Console.WriteLine();
        }

        /// <summary>
        /// Генерация расписания
        /// </summary>
        ///   <param   name="schedule">Базовое
расписание</param>
        ///          <returns>Сгенерированное
расписание</returns>
        public     virtual     TimetableData
GenerateTimetable(TimetableData schedule)
        {
            InfoAction?.Invoke("Начало
генерации расписания");
            List<TimetableData> timetables =
new List<TimetableData>();
            List<Task>     tsks     =     new
List<Task>();

            if (StartTimetablesFiles.Count ==
0)
            {
                InfoAction?.Invoke("Генерация
начальной популяции");
                for   (int   i   =   0;   i   <
CountTimetable; i++)
                {
```

```
                    tsks.Add(new Task((n) =>
{
                        try
                        {
                            TimetableData
items       =       Initiator.Generate(this,
n.ToString(), schedule);

timetables.Add(items);
                        }
                        catch { }
                    }, i + 1));
                    tsks[tsks.Count         -
1].Start();
                }
                Task.WaitAll(tsks.ToArray());
                tsks.Clear();
                InfoAction?.Invoke("Генерация
начальной популяции завершена");
            }
            else
            {
                InfoAction?.Invoke("Загрузка
начальной популяции");
                foreach    (string    fn    in
StartTimetablesFiles)
                {
                    timetables.Add(new
TimetableData(this, fn));

timetables[timetables.Count                 -
1].UpdateRestrictValues(Hard, Soft);
                }

timetables[0].UpdateRestrictValues(Hard,
Soft);
                foreach     (var     v     in
timetables[0].vals)
                {
                    InfoAction?.Invoke(v.Key
+ ": " + v.Value.ToString());
                }

                InfoAction?.Invoke("Загрузка
начальной популяции завершена");
            }
            InitialInfo?.Invoke(timetables);

            bool isEndFind = false;
            double  cHard  =  Hard.Sum(f  =>
f.Factor);
            InfoAction?.Invoke("Начало  цикла
поиска решения");
            int stepNumber = 0;
            double valEnd = 0;
            int valCount = 0;
            do
            {
                stepNumber++;
                if (!isEndFind)
                {

StartStep?.Invoke(stepNumber);

                    // Мутации

InfoAction?.Invoke("Мутации");

MutateHardExec(timetables);

                    // Скрещивания

InfoAction?.Invoke("Скрещивания");
                    CrossingExec(timetables);

                    // Проверка завершения

InfoAction?.Invoke("Заверешние   мутаций   и
скрещивания");
                    foreach    (TimetableData
table in timetables)
                    {
                        isEndFind = isEndFind
||

((table.HardRestrictValue == cHard) &&

(table.RestrictValue >= FitnessValue));
                        table.Generation++;
                    }

                    //  Определяем  выживщие
особи

InfoAction?.Invoke("Определяем      выживание
особей");

timetables.Sort(SortTimetables);

FinishExecute?.Invoke(timetables);
                    Random rd = new Random();
                    List<TimetableData>  dt  =
new List<TimetableData>();
                    dt.Add(timetables[0]);
                    timetables.RemoveAt(0);
                    foreach(IRestrict
restrict in Hard)
                    {
                        TimetableData   df   =
timetables
                            .Where(f       =>
f.GetRestrictItemValue(restrict)            >
dt[0].GetRestrictItemValue(restrict))
                            .OrderBy(f     =>
rd.NextDouble())

.FirstOrDefault();
                        if (df != null)
                        {
                            dt.Add(df);

timetables.Remove(df);
                        }
                    }
                    foreach        (IRestrict
restrict in Soft)
                    {
                        TimetableData   df   =
timetables
                            .Where(f       =>
f.GetRestrictItemValue(restrict)            >
dt[0].GetRestrictItemValue(restrict))
                            .OrderBy(f     =>
rd.NextDouble())

.FirstOrDefault();
                        if (df != null)
                        {
                            dt.Add(df);

timetables.Remove(df);
                        }
                    }
                    while (timetables.Count +
dt.Count - CountTimetableCalc > 0)
                    {
                        if  (timetables.Count
== 0) break;
                        try
                        {

timetables.RemoveAt(rd.Next(0,
timetables.Count));
                        }
                        catch { }
                    }
                    dt.AddRange(timetables);
                    timetables = dt;
```

```
timetables.Sort(SortTimetables);

                    if
(timetables[0].RestrictValue != valEnd)
                    {
                        valEnd              =
timetables[0].RestrictValue;
                        valCount = 1;
                    }
                    else
                    {
                        valCount++;
                        if
((timetables[0].HardRestrictValue          ==
Hard.Sum(f       =>       f.Factor))       &&
(timetables[0].RestrictValue               >=
FitnessValue))
                        {
                            isEndFind = true;
                        }

                    }
                    bool?         r         =
FinishAlive?.Invoke(timetables);
                    if    ((r.HasValue)    &&
r.Value)
                    {
                        isEndFind = true;
                    }
                    GC.Collect();
                }
            } while (!isEndFind);
            timetables.Sort(SortTimetables);
            return timetables[0];
        }

    }
}
```

Файл ektu.Timetable.csproj

```
<?xml version="1.0" encoding="utf-8"?>
<Project                  ToolsVersion="14.0"
DefaultTargets="Build"
xmlns="http://schemas.microsoft.com/developer
/msbuild/2003">
  <Import
Project="$(MSBuildExtensionsPath)\$(MSBuildTo
olsVersion)\Microsoft.Common.props"
Condition="Exists('$(MSBuildExtensionsPath)\$
(MSBuildToolsVersion)\Microsoft.Common.props'
)" />
  <PropertyGroup>
    <Configuration               Condition="
'$(Configuration)'           ==           ''
">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == ''
">AnyCPU</Platform>
    <ProjectGuid>{924363D5-6BC2-45C7-8AE3-
F472842B36EE}</ProjectGuid>
    <OutputType>Library</OutputType>

<AppDesignerFolder>Properties</AppDesignerFol
der>

<RootNamespace>ektu.Timetable</RootNamespace>

<AssemblyName>ektu.Timetable</AssemblyName>

<TargetFrameworkVersion>v4.6</TargetFramework
Version>
    <FileAlignment>512</FileAlignment>
  </PropertyGroup>
  <PropertyGroup                  Condition="
'$(Configuration)|$(Platform)'             ==
'Debug|AnyCPU' ">
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <Optimize>false</Optimize>
    <OutputPath>bin\Debug\</OutputPath>

<DefineConstants>DEBUG;TRACE</DefineConstants
>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <PropertyGroup                  Condition="
'$(Configuration)|$(Platform)'             ==
'Release|AnyCPU' ">
    <DebugType>pdbonly</DebugType>
    <Optimize>true</Optimize>
    <OutputPath>bin\Release\</OutputPath>
    <DefineConstants>TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
    <PlatformTarget>x64</PlatformTarget>
  </PropertyGroup>
  <PropertyGroup
Condition="'$(Configuration)|$(Platform)'  ==
'Test1|AnyCPU'">
    <DebugSymbols>true</DebugSymbols>
    <OutputPath>bin\Test1\</OutputPath>

<DefineConstants>DEBUG;TRACE</DefineConstants
>
    <DebugType>full</DebugType>
    <PlatformTarget>AnyCPU</PlatformTarget>
    <ErrorReport>prompt</ErrorReport>

<CodeAnalysisRuleSet>MinimumRecommendedRules.
ruleset</CodeAnalysisRuleSet>
  </PropertyGroup>
  <ItemGroup>
    <Reference Include="System" />
    <Reference Include="System.Core" />
    <Reference Include="System.Xml.Linq" />
    <Reference
Include="System.Data.DataSetExtensions" />
    <Reference Include="Microsoft.CSharp" />
    <Reference Include="System.Data" />
    <Reference Include="System.Net.Http" />
    <Reference Include="System.Xml" />
  </ItemGroup>
  <ItemGroup>
    <Compile Include="Dict\Item.cs" />
    <Compile Include="Dict\Discipline.cs" />
    <Compile Include="Dict\WeekDayItem.cs" />
    <Compile Include="Flow\FlowData.cs" />
    <Compile Include="Flow\FlowItem.cs" />
    <Compile Include="Dict\Group.cs" />
    <Compile
Include="Flow\ForelClassificator.cs" />
    <Compile  Include="Timetable\Excluder.cs"
/>
    <Compile Include="Generator.cs" />
    <Compile
Include="Properties\AssemblyInfo.cs" />
    <Compile Include="Dict\Room.cs" />
    <Compile  Include="Restrict\IRestrict.cs"
/>
    <Compile
Include="Restrict\RestrictDayDiscipline.cs"
/>
    <Compile
Include="Restrict\RestrictEmployee.cs" />
    <Compile
Include="Restrict\RestrictEmployeeFill.cs" />
    <Compile
Include="Restrict\RestrictFollowNextPred.cs"
/>
    <Compile
Include="Restrict\RestrictGroup.cs" />
    <Compile
Include="Restrict\RestrictGroupFill.cs" />
    <Compile
Include="Restrict\RestrictEmployeeHours.cs"
/>
```

```
    <Compile
Include="Restrict\RestrictGroupHours.cs" />
    <Compile
Include="Restrict\RestrictDiscDays.cs" />
    <Compile
Include="Restrict\RestrictDayExam.cs" />
    <Compile
Include="Restrict\RestrictLectureBefore.cs"
/>
    <Compile
Include="Restrict\RestrictLecturesAfter.cs"
/>
    <Compile
Include="Restrict\RestrictOneDayDisciplines.c
s" />
    <Compile
Include="Restrict\RestrictRoom.cs" />
    <Compile
Include="Timetable\TimetableData.cs" />
    <Compile
Include="Timetable\TimetableDataCrosser.cs"
/>
    <Compile
Include="Timetable\TimetableDataInitial.cs"
/>
    <Compile
Include="Timetable\TimetableDataMutator.cs"
/>
    <Compile
Include="Timetable\TimetableItem.cs" />
    <Compile Include="Dict\Teacher.cs" />
    <Compile Include="Dict\LectureTime.cs" />
    <Compile
Include="Timetable\TimetableOptimalFinder.cs"
/>
    <Compile
Include="Timetable\TimetableSubject.cs" />
  </ItemGroup>
  <ItemGroup>
    <None  Include="ClassDiagram_Restrict.cd"
/>
    <None Include="ClassDiagram_Timetable.cd"
/>
    <None Include="ClassDiagram_Flow.cd" />
    <None Include="ClassDiagram_Dict.cd" />
  </ItemGroup>
  <Import
Project="$(MSBuildToolsPath)\Microsoft.CSharp
.targets" />
  <!-- To modify your build process, add your
task inside one of the targets below and
uncomment it.
       Other similar extension points exist,
see Microsoft.Common.targets.
  <Target Name="BeforeBuild">
  </Target>
  <Target Name="AfterBuild">
  </Target>
  -->
</Project>
```

Приложение Г – Исходный код шаблона консольного приложения для генерации расписания занятий

Файл TimetableTemplate.csproj

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="14.0"
DefaultTargets="Build"
xmlns="http://schemas.microsoft.com/developer
/msbuild/2003">
  <Import
Project="$(MSBuildExtensionsPath)\$(MSBuildTo
olsVersion)\Microsoft.Common.props"
Condition="Exists('$(MSBuildExtensionsPath)\$
(MSBuildToolsVersion)\Microsoft.Common.props'
)" />
  <PropertyGroup>
    <Configuration Condition="
'$(Configuration)' == ''
">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == ''
">AnyCPU</Platform>
    <ProjectGuid>{B9854533-3634-40B3-B828-
BDA9976EF75D}</ProjectGuid>
    <OutputType>Exe</OutputType>

<AppDesignerFolder>Properties</AppDesignerFol
der>

<RootNamespace>TimetableTemplate</RootNamespa
ce>

<AssemblyName>TimetableTemplate</AssemblyName
>

<TargetFrameworkVersion>v4.6</TargetFramework
Version>
    <FileAlignment>512</FileAlignment>

<AutoGenerateBindingRedirects>true</AutoGener
ateBindingRedirects>
  </PropertyGroup>
  <PropertyGroup Condition="
'$(Configuration)|$(Platform)' ==
'Debug|AnyCPU' ">
    <PlatformTarget>x64</PlatformTarget>
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <Optimize>false</Optimize>
    <OutputPath>bin\Debug\</OutputPath>

<DefineConstants>DEBUG;TRACE</DefineConstants
>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
    <Prefer32Bit>false</Prefer32Bit>
  </PropertyGroup>
  <PropertyGroup Condition="
'$(Configuration)|$(Platform)' ==
'Release|AnyCPU' ">
    <PlatformTarget>x64</PlatformTarget>
    <DebugType>pdbonly</DebugType>
    <Optimize>true</Optimize>
    <OutputPath>bin\Release\</OutputPath>
    <DefineConstants>TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <ItemGroup>
    <Reference Include="System" />
    <Reference Include="System.Configuration"
/>
    <Reference Include="System.Core" />
    <Reference
Include="System.Web.Extensions" />
    <Reference Include="System.Xml.Linq" />
    <Reference
Include="System.Data.DataSetExtensions" />
    <Reference Include="Microsoft.CSharp" />
    <Reference Include="System.Data" />
    <Reference Include="System.Net.Http" />
    <Reference Include="System.Xml" />
  </ItemGroup>
  <ItemGroup>
    <Compile Include="DayTimeSetting.cs" />
    <Compile Include="GeneratorImpl.cs" />
    <Compile Include="MyExcluder.cs" />
    <Compile Include="Program.cs" />
    <Compile
Include="Properties\AssemblyInfo.cs" />
  </ItemGroup>
  <ItemGroup>
    <None Include="App.config" />
  </ItemGroup>
  <ItemGroup>
    <ProjectReference
Include="..\ektu.Timetable\ektu.Timetable.csp
roj">
      <Project>{924363d5-6bc2-45c7-8ae3-
f472842b36ee}</Project>
      <Name>ektu.Timetable</Name>
    </ProjectReference>
  </ItemGroup>
  <ItemGroup />
  <ItemGroup>
    <Content Include="Viewer.html" />
  </ItemGroup>
  <Import
Project="$(MSBuildToolsPath)\Microsoft.CSharp
.targets" />
  <!-- To modify your build process, add your
task inside one of the targets below and
uncomment it.
       Other similar extension points exist,
see Microsoft.Common.targets.
  <Target Name="BeforeBuild">
  </Target>
  <Target Name="AfterBuild">
  </Target>
  -->
</Project>
```

Файл Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.IO;
using ektu.Timetable;
using ektu.Timetable.Timetable;
using System.Dynamic;
using ektu.Timetable.Restrict;

namespace TimetableTemplate
{
    class Program
    {
        private static void
OutTables(List<TimetableData> tables)
        {
            int num = 1;
```

```
            const string st1 =
"┌──────┬──────────────────┬──────────────────┬──────────
──────┬──────────┬──────────┬──────────────────
──────┐";
            const string st2 =
"├──────┼──────────────────┼──────────────────┼──────────
──────┼──────────┼──────────┼──────────────────
──────┤";
            const string st3 =
"└──────┴──────────────────┴──────────────────┴──────────
──────┴──────────┴──────────┴──────────────────
──────┘";
            Console.WriteLine(st1);

Console.WriteLine(String.Format("│ {0,-3} │
{1,-13} │ {2,-12} │ {3,-12} │ {4,-6} │ {5,-
10} │ {6,-16} │",
                "num", "value", "hard",
"soft", "defect", "generation", "type"));
            Console.WriteLine(st2);
            foreach (TimetableData t in
tables)
            {

                Console.WriteLine(
                    String.Format("│{0,4} │
{1,-11:0.0000000} │ {2,-11:0.000000} │ {3,-
11:0.0000000} │ {4,6} │ {5,10} │ {6,-16} │",
                    num++, t.RestrictValue,
t.HardRestrictValue, t.SoftRestrictValue,
t.Defected().Sum(f => f.DefectedCount),
t.Generation, t.Type.ToString()));
            }
            Console.WriteLine(st3);
        }

        public static void Main(string[]
args)
        {
            Generator gen = new
GeneratorImpl();

            #region Настройка логирования
            int stepNumber = 0;
            string path =
DateTime.Now.ToString("yyyyMMdd_HHmm") +
"\\";
            Directory.CreateDirectory(path);
            gen.StartStep +=
                num =>
                {
                    Console.WriteLine("Начало
шага №" + num.ToString());
                    stepNumber = num;
                };
            gen.InitialInfo +=
                t =>
                {
                    int id = 1;
                    foreach (TimetableData
item in t)
                    {

item.GetXml().Save(path + "Start_" +
(id++).ToString() + ".xml");
                    }
                    OutTables(t);
                    return false;
                };
            gen.FinishExecute +=
                t =>
                {
                    Console.WriteLine("
Заверешние мутаций и скрещивания");
                    Console.WriteLine("
Всего особей: " + t.Count.ToString());
                    OutTables(t);
                    return false;
                };
            gen.FinishAlive +=
                t =>
                {
                    Console.WriteLine("
Выжило особей: " + t.Count.ToString());
                    OutTables(t);
                    string np = path +
stepNumber.ToString() + "\\";

Directory.CreateDirectory(np);
                    for (int i = 0; i <
t.Count; i++)
                    {
                        t[i].GetXml().Save(np
+ (i + 1).ToString() + ".xml");
                    }
                    return false;
                };
            gen.InfoAction += msg =>
Console.WriteLine(msg);
            #endregion

            // TODO: Задать размер начальной
популяции
            gen.CountTimetable = 10;
            // TODO: Задать размер популяции
            gen.CountTimetableCalc = 20;
            gen.FitnessValue = gen.Hard.Sum(f
=> f.Factor) + gen.Soft.Sum(f => f.Factor) *
2;
            gen.Excluder = new MyExcluder();

            TimetableData d =
gen.GenerateTimetable(null);

        }
    }
}
```

Файл GenratorImpl.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable;
using ektu.Timetable.Dict;
using ektu.Timetable.Restrict;
using ektu.Timetable.Flow;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
using System.Xml;

namespace TimetableTemplate
{
    /// <summary>
    /// Реализация класса генератора
расписания
    /// </summary>
    internal class GeneratorImpl : Generator
    {
        private Dictionary<int, TypeFlowBase>
mFlowTypes;
        /// <summary>
```

```
        /// Временные интервалы
        /// </summary>
        private Dictionary<int, LectureTime>
mTimes;
        /// <summary>
        /// Группы
        /// </summary>
        private Dictionary<int, Group>
mGroups;
        /// <summary>
        /// Аудитории
        /// </summary>
        private Dictionary<int, Room> mRooms;
        /// <summary>
        /// ППС
        /// </summary>
        private Dictionary<int, Teacher>
mTeachers;
        /// <summary>
        /// Дисциплины
        /// </summary>
        private Dictionary<int, Discipline>
mDisciplines;
        /// <summary>
        /// Учебные потоки
        /// </summary>
        private List<FlowData> mFlows;
        /// <summary>
        /// Жесткие ограничения
        /// </summary>
        private List<IRestrict> mHard;
        /// <summary>
        /// Мягкие ограничения
        /// </summary>
        private List<IRestrict> mSoft;
        /// <summary>
        /// Дни
        /// </summary>
        private List<DayItem> mWeekDays;
        public override List<DayItem>
WeekDays
        {
            get { return mWeekDays; }
        }
        public RestrictFollowNextPred resNext
{ get; private set; }

        /// <summary>
        /// Выполнение чтения данных из БД
        /// </summary>
        /// <param name="con">Соединение с
БД</param>
        /// <param name="cmd">Строка
запроса</param>
        /// <param name="action">Процедура
чтения</param>
        private void ReadDB(SqlConnection
con, string cmd, Action<IDataRecord> action)
        {
            SqlCommand cmds =
con.CreateCommand();
            cmds.CommandText = cmd;
            cmds.CommandTimeout = 3600;
            using (SqlDataReader reader =
cmds.ExecuteReader())
            {
                while (reader.Read())
                {
                    action(reader);
                }
            }
        }

        /// <summary>
        /// Чтение временных интервалов
        /// </summary>
        /// <param
name="connection">Подключение к БД</param>
        /// <param name="ManagerID">ИД
расписания</param>
        private void InitTimes(SqlConnection
connection, int ManagerID)
        {
            mTimes = new Dictionary<int,
LectureTime>();
            const string ltSql = "SELECT *
FROM Timetable.LectureTime LT WHERE
LT.ManagerID = {0} ORDER BY
LT.NumberLecture";
            ReadDB(connection,
String.Format(ltSql, ManagerID),
                r =>
                {
                    LectureTime lt = new
LectureTime();
                    lt["LectureTimeID"] =
r["LectureTimeID"];
                    lt["StartTime"] =
r["StartTime"];
                    lt["FinishTime"] =
r["FinishTime"];
                    lt["NumberLecture"] =
r["NumberLecture"];

mTimes.Add(lt.LectureTimeID, lt);
                });

            LectureTime next = null;
            foreach (LectureTime lt in
Times.Values.OrderByDescending(f =>
f.NumberLecture))
            {
                lt.Next = next;
                next = lt;
            }

            foreach (LectureTime lt in
Times.Values)
            {

                lt.Cross = Times.Where(
                    f => (((f.Value.StartTime
>= lt.StartTime) &&
                           (f.Value.StartTime
<= lt.FinishTime)) ||
                          ((f.Value.FinishTime
>= lt.StartTime) &&
                           (f.Value.FinishTime
<= lt.FinishTime))) && (f.Value != lt)
                ).Select(f =>
f.Value).ToList();
            }
        }

        /// <summary>
        /// Чтение данные по группам
        /// </summary>
        /// <param
name="connection">Подключение к БД</param>
        /// <param name="ManagerID">ИД
расписания</param>
        private void InitGroups(SqlConnection
connection, int ManagerID)
        {
            mGroups = new Dictionary<int,
Group>();
```

```
            const string grSql = "SELECT *
FROM Timetable.Groups G WHERE G.ManagerID =
{0}";
            const string grSqlAttr = "SELECT
Name, Value FROM Timetable.GroupAttributes
WHERE TimetableGroupID = {0}";
            ReadDB(connection,
String.Format(grSql, ManagerID),
                r =>
                {
                    Group gr = new Group();
                    gr["GroupID"] =
r["GroupID"];
                    gr["Code"] = r["Code"];
                    ReadDB(connection,
String.Format(grSqlAttr,
r["TimetableGroupID"]),
                        rs =>
                        {
                            string name =
(string)rs[0];
                            if
(gr.HasAttribute(name))
                            {
                                if
(!(gr[name] is IList<string>))
                                {

List<string> lst = new List<string>();

lst.Add((string)gr[name]);
                                    gr[name]
= lst;
                                }
                                (gr[name] as
List<string>).Add((string)rs[1]);
                            }
                            else
                            {
                                gr[name] =
rs[1];
                            }
                        });
                    mGroups.Add(gr.GroupID,
gr);
                });
        }

        /// <summary>
        /// Чтение данных по аудиториям
        /// </summary>
        /// <param
name="connection">Поключение к БД</param>
        /// <param name="ManagerID">ИД
расписания</param>
        private void InitRooms(SqlConnection
connection, int ManagerID)
        {
            mRooms = new Dictionary<int,
Room>();
            const string sqlRoom = "SELECT *
FROM Timetable.Rooms R WHERE R.ManagerID =
{0}";
            const string sqlRoomAttr =
"SELECT Name, Value FROM
Timetable.RoomAttributes WHERE
TimetableRoomID = {0}";
            ReadDB(connection,
String.Format(sqlRoom, ManagerID),
                r =>
                {
                    Room rm = new Room();
                    rm["RoomID"] =
r["RoomID"];
                    rm["Name"] = r["Name"];
                    rm["Places"] =
r["Places"];
                    rm["RoomTypeID"] =
r["RoomTypeID"];
                    ReadDB(connection,
String.Format(sqlRoomAttr,
r["TimetableRoomID"]),
                        rs =>
                        {
                            string name =
(string)rs[0];
                            if
(rm.HasAttribute(name))
                            {
                                if
(!(rm[name] is IList<string>))
                                {

List<string> lst = new List<string>();

lst.Add((string)rm[name]);
                                    rm[name]
= lst;
                                }
                                (rm[name] as
List<string>).Add((string)rs[1]);
                            }
                            else
                            {
                                rm[name] =
rs[1];
                            }
                        });
                    mRooms.Add(rm.RoomID,
rm);
                });
        }

        /// <summary>
        /// Чтение данных по ППС
        /// </summary>
        /// <param
name="connection">Подключение к БД</param>
        /// <param name="ManagerID">ИД
расписания</param>
        private void
InitTeachers(SqlConnection connection, int
ManagerID)
        {
            mTeachers = new Dictionary<int,
Teacher>();
            const string sqlTeacher = "SELECT
* FROM Timetable.Teachers WHERE ManagerID =
{0}";
            const string sqlTeacherAttr =
"SELECT Name, Value FROM
Timetable.TeacherAttributes WHERE
TimetableTeacherID = {0}";
            ReadDB(connection,
String.Format(sqlTeacher, ManagerID),
                r =>
                {
                    Teacher th = new
Teacher();
                    th["TeacherID"] =
r["TeacherID"];
                    th["Name"] = r["Name"];
                    ReadDB(connection,
String.Format(sqlTeacherAttr,
r["TimetableTeacherID"]),
```

```
                        rs =>
                        {
                            string name =
(string)rs[0];
                            if
(th.HasAttribute(name))
                            {
                                if
(!(th[name] is IList<string>))
                                {

List<string> lst = new List<string>();

lst.Add((string)th[name]);
                                    th[name]
= lst;
                                }
                                (th[name] as
List<string>).Add((string)rs[1]);
                            }
                            else
                            {
                                th[name] =
rs[1];
                            }
                        });

mTeachers.Add(th.TeacherID, th);
                });
        }

        /// <summary>
        /// Чтение данных по дням
        /// </summary>
        /// <param
name="connection">Подключение к БД</param>
        /// <param name="ManagerID">ИД
расписания</param>
        private void InitDays(SqlConnection
connection, int ManagerID)
        {
            mWeekDays = new List<DayItem>();

mWeekDays.AddRange(WeekDay2Item.Monday);

mWeekDays.AddRange(WeekDay2Item.Tuesday);

mWeekDays.AddRange(WeekDay2Item.Wednesday);

mWeekDays.AddRange(WeekDay2Item.Thursday);

mWeekDays.AddRange(WeekDay2Item.Friday);

mWeekDays.AddRange(WeekDay2Item.Saturday);
        }

        private void
InitTypeFlow(SqlConnection connection, int
ManagerID)
        {
            mFlowTypes = new Dictionary<int,
TypeFlowBase>();

mFlowTypes.Add(TypeFlowLecture.Item.ID,
TypeFlowLecture.Item);

mFlowTypes.Add(TypeFlowPractice.Item.ID,
TypeFlowPractice.Item);

mFlowTypes.Add(TypeFlowLaboratory.Item.ID,
TypeFlowLaboratory.Item);
        }

        /// <summary>
        /// Чтение данных по дисциплинам
        /// </summary>
        /// <param
name="connection">Подключение к БД</param>
        /// <param name="ManagerID">ИД
расписания</param>
        private void
InitDisciplines(SqlConnection connection, int
ManagerID)
        {
            mDisciplines = new
Dictionary<int, Discipline>();
            const string sqlDisc = "SELECT *
FROM Timetable.Discipline WHERE ManagerID =
{0}";
            ReadDB(connection,
String.Format(sqlDisc, ManagerID),
                r =>
                {
                    Discipline ds = new
Discipline();
                    ds["DisciplineID"] =
r["DisciplineID"];
                    ds["Name"] = r["Name"];

mDisciplines.Add(ds.DisciplineID, ds);
                });
        }

        /// <summary>
        /// Сортировка потоков
        /// </summary>
        private void SortFlows()
        {
            Dictionary<int, int> disc = new
Dictionary<int, int>();
            mFlows.ForEach(
                f =>
                {
                    int id = f.Items.Min(fx
=> fx.Discipline.DisciplineID);
                    if
(!disc.ContainsKey(id))
                    {
                        disc[id] = 0;
                    }
                    disc[id] +=
f.CountStudent;
                });

            ForelClassificator classificator
= new ForelClassificator(mFlows, 0.8, 0.1,
0.1);
            List<List<FlowData>> clusters =
classificator.GetClusters(0.5);
            mFlows.Clear();
            clusters.Sort((f1, f2) =>
f2.Sum(f => f.CountStudent) - f1.Sum(f =>
f.CountStudent));
            while (clusters.Count > 0)
            {

mFlows.AddRange(clusters[0].OrderByDescending
(f => f.Items.Sum(fx => fx.CountStudent)));
                clusters.RemoveAt(0);
            }
        }

        /// <summary>
        /// Чтение аудиторий для дисциплин
        /// </summary>
```

```
        /// <param
name="connection">Подключение к БД</param>
        /// <param name="ManagerID">ИД
расписания</param>
        private void
ReadDiscRooms(SqlConnection connection, int
ManagerID)
        {
            ReadDB(connection, "SELECT * FROM
Timetable.DisciplineRooms D WHERE D.ManagerID
= " + ManagerID.ToString(),
                r =>
                {
                    Discipline ds =
mDisciplines[(int)r["DisciplineID"]];
                    TypeFlowBase t =
mFlowTypes[(int)r["TypeLecture"]];
                    string val = null;
                    if (r["RoomTypeID"] !=
DBNull.Value)
                    {
                        val = "RoomTypeID=" +
r["RoomTypeID"].ToString() + "," +
r["PrioritySelect"].ToString();
                    }
                    if (r["RoomID"] !=
DBNull.Value)
                    {
                        val = "RoomID=" +
r["RoomID"].ToString() + "," +
r["PrioritySelect"].ToString();
                        if (r["TeacherID"] !=
DBNull.Value)
                        {
                            val +=
",TeacherID=" + r["TeacherID"].ToString();
                        }
                    }
                    if
(!ds.Rooms.ContainsKey(t))
                    {
                        ds.Rooms.Add(t, new
List<string>());
                    }
                    ds.Rooms[t].Add(val);
                });
        }

        /// <summary>
        /// Чтение потоков из БД
        /// </summary>
        /// <param
name="connection">Подключение к БД</param>
        /// <param name="ManagerID">ИД
расписания</param>
        private void ReadFlows(SqlConnection
connection, int ManagerID)
        {
            const string sqlFlow = "SELECT *
FROM Timetable.Flows WHERE ManagerID = {0}";
            const string sqlFlowAttr =
"SELECT Name, Value FROM
Timetable.FlowAttributes WHERE FlowID = {0}";
            const string sqlFlowTeacher =
"SELECT * FROM Timetable.FlowTeachers WHERE
FlowID = {0}";
            int fid = 0;
            ReadDB(connection,
String.Format(sqlFlow, ManagerID),
                r =>
                {
                    int FlowID =
(int)r["FlowID"];
                    int HoursCount =
(int)r["HoursCount"];
                    int CountElements =
(int)r["CountElements"];
                    TypeFlowBase type =
mFlowTypes[(int)r["TypeLecture"]];

                    // атрибуты потока
                    Dictionary<string,
object> attr = new Dictionary<string,
object>();
                    ReadDB(connection,
String.Format(sqlFlowAttr, FlowID),
                        rs =>
                        {
                            string name =
(string)rs[0];
                            if
(attr.ContainsKey(name))
                            {
                                if
(!(attr[name] is IList<string>))
                                {

List<string> lst = new List<string>();

lst.Add((string)attr["name"]);
                                }
                                (attr[name]
as List<string>).Add((string)rs[1]);
                            }
                            else
                            {
                                attr[name] =
rs[1];
                            }
                        });


                    // ППС
                    Dictionary<int,
List<int>> teachers = new Dictionary<int,
List<int>>();
                    ReadDB(connection,
String.Format(sqlFlowTeacher, FlowID),
                        rt =>
                        {
                            int s =
(int)rt["SubGroup"];
                            if
(!teachers.ContainsKey(s))
                            {

teachers.Add(s, new List<int>());
                            }

teachers[s].Add((int)rt["TeacherID"]);
                        });

                    // Группы
                    Dictionary<int,
List<FlowItem>> groups = new Dictionary<int,
List<FlowItem>>();
                    ReadDB(connection,
"SELECT * FROM Timetable.FlowGroups WHERE
FlowID = " + FlowID.ToString(),
                        rt =>
                        {
                            int s =
(int)rt["SubGroup"];
                            if
(!groups.ContainsKey(s))
                            {
```

```
                                groups.Add(s,
new List<FlowItem>());
                            }
                            if
(Groups.ContainsKey((int)rt["GroupID"]))
                            {
                                FlowItem item
= new FlowItem()
                                {
                                    SubGroup
= s,

CountStudent = (int)rt["CountStudent"],
                                    Group =
Groups[(int)rt["GroupID"]],

Discipline =
Disciplines[(int)rt["DisciplineID"]]
                                };
                                string sm =
(string)rt["SubGroupMask"];

item.SubGroupMask = new BitArray(sm.Length);
                                for(int i =
0; i < sm.Length; i++)
                                {

item.SubGroupMask.Set(i, sm[i] == '1');
                                }

groups[s].Add(item);
                            }
                        });


                    for (int i = 0; i <
CountElements; i++)
                    {
                        Dictionary<int, int>
teachersCount = new Dictionary<int, int>();
                        List<FlowData> flows
= new List<FlowData>();
                        foreach
(KeyValuePair<int, List<FlowItem>> gr in
groups)
                        {
                            FlowData flowData
= new FlowData()
                            {
                                FlowId =
(++fid).ToString(),
                                TypeFlow =
type,
                                CountHours =
HoursCount
                            };

foreach(KeyValuePair<string, object> v in
attr)
                            {

flowData.FlowParam.Add(v.Key, v.Value);
                            }

flowData.Items.AddRange(gr.Value);
                            if
(teachers.ContainsKey(gr.Key))
                            {
                                foreach (int
tid in teachers[gr.Key])
                                {

flowData.Teachers.Add(tid, Teachers[tid]);
                                    if
(!teachersCount.ContainsKey(tid))
                                    {

teachersCount.Add(tid, 0);
                                    }

teachersCount[tid]++;
                                }
                            }

mFlows.Add(flowData);

flows.Add(flowData);
                        }
                        if ((flows.Count > 1)
&& (teachersCount.All(f => f.Value == 1)))
                        {
                            foreach (FlowData
fd in flows)
                            {
                                foreach
(FlowData fi in flows)
                                {
                                    if (fd !=
fi)
                                    {

fd.Par.Add(fi);
                                    }
                                }
                            }
                        }
                    }
                });
        }

        /// <summary>
        /// Фильтрация дней и временных
интевалов для потока
        /// </summary>
        /// <param name="day">День</param>
        /// <param name="lt">Временной
интервал</param>
        /// <param name="flow">Поток</param>
        /// <returns>Приоритет
выбора</returns>
        private int
FilterDayTime(WeekDay2Item day, LectureTime
lt, FlowData flow)
        {
            // TODO: Реализуйте логику
фильтрации дней и временных интервалов для
потока
            return 1;
        }

        /// <summary>
        /// Определение возможных комбинаций
дней и временных интевалов для потока
        /// </summary>
        /// <param name="flow">Поток</param>
        private void SetDayTimes(FlowData
flow)
        {
            foreach (DayItem wd in WeekDays)
            {
                var t = Times;
                foreach (KeyValuePair<int,
LectureTime> lt in t.Take(t.Count() -
flow.CountHours + 1))
                {
```

```
                    int priority =
FilterDayTime((WeekDay2Item)wd, lt.Value,
flow);
                    if (priority != -1)
                    {

switch(flow.CountHours)
                        {
                            // TODO:
Реализуйте логику определения приоритета для
комбинации дня и временного интервала
                            default:

flow.DayTimes.Add(new SelectDataWeekTime(wd,
lt.Value, priority));
                                break;
                        }
                    }
                }
            }
            DayTimeSetting.Filter(flow);
            if (flow.DayTimes.Count == 0)
            {
                throw new Exception("Нет
времени для занятий ");
            }
        }

        /// <summary>
        /// Определение аудиторий для потока
        /// </summary>
        /// <param name="flow">Поток</param>
        private void SetRooms(FlowData flow)
        {
            // TODO: Реализуйте назначение
аудиторий для потока
        }

        /// <summary>
        /// Конструктор
        /// </summary>
        /// <param name="ManagerID">ИД
менеджера расписания из БД</param>
        public GeneratorImpl()
        {
            int ManagerID = 2017;
            mFlows = new List<FlowData>();
            mHard = new List<IRestrict>();
            mSoft = new List<IRestrict>();

            string conStr =
ConfigurationManager.ConnectionStrings["Timet
ableConStr"].ConnectionString;
            using (SqlConnection connection =
new SqlConnection(conStr))
            {
                connection.Open();

                Console.WriteLine("Загрузка
дней");
                InitDays(connection,
ManagerID);

                Console.WriteLine("Загрузка
типов занятий");
                InitTypeFlow(connection,
ManagerID);

                Console.WriteLine("Загрузка
временных интервалов");
                InitTimes(connection,
ManagerID);

                Console.WriteLine("Загрузка
групп");
                InitGroups(connection,
ManagerID);

                Console.WriteLine("Загрузка
аудиторий");
                InitRooms(connection,
ManagerID);

                Console.WriteLine("Загрузка
ППС");
                InitTeachers(connection,
ManagerID);

                Console.WriteLine("Загрузка
дисциплин");
                InitDisciplines(connection,
ManagerID);

                DayTimeSetting.Set(this);

                Console.WriteLine("Загрузка
аудиторий для дисциплин");
                ReadDiscRooms(connection,
ManagerID);

                Console.WriteLine("Загрузка
академических потоков");
                ReadFlows(connection,
ManagerID);

                Console.WriteLine("Установка
времени и аудиторий для академических
потоков");
                mFlows.ForEach(
                    f =>
                    {
                        SetDayTimes(f);
                        SetRooms(f);
                    });

                Console.WriteLine("Сортировка
потоков");
                SortFlows();

            }

            // TODO: Добавьте жесткие и
мягкие ограничения для расписания

            // TODO: Добавьте дополнительные
настройки для расписания, если требуется
        }

        public override
IReadOnlyDictionary<int, LectureTime> Times
=> mTimes;
        public override
IReadOnlyDictionary<int, Group> Groups =>
mGroups;
        public override
IReadOnlyDictionary<int, Room> Rooms =>
mRooms;
        public override
IReadOnlyDictionary<int, Teacher> Teachers =>
mTeachers;
        public override
IReadOnlyDictionary<int, Discipline>
Disciplines => mDisciplines;
        public override List<FlowData> Flows
=> mFlows;
```

```
        public override
IReadOnlyList<IRestrict> Hard => mHard;
        public override
IReadOnlyList<IRestrict> Soft => mSoft;
    }
}
```

Файл DayTimeSetting.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Dict;
using ektu.Timetable.Flow;
using ektu.Timetable;
using System.Xml;
using System.Data;

namespace TimetableTemplate
{
    /// <summary>
    /// Установки для дней недели и времени
занятий
    /// </summary>
    public sealed class DayTimeSetting
    {
        /// <summary>
        /// Преподаватель
        /// </summary>
        private Teacher Teacher { get; set; }
        /// <summary>
        /// Группа
        /// </summary>
        private Group Group { get; set; }
        /// <summary>
        /// День недели
        /// </summary>
        private DayItem WeekDays { get; set;
}
        private TimeSpan? StartTime { get;
set; }
        private TimeSpan? FinishTime { get;
set; }

        /// <summary>
        /// Создание установки для
преподавателя
        /// </summary>
        /// <param
name="teacher">Преподаватель</param>
        /// <param name="type">Вид
установки</param>
        private DayTimeSetting(Teacher
teacher)
        {
            Teacher = teacher;
        }

        /// <summary>
        /// Создание установки для группы
        /// </summary>
        /// <param name="group"></param>
        /// <param name="type"></param>
        private DayTimeSetting(Group group)
        {
            Group = group;
        }

        private void
Excule(List<SelectDataWeekTime> times, int
hours)
        {
            if ((StartTime == null) &&
(FinishTime == null))
            {
                // Удалить весь день
                times.RemoveAll(f => f.Day ==
WeekDays);
            }
            else
            {
                // Удалить занятия в заданный
промежуток
                times.RemoveAll(f => (f.Day
== WeekDays) &&

(((f.Time.StartTime >= StartTime) &&
(f.Time.NextHours(hours).FinishTime <=
StartTime)) ||

((f.Time.StartTime >= FinishTime) &&
(f.Time.NextHours(hours).FinishTime <=
FinishTime))));
            }
        }

        private void Exec(FlowData flow)
        {
            if ((Teacher != null) &&
(flow.Teachers.ContainsValue(Teacher) ||
flow.Par.Exists(f =>
f.Teachers.ContainsValue(Teacher))))
            {
                Excule(flow.DayTimes,
flow.CountHours);
            }
            if ((Group != null) &&
(flow.Items.Exists(f => f.Group == Group)))
            {
                Excule(flow.DayTimes,
flow.CountHours);
            }
        }

        private static List<DayTimeSetting>
data;
        private static Generator gen;
        private static Tuple<int, TimeSpan?,
TimeSpan?> ReadItem(string data)
        {
            string[] d = data.Split(' ');
            int de = 0;
            switch (d[0])
            {
                case "Понедельник":
                    de = 1;
                    break;
                case "Вторник":
                    de = 2;
                    break;
                case "Среда":
                    de = 3;
                    break;
                case "Четверг":
                    de = 4;
                    break;
                case "Пятница":
                    de = 5;
                    break;
                case "Суббота":
                    de = 6;
                    break;
                case "Воскресение":
                    de = 7;
                    break;
```

```
            }
            if (d.Length == 2)
            {
                string[] ts = d[1].Split('-
');
                TimeSpan start =
TimeSpan.Parse(ts[0]);
                TimeSpan finish =
TimeSpan.Parse(ts[1]);
                return new Tuple<int,
TimeSpan?, TimeSpan?>(de, start, finish);
            }
            return new Tuple<int, TimeSpan?,
TimeSpan?>(de, null, null);
        }
        private static List<Tuple<int,
TimeSpan?, TimeSpan?>> ReadExcluding(object
data)
        {
            List<Tuple<int, TimeSpan?,
TimeSpan?>> rs = new List<Tuple<int,
TimeSpan?, TimeSpan?>>();
            if (data is List<string>)
            {
                foreach(string e in
(List<string>)data)
                {
                    rs.Add(ReadItem(e));
                }
            }
            else
            {

rs.Add(ReadItem((string)data));
            }
            return rs;
        }
        public static void Set(Generator
generator)
        {
            data = new
List<DayTimeSetting>();
            gen = generator;

            foreach (Group gr in
generator.Groups.Values)
            {
                if
(gr.HasAttribute("Исключение"))
                {
                    foreach(Tuple<int,
TimeSpan?, TimeSpan?> eg in
ReadExcluding(gr["Исключение"]))
                    {
                        foreach (var r in
generator.WeekDays.Where(f =>
((WeekDay2Item)f).DayID == eg.Item1))
                        {
                            DayTimeSetting
day = new DayTimeSetting(gr);
                            day.WeekDays = r;
                            day.StartTime =
eg.Item2;
                            day.FinishTime =
eg.Item3;
                            data.Add(day);
                        }
                    }
                }
            }

            foreach (Teacher th in
generator.Teachers.Values)
            {
                if
(th.HasAttribute("Исключение"))
                {
                    foreach (Tuple<int,
TimeSpan?, TimeSpan?> eg in
ReadExcluding(th["Исключение"]))
                    {
                        foreach (var r in
generator.WeekDays.Where(f =>
((WeekDay2Item)f).DayID == eg.Item1))
                        {
                            DayTimeSetting
day = new DayTimeSetting(th);
                            day.WeekDays = r;
                            day.StartTime =
eg.Item2;
                            day.FinishTime =
eg.Item3;
                            data.Add(day);
                        }
                    }
                }
            }
        }

        public static void Filter(FlowData
flow)
        {
            foreach(DayTimeSetting d in data)
            {
                d.Exec(flow);
            }
        }
    }
}
```

Файл MyExcluder.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml;
using ektu.Timetable.Dict;
using ektu.Timetable.Flow;
using ektu.Timetable.Timetable;

namespace TimetableTemplate
{
    public class MyExcluder: Excluder
    {
        public override bool
Exclude(TimetableItem item, Room room,
DayItem day, LectureTime lt)
        {
            // TODO: Реализация исключений
            return false;
        }
        public MyExcluder()
        {

        }
    }
}
```

Файл App.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <connectionStrings>
    <add name="TimetableConStr"
connectionString="data
source=(local);Integrated
```

```
Security=True;Database=Timetable;MultipleActi
veResultSets=True"/>
  </connectionStrings>
  <startup>
    <supportedRuntime version="v4.0"
sku=".NETFramework,Version=v4.6" />
  </startup>
</configuration>
```

Файл Properties\AssemblyInfo.cs

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// Управление общими сведениями о сборке
осуществляется с помощью
// набора атрибутов. Измените значения этих
атрибутов, чтобы изменить сведения,
// связанные со сборкой.
[assembly:
AssemblyTitle("TimetableTemplate")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly:
AssemblyProduct("TimetableTemplate")]
[assembly: AssemblyCopyright("Copyright ©
2023")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Параметр ComVisible со значением FALSE
делает типы в сборке невидимыми
// для COM-компонентов.  Если требуется
обратиться к типу в этой сборке через
// COM, задайте атрибуту ComVisible значение
TRUE для этого типа.
[assembly: ComVisible(false)]

// Следующий GUID служит для идентификации
библиотеки типов, если этот проект будет
видимым для COM
[assembly: Guid("972ab680-e5e5-44aa-b9b4-
fb9b99266e9e")]

// Сведения о версии сборки состоят из
следующих четырех значений:
//
//      Основной номер версии
//      Дополнительный номер версии
//   Номер сборки
//      Редакция
//
// Можно задать все значения или принять
номера сборки и редакции по умолчанию
// используя "*", как показано ниже:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

Приложение Д – Свидетельство на программный продукт

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ

РЕСПУБЛИКА КАЗАХСТАН

СВИДЕТЕЛЬСТВО

О ВНЕСЕНИИ СВЕДЕНИЙ В ГОСУДАРСТВЕННЫЙ РЕЕСТР ПРАВ НА ОБЪЕКТЫ, ОХРАНЯЕМЫЕ АВТОРСКИМ ПРАВОМ

№ 20431 от «23» сентября 2021 года

Фамилия, имя, отчество, (если оно указано в документе, удостоверяющем личность) автора (ов):
КУМАРГАЖАНОВА САУЛЕ КУМАРГАЖАНОВНА, ФЕДЬКИН ЕВГЕНИЙ МИХАЙЛОВИЧ, ДЕНИСОВА НАТАЛЬЯ ФЕДОРОВНА

Вид объекта авторского права: **программа для ЭВМ**

Название объекта: **Программа для составления расписания семестровых занятий для ВУЗов**

Дата создания объекта: **11.06.2019**

Құжат түпнұсқалығын http://www.kazpatent.kz/ru сайтының "Авторлық құқық" бөлімінде тексеруге болады https://copyright.kazpatent.kz

Подлинность документа возможно проверить на сайте kazpatent.kz в разделе «Авторское право» https://copyright.kazpatent.kz

Подписано ЭЦП Оспанов Е.К.

Приложение Е – SQL-скрипты для импорта данных из базы данных образовательного портала в базу данных составления расписания

Е.1 Скрипт представлений базы данных образовательного портал для извлечения данных

```
USE [SPortal]
GO

CREATE VIEW [TimetableGenerator].[Employee]
AS
SELECT E.EmployeeID, E.Name
  FROM dbo.Employee E
GO

CREATE VIEW [TimetableGenerator].[LectureTime]
AS
SELECT LT.LectureTimeID,
       CAST(LEFT(LT.Name, 5) AS time) AS StartTime,
       CAST(RIGHT(LT.Name, 5) AS time) AS FinishTime,
       LT.NumberLecture
  FROM dbo.LectureTime LT
  WHERE LT.StudySystemID = 2 AND LT.LectureTimeSchemeID = 4
GO

CREATE VIEW [TimetableGenerator].[Rooms]
AS
SELECT LR.LectureRoomID AS RoomID,
       ISNULL(LR.PlaceCount, 30) AS Places,
       LR.RoomTypeID,
       B.Alias + '-' + LR.Number AS Name,
       D.ShortName AS DepartmentName
  FROM dbo.LectureRoom LR INNER JOIN dbo.Building B
    ON LR.BuildingID = B.BuildingID LEFT OUTER JOIN dbo.Department D
    ON LR.DepartmentID = D.DepartmentID
  WHERE LR.IsStudyRoom = 'Д' --AND B.Alias + '-' + LR.Number NOT IN ('Г-1-301', 'Г-1-416')

GO

CREATE VIEW [TimetableGenerator].[vwFlowData]
AS

SELECT I.ShortName AS InstName,
       Dp.ShortName,
       D.DisciplineID,
       D.Name,
       CDTD.HoursLecture AS HoursLecture,
       CDTD.HoursLaboratory AS HoursLaboratory,
       CASE WHEN Sp.EducationProgramID IN (2,3) AND CD.CycleDisciplineID = 15 THEN CDTD.HoursSRSP
            ELSE CDTD.HoursPractice END AS HoursPractice,

dbo.CountStudyDiscipline(CDTD.TermDistributionsID, G.GroupID) -
         (SELECT COUNT(*)
            FROM dbo.StudyNotDiscipline N INNER JOIN dbo.Study S
              ON N.StudyID = S.StudyID
            WHERE S.GroupID = G.GroupID AND N.TermDistributionsID = CDTD.TermDistributionsID)
       AS CountStudent,
       G.GroupID,
       CASE WHEN CDTD.HoursLecture > 0 THEN ISNULL(CAST(WSG.LectureFlowID AS nvarchar(50)), CAST(CDTD.TermDistributionsID AS nvarchar(20)) + '/' + CAST(G.GroupID AS nvarchar(20))) END AS FlowLecture,
       CASE WHEN Sp.EducationProgramID IN (2,3) AND CD.CycleDisciplineID = 15 THEN ISNULL(CAST(WSG.SRSPFlowID AS nvarchar(50)), CAST(CDTD.TermDistributionsID AS nvarchar(20)) + '/' + CAST(G.GroupID AS nvarchar(20)))
            WHEN CDTD.HoursPractice > 0 THEN ISNULL(CAST(WSG.PracticFlowID AS nvarchar(50)), CAST(CDTD.TermDistributionsID AS nvarchar(20)) + '/' + CAST(G.GroupID AS nvarchar(20))) END AS FlowPracticе,
       CASE WHEN CDTD.HoursLaboratory > 0 THEN ISNULL(CAST(WSG.LaboratoryFlowID AS nvarchar(50)), CAST(CDTD.TermDistributionsID AS nvarchar(20)) + '/' + CAST(G.GroupID AS nvarchar(20))) END AS FlowLab,
       CASE WHEN Sp.EducationProgramID IN (2,3) AND CD.CycleDisciplineID = 15 AND T.W = 3 THEN 1 ELSE T.W END AS TypeWork,
       WSE.LectureRoomID,
       CASE WHEN Dp.ShortName = 'ИЯ' AND WSE.TypeWork = 1 THEN DENSE_RANK() OVER (PARTITION BY CASE WHEN CDTD.HoursPractice > 0 THEN ISNULL(CAST(CASE WHEN D.DisciplineID = 30 THEN NULL ELSE WSG.PracticFlowID END AS nvarchar(50)), CAST(CDTD.TermDistributionsID AS nvarchar(20)) + '/' + CAST(G.GroupID AS nvarchar(20))) END ORDER BY WSE.EmployeeID) - 1 ELSE ISNULL(WSE.SubGroup, 1) END AS SubGroup,
       WSE.EmployeeID,
       CDTD.TermNumber,
       CASE WHEN WSG.CourseTypeID = 2 THEN 1
            ELSE 0 END AS IsOnline,

TimetableGenerator.GetSubGroupMask(G.GroupID, CDTD.TermDistributionsID) AS SubGroupMask
  FROM dbo.Institutes I INNER JOIN dbo.DictDepartmentType DTT
    ON I.DepartmentTypeID = DTT.DepartmentTypeID INNER JOIN dbo.Department Dp
    ON I.InstituteID = Dp.InstituteID INNER JOIN dbo.DictDepartmentType DTTD
    ON Dp.DepartmentTypeID = DTTD.DepartmentTypeID INNER JOIN dbo.Discipline D
    ON Dp.DepartmentID = D.DepartmentID INNER JOIN dbo.CurriculumDisciplines CD
    ON D.DisciplineID = CD.DisciplineID INNER JOIN
```

```
dbo.CurriculumDisciplinesTermsDistributions
CDTD
              ON  CD.CurriculumDisciplinesID
=  CDTD.CurriculumDisciplinesID  INNER  JOIN
dbo.Curriculum C
              ON      CD.CurriculumID      =
C.CurriculumID          INNER          JOIN
dbo.DictBaseEducation DBE
              ON     C.BaseEducationID     =
DBE.BaseEducationID INNER JOIN dbo.Groups G
              ON      C.CurriculumID      =
G.CurriculumID          INNER          JOIN
dbo.StateOrdersSpecialty SOS
              ON      C.StateOrderID      =
SOS.StateOrderID INNER JOIN dbo.Specialty Sp
              ON      SOS.SpecialtyID      =
Sp.SpecialtyID  INNER  JOIN  (VALUES(0),  (1),
(2)) AS T(W)
              ON  1  =  1  LEFT  OUTER  JOIN
dbo.WorkScheduleGroup WSG
              ON  WSG.TermDistributionsID  =
CDTD.TermDistributionsID AND
                  WSG.GroupID   =   G.GroupID
LEFT OUTER JOIN dbo.WorkScheduleEmployee WSE
              ON  WSG.WorkScheduleGroupID  =
WSE.WorkScheduleGroupID AND
                  WSE.TypeWork = T.W AND
                        WSE.EmployeeID  NOT  IN
(4877, 4879, 4876, 4878)
       WHERE   CDTD.TermNumber   =   (2023   -
SOS.OrderYear) * 2 + 1 AND
             CDTD.TypeCurriculumTermItemID =
1 AND
             DTT.IsFaculty = 1 AND
             DTTD.IsChair = 1 AND
             G.GroupStatusID IN (1,3) AND
             G.StudyTechnologyID  IN  (1,  2)
AND
             C.KindFormStudyID = 1 AND
              ISNULL(WSG.CourseTypeID, 1) <>
3 AND
             Sp.EducationProgramID        IN
(1,2,3,7) AND

dbo.CountStudyDiscipline(CDTD.TermDistributio
nsID, G.GroupID) -
             (SELECT COUNT(*)
                FROM  dbo.StudyNotDiscipline
N INNER JOIN dbo.Study S
                         ON     N.StudyID     =
S.StudyID
                WHERE  S.GroupID  =  G.GroupID
AND          N.TermDistributionsID          =
CDTD.TermDistributionsID)  > 0
      GO
```

E.2 – Скрипты импорта данных в базу данных составления расписания

а) Скрипт импорта данных по дисциплинам

```
DECLARE @ManagerID int = 2017

BEGIN TRY

  BEGIN TRAN

  DELETE FROM Timetable.DisciplineRooms
    WHERE ManagerID = @ManagerID

  DELETE FROM Timetable.Discipline
    WHERE ManagerID = @ManagerID

  INSERT                              INTO
Timetable.Discipline(Name,      DisciplineID,
ManagerID)
       SELECT DISTINCT Ds.Name,
            D.DisciplineID,
              @ManagerID
         FROM  SPortal.vwFlowData  D  INNER
JOIN SPortal.Discipline Ds
                ON    Ds.DisciplineID    =
D.DisciplineID

  INSERT                              INTO
Timetable.DisciplineRooms(TypeLecture,
RoomTypeID,      RoomID,      PrioritySelect,
TeacherID, ManagerID, DisciplineID)
       SELECT LR.TypeLecture,
            LR.RoomTypeID,
              LR.LectureRoomID,
              LR.PrioritySelect,
              LR.EmployeeID,
              @ManagerID,
              D.DisciplineID
         FROM
SPortal.LectureRoomDiscipline LR  INNER JOIN
Timetable.Discipline D
                ON     D.DisciplineID     =
LR.DisciplineID      LEFT      OUTER      JOIN
SPortal.dbo.LectureRoom R
                  ON  R.LectureRoomID  =
LR.LectureRoomID
         WHERE  D.ManagerID  =  @ManagerID
AND
           (R.IsStudyRoom  = 'Д' OR
                  LR.RoomTypeID  IS  NOT
NULL OR
                  LR.EmployeeID  IS  NOT
NULL)

  COMMIT

END TRY
BEGIN CATCH

  IF @@TRANCOUNT > 0
    ROLLBACK TRAN

  SELECT ERROR_MESSAGE()

END CATCH
```

б) Скрипт импорта данных по учебным группам

```
DECLARE @ManagerID int = 2017

BEGIN TRY

  BEGIN TRAN

  DELETE FROM Timetable.Groups
    WHERE ManagerID = @ManagerID

  DECLARE GC CURSOR LOCAL FORWARD_ONLY
FOR
    SELECT DISTINCT G.GroupID,
           G.Code,
           DBE.BaseEducationKindID   AS
kind,
           Sp.EducationProgramID     AS
prog,
           CEILING(D.TermNumber  /  3.0)
AS course,
           Sp.Code AS SpecCode,
           Sp.InstituteID
      FROM  SPortal.vwFlowData  D  INNER
JOIN SPortal.Groups G
             ON  D.GroupID  =  G.GroupID
INNER JOIN SPortal.Curriculum C
              ON     G.CurriculumID     =
C.CurriculumID          INNER          JOIN
SPortal.StateOrdersSpecialty SOS
```

```
                  ON     C.StateOrderID     =
SOS.StateOrderID INNER JOIN SPortal.Specialty
Sp
                  ON     SOS.SpecialtyID    =
Sp.SpecialtyID           INNER           JOIN
SPortal.DictBaseEducation DBE
                  ON   C.BaseEducationID   =
DBE.BaseEducationID
            ORDER BY Sp.InstituteID, Sp.Code,
course

        OPEN GC

        DECLARE @GroupID int
        DECLARE @Code nvarchar(100)
        DECLARE @kind int
        DECLARE @prog int
        DECLARE @Course int
        DECLARE @SpecCode nvarchar(30)
        DECLARE @InstituteID int

        FETCH  NEXT  FROM  GC  INTO  @GroupID,
@Code,  @kind,  @prog,  @Course,  @SpecCode,
@InstituteID

        WHILE @@FETCH_STATUS = 0
        BEGIN

       INSERT                              INTO
Timetable.Groups(ManagerID, GroupID, Code)
         VALUES(@ManagerID, @GroupID, @Code)

       DECLARE @TGroupID int
       SET @TGroupID = SCOPE_IDENTITY()

       INSERT                              INTO
Timetable.GroupAttributes(TimetableGroupID,
Name, Value)
         VALUES(@TGroupID, 'Вид', @kind)

       INSERT                              INTO
Timetable.GroupAttributes(TimetableGroupID,
Name, Value)
         VALUES(@TGroupID,       'Программа',
@prog)

       INSERT                              INTO
Timetable.GroupAttributes(TimetableGroupID,
Name, Value)
         VALUES(@TGroupID, 'Курс', @Course)

       INSERT                              INTO
Timetable.GroupAttributes(TimetableGroupID,
Name, Value)
         VALUES(@TGroupID, 'ОП', @SpecCode)

       INSERT                              INTO
Timetable.GroupAttributes(TimetableGroupID,
Name, Value)
         VALUES(@TGroupID,    'ИД    школы',
@InstituteID)

          FETCH  NEXT  FROM  GC  INTO  @GroupID,
@Code,  @kind,  @prog,  @Course,  @SpecCode,
@InstituteID

        END

        CLOSE GC
        DEALLOCATE GC

        COMMIT

      END TRY
      BEGIN CATCH

        IF @@TRANCOUNT > 0
          ROLLBACK TRAN

        SELECT ERROR_MESSAGE()

      END CATCH
```

в) Скрипт импорт данных по временным интервалам

```
      DECLARE @ManagerID int = 2017

      BEGIN TRY

        BEGIN TRAN

        DELETE FROM Timetable.LectureTime
          WHERE ManagerID = @ManagerID

        INSERT                              INTO
Timetable.LectureTime(LectureTimeID,
StartTime,     FinishTime,     NumberLecture,
ManagerID)
          SELECT LectureTimeID,
              StartTime,
                 FinishTime,
                 NumberLecture,
                 @ManagerID
         FROM SPortal.LectureTime

        COMMIT

      END TRY
      BEGIN CATCH

        IF @@TRANCOUNT > 0
          ROLLBACK TRAN

        SELECT ERROR_MESSAGE()

      END CATCH
```

г) Скрипт импорт данных по учебным аудиториям

```
      DECLARE @ManagerID int = 2017

      BEGIN TRY

        BEGIN TRAN

        DELETE FROM Timetable.Rooms
          WHERE ManagerID = @ManagerID

        DECLARE RC CURSOR LOCAL FORWARD_ONLY
FOR
          SELECT RoomID,
                 Places,
                 RoomTypeID,
                 Name,
                 DepartmentName
            FROM SPortal.Rooms
          UNION
          SELECT R.LectureRoomID,
                 ISNULL(R.PlaceCount, 30),
                 -1,
                 B.Alias + '-'+ R.Number,
                 NULL
            FROM
SPortal.LectureRoomDiscipline  LR  INNER  JOIN
SPortal.LectureRoom R
                   ON    LR.LectureRoomID    =
R.LectureRoomID INNER JOIN SPortal.Building B
                   ON      R.BuildingID      =
B.BuildingID
             WHERE  LR.LectureRoomID  NOT  IN
(SELECT R.RoomID

FROM SPortal.Rooms R) AND
                      (R.IsStudyRoom   =   'Д'
OR
                       LR.RoomTypeID  IS  NOT
NULL OR
                       LR.EmployeeID  IS  NOT
NULL)

        OPEN RC
```

```
        DECLARE @RoomID int
        DECLARE @Places int
        DECLARE @RoomTypeID int
        DECLARE @Name nvarchar(1000)
        DECLARE               @DepartmentName
nvarchar(1000)

        FETCH  NEXT  FROM  RC  INTO  @RoomID,
@Places, @RoomTypeID, @Name, @DepartmentName

        WHILE @@FETCH_STATUS = 0
        BEGIN

       INSERT  INTO  Timetable.Rooms(RoomID,
Places, RoomTypeID, Name, ManagerID)
         VALUES(@RoomID,             @Places,
@RoomTypeID, @Name, @ManagerID)

       DECLARE @TRoomID int
       SET @TRoomID = SCOPE_IDENTITY()

       IF @DepartmentName IS NOT NULL
         INSERT                          INTO
Timetable.RoomAttributes(TimetableRoomID,
Name, Value)
           VALUES(@TRoomID,       'Кафедра',
@DepartmentName)

       FETCH  NEXT  FROM  RC  INTO  @RoomID,
@Places, @RoomTypeID, @Name, @DepartmentName

        END

        CLOSE RC
        DEALLOCATE RC

        COMMIT

      END TRY
      BEGIN CATCH

        IF @@TRANCOUNT > 0
          ROLLBACK TRAN

        SELECT ERROR_MESSAGE()

      END CATCH
```

д) Скрипт импорта данных по ППС

```
      DECLARE @ManagerID int = 2017

      BEGIN TRY

        BEGIN TRAN

        DELETE FROM Timetable.Teachers
          WHERE ManagerID = @ManagerID

        INSERT                          INTO
Timetable.Teachers(TeacherID,          Name,
ManagerID)
         SELECT  DISTINCT  E.EmployeeID  AS
TeacherID,
                E.Name,
                @ManagerID
           FROM  SPortal.vwFlowData  D  INNER
JOIN SPortal.Employee E
                 ON     D.EmployeeID     =
E.EmployeeID

        COMMIT

      END TRY
      BEGIN CATCH

        IF @@TRANCOUNT > 0
          ROLLBACK TRAN

        SELECT ERROR_MESSAGE()

      END CATCH
```

е) Скрипт импорта данных по академическим потокам

```
      DECLARE @ManagerID int = 2017

      BEGIN TRY

        BEGIN TRAN

        DECLARE @FID nvarchar(100)
        DECLARE @MS int
        DECLARE @InstName nvarchar(100)
        DECLARE @DepName nvarchar(100)
        DECLARE @Hours int
        DECLARE @IsOnline int
        DECLARE @FlowID1 int
        DECLARE @FlowID2 int

        DECLARE       @T       TABLE(InstName
nvarchar(500),
                      ShortName
nvarchar(500),
                      DisciplineID int,
                      Name nvarchar(1500),
                      HoursLecture int,
                      HoursLaboratory int,
                      HoursPractice int,
                      CountStudent int,
                      GroupID int,
                      FlowLecture
nvarchar(50),
                      FlowPractice
nvarchar(50),
                      FlowLab
nvarchar(50),
                      TypeWork int,
                      LectureRoomID int,
                      SubGroup int,
                      EmployeeID int,
                      TermNumber int,
                             IsOnline
int,
                             SubGroupMask
nvarchar(4000))

        INSERT INTO @T
          SELECT *
          FROM SPortal.vwFlowData

        DELETE FROM Timetable.Flows
          WHERE ManagerID = @ManagerID

        -- лекции
        DECLARE  LC  CURSOR  LOCAL  FORWARD_ONLY
FOR
          SELECT DISTINCT FlowLecture
            FROM @T
            WHERE FlowLecture IS NOT NULL

        OPEN LC

        FETCH NEXT FROM LC INTO @FID

        WHILE @@FETCH_STATUS = 0
        BEGIN

       SELECT           @InstName           =
dbo.StringConcatAgg(distinct InstName, ', '),
                @DepName                =
dbo.StringConcatAgg(distinct   ShortName,   ',
'),
                @Hours = MAX(HoursLecture),
                @IsOnline = MAX(IsOnline),
                @MS = MAX(SubGroup)
          FROM @T
          WHERE   FlowLecture   =   @FID   AND
TypeWork = 0
```

```
       INSERT                              INTO
Timetable.Flows(ManagerID,        HoursCount,
CountElements, TypeLecture)
         VALUES(@ManagerID, @Hours, 1, 0)
       SET @FlowID1 = SCOPE_IDENTITY()

       INSERT                              INTO
Timetable.Flows(ManagerID,        HoursCount,
CountElements, TypeLecture)
         VALUES(@ManagerID, @Hours, 1, 0)
       SET @FlowID2 = SCOPE_IDENTITY()

       INSERT                              INTO
Timetable.FlowAttributes(FlowID, Name, Value)
         VALUES(@FlowID1,           'Школа',
@InstName),
               (@FlowID2,           'Школа',
@InstName)

       INSERT                              INTO
Timetable.FlowAttributes(FlowID, Name, Value)
         VALUES(@FlowID1,          'Кафедра',
@DepName),
               (@FlowID2,          'Кафедра',
@DepName)

       INSERT                              INTO
Timetable.FlowAttributes(FlowID, Name, Value)
         VALUES(@FlowID1,   'Недели',   'Все
недели'),
                     (@FlowID2,    'Недели',
'1 раз в 2 недели')

       IF @IsOnline = 1
         INSERT                            INTO
Timetable.FlowAttributes(FlowID, Name, Value)
           VALUES(@FlowID1, 'Онлайн', 1),
                 (@FlowID2, 'Онлайн', 1)

       INSERT                              INTO
Timetable.FlowTeachers(FlowID,      SubGroup,
TeacherID)
         SELECT DISTINCT F.ID,
                       CASE  WHEN  @MS  =  1
THEN 0 ELSE SubGroup END,
                       EmployeeID
              FROM   @T,   (VALUES(@FlowID1),
(@FlowID2)) AS F(ID)
              WHERE FlowLecture = @FID AND
                    EmployeeID  IS  NOT  NULL
AND
                       TypeWork = 0

       INSERT                              INTO
Timetable.FlowGroups(FlowID,        SubGroup,
GroupID,     CountStudent,     DisciplineID,
TermNumber, SubGroupMask)
         SELECT DISTINCT F.ID,
                       CASE  WHEN  @MS  =  1
THEN 0 ELSE SubGroup END,
                       GroupID,
                       CountStudent,
                       DisciplineID,
                       TermNumber,
                       SubGroupMask
              FROM   @T,   (VALUES(@FlowID1),
(@FlowID2)) AS F(ID)
              WHERE FlowLecture = @FID AND
                    TypeWork = 0

          FETCH NEXT FROM LC INTO @FID
       END

       CLOSE LC
       DEALLOCATE LC

       -- практика
       DECLARE  PC  CURSOR  LOCAL  FORWARD_ONLY
FOR
          SELECT DISTINCT FlowPractice
            FROM @T
            WHERE FlowPractice IS NOT NULL

       OPEN PC

       FETCH NEXT FROM PC INTO @FID

       WHILE @@FETCH_STATUS = 0
       BEGIN
       SELECT           @InstName           =
dbo.StringConcatAgg(distinct InstName, ', '),
                  @DepName                 =
dbo.StringConcatAgg(distinct   ShortName,   ',
'),
                  @Hours                   =
MAX(HoursPractice),
                  @IsOnline = MAX(IsOnline),
                  @MS = MAX(SubGroup)
         FROM @T
         WHERE   FlowPractice   =   @FID   AND
TypeWork = 1

       INSERT                              INTO
Timetable.Flows(ManagerID,        HoursCount,
CountElements, TypeLecture)
         VALUES(@ManagerID, @Hours, 1, 1)
       SET @FlowID1 = SCOPE_IDENTITY()

       INSERT                              INTO
Timetable.Flows(ManagerID,        HoursCount,
CountElements, TypeLecture)
         VALUES(@ManagerID, @Hours, 1, 1)
       SET @FlowID2 = SCOPE_IDENTITY()

       INSERT                              INTO
Timetable.FlowAttributes(FlowID, Name, Value)
         VALUES(@FlowID1,           'Школа',
@InstName),
               (@FlowID2,           'Школа',
@InstName)

       INSERT                              INTO
Timetable.FlowAttributes(FlowID, Name, Value)
         VALUES(@FlowID1,          'Кафедра',
@DepName),
               (@FlowID2,          'Кафедра',
@DepName)

       INSERT                              INTO
Timetable.FlowAttributes(FlowID, Name, Value)
         VALUES(@FlowID1,   'Недели',   'Все
недели'),
                     (@FlowID2,    'Недели',
'1 раз в 2 недели')

       IF @IsOnline = 1
         INSERT                            INTO
Timetable.FlowAttributes(FlowID, Name, Value)
           VALUES(@FlowID1, 'Онлайн', 1),
                 (@FlowID2, 'Онлайн', 1)

       INSERT                              INTO
Timetable.FlowTeachers(FlowID,      SubGroup,
TeacherID)
         SELECT DISTINCT F.ID,
                       CASE  WHEN  @MS  =  1
THEN 0 ELSE SubGroup END,
                       EmployeeID
              FROM   @T,   (VALUES(@FlowID1),
(@FlowID2)) AS F(ID)
              WHERE FlowPractice = @FID AND
                    EmployeeID  IS  NOT  NULL
AND
                       TypeWork = 1

       INSERT                              INTO
Timetable.FlowGroups(FlowID,        SubGroup,
```

```
GroupID,      CountStudent,      DisciplineID,
TermNumber, SubGroupMask)
         SELECT DISTINCT F.ID,
                        CASE  WHEN  @MS  =  1
THEN 0 ELSE SubGroup END,
                        GroupID,
                        CountStudent,
                        DisciplineID,
                        TermNumber,
                        SubGroupMask
                FROM   @T,   (VALUES(@FlowID1),
(@FlowID2)) AS F(ID)
                WHERE FlowPractice = @FID AND
                      TypeWork = 1

          FETCH NEXT FROM PC INTO @FID
        END

        CLOSE PC
        DEALLOCATE PC

        -- лабораторные
        DECLARE     LabC     CURSOR     LOCAL
FORWARD_ONLY FOR
          SELECT DISTINCT FlowLab
            FROM @T
            WHERE FlowLab IS NOT NULL

        OPEN LabC

        FETCH NEXT FROM LabC INTO @FID

        WHILE @@FETCH_STATUS = 0
        BEGIN

       SELECT           @InstName           =
dbo.StringConcatAgg(distinct InstName, ', '),
                  @DepName                  =
dbo.StringConcatAgg(distinct   ShortName,   ',
'),
                  @Hours                    =
MAX(HoursLaboratory),
                  @IsOnline = MAX(IsOnline),
                  @MS = MAX(SubGroup)
          FROM @T
          WHERE FlowLab = @FID AND TypeWork =
2

       INSERT                               INTO
Timetable.Flows(ManagerID,        HoursCount,
CountElements, TypeLecture)
          VALUES(@ManagerID, @Hours, 1, 2)
        SET @FlowID1 = SCOPE_IDENTITY()

       INSERT                               INTO
Timetable.Flows(ManagerID,        HoursCount,
CountElements, TypeLecture)
          VALUES(@ManagerID, @Hours, 1, 2)
        SET @FlowID2 = SCOPE_IDENTITY()

       INSERT                               INTO
Timetable.FlowAttributes(FlowID, Name, Value)
          VALUES(@FlowID1,           'Школа',
@InstName),
                (@FlowID2,           'Школа',
@InstName)

       INSERT                               INTO
Timetable.FlowAttributes(FlowID, Name, Value)
          VALUES(@FlowID1,          'Кафедра',
@DepName),
                (@FlowID2,          'Кафедра',
@DepName)

       INSERT                               INTO
Timetable.FlowAttributes(FlowID, Name, Value)
          VALUES(@FlowID1,    'Недели',    'Все
недели'),
                        (@FlowID2,    'Недели',
'1 раз в 2 недели')

       IF @IsOnline = 1
         INSERT                             INTO
Timetable.FlowAttributes(FlowID, Name, Value)
           VALUES(@FlowID1, 'Онлайн', 1),
                    (@FlowID2, 'Онлайн', 1)

       INSERT                               INTO
Timetable.FlowTeachers(FlowID,      SubGroup,
TeacherID)
         SELECT DISTINCT F.ID,
                        CASE  WHEN  @MS  =  1
THEN 0 ELSE SubGroup END,
                        EmployeeID
               FROM   @T,   (VALUES(@FlowID1),
(@FlowID2)) AS F(ID)
               WHERE FlowLab = @FID AND
                     EmployeeID  IS  NOT  NULL
AND
                        TypeWork = 2

       INSERT                               INTO
Timetable.FlowGroups(FlowID,        SubGroup,
GroupID,      CountStudent,      DisciplineID,
TermNumber, SubGroupMask)
         SELECT DISTINCT F.ID,
                        CASE  WHEN  @MS  =  1
THEN 0 ELSE SubGroup END,
                        GroupID,
                        CountStudent,
                        DisciplineID,
                        TermNumber,
                        SubGroupMask
               FROM   @T,   (VALUES(@FlowID1),
(@FlowID2)) AS F(ID)
               WHERE FlowLab = @FID AND
                     TypeWork = 2

          FETCH NEXT FROM LabC INTO @FID
        END

        CLOSE LabC
        DEALLOCATE LabC

        -- удаление потоков без ППС
        DELETE FROM Timetable.Flows
          WHERE FlowID IN (SELECT F.FlowID
                                FROM
Timetable.Flows       F       INNER       JOIN
Timetable.FlowGroups FG
                                     ON
F.FlowID     =     FG.FlowID     INNER     JOIN
Timetable.Groups G
                                     ON
FG.GroupID = G.GroupID AND

            G.ManagerID = F.ManagerID
                                   WHERE
F.ManagerID = @ManagerID AND
                                NOT
EXISTS(SELECT *

                      FROM
Timetable.FlowTeachers T

                                   WHERE
T.FlowID = F.FlowID))

        COMMIT

      END TRY
      BEGIN CATCH

        IF @@TRANCOUNT > 0
          ROLLBACK TRAN

        SELECT ERROR_MESSAGE(), ERROR_LINE()
```

```
END CATCH
```

Приложение Ж – Исходный код консольного приложения для составления расписания на 3-й триместр 2022-2023 учебного года

Файл TesingApp.csproj

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="14.0" DefaultTargets="Build" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <Import Project="$(MSBuildExtensionsPath)\$(MSBuildToolsVersion)\Microsoft.Common.props" Condition="Exists('$(MSBuildExtensionsPath)\$(MSBuildToolsVersion)\Microsoft.Common.props')" />
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
    <ProjectGuid>{972AB680-E5E5-44AA-B9B4-FB9B99266E9E}</ProjectGuid>
    <OutputType>Exe</OutputType>

<AppDesignerFolder>Properties</AppDesignerFolder>

<RootNamespace>TestingApp</RootNamespace>

<AssemblyName>TestingApp</AssemblyName>

<TargetFrameworkVersion>v4.6</TargetFrameworkVersion>
    <FileAlignment>512</FileAlignment>

<AutoGenerateBindingRedirects>true</AutoGenerateBindingRedirects>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">

<PlatformTarget>x64</PlatformTarget>
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <Optimize>false</Optimize>
    <OutputPath>bin\Debug\</OutputPath>

<DefineConstants>DEBUG;TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
    <Prefer32Bit>false</Prefer32Bit>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">

<PlatformTarget>x64</PlatformTarget>
    <DebugType>pdbonly</DebugType>
    <Optimize>true</Optimize>

<OutputPath>bin\Release\</OutputPath>

<DefineConstants>TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)' == 'Test1|AnyCPU'">
    <DebugSymbols>true</DebugSymbols>
    <OutputPath>bin\Test1\</OutputPath>

<DefineConstants>DEBUG;TRACE</DefineConstants>
    <DebugType>full</DebugType>

<PlatformTarget>AnyCPU</PlatformTarget>
    <ErrorReport>prompt</ErrorReport>

<CodeAnalysisRuleSet>MinimumRecommendedRules.ruleset</CodeAnalysisRuleSet>
    <Prefer32Bit>true</Prefer32Bit>
  </PropertyGroup>
  <ItemGroup>
    <Reference Include="System" />
    <Reference Include="System.Configuration" />
    <Reference Include="System.Core" />
    <Reference Include="System.Web.Extensions" />
    <Reference Include="System.Xml.Linq" />
    <Reference Include="System.Data.DataSetExtensions" />
    <Reference Include="Microsoft.CSharp" />
    <Reference Include="System.Data" />
    <Reference Include="System.Net.Http" />
    <Reference Include="System.Xml" />
  </ItemGroup>
  <ItemGroup>
    <Compile Include="DayTimeSetting.cs" />
    <Compile Include="GeneratorImpl.cs" />
    <Compile Include="MyExcluder.cs" />
    <Compile Include="Program.cs" />
    <Compile Include="Properties\AssemblyInfo.cs" />
  </ItemGroup>
  <ItemGroup>
    <None Include="App.config" />
  </ItemGroup>
  <ItemGroup>
    <ProjectReference Include="..\ektu.Timetable\ektu.Timetable.csproj">
      <Project>{924363d5-6bc2-45c7-8ae3-f472842b36ee}</Project>
      <Name>ektu.Timetable</Name>
    </ProjectReference>
  </ItemGroup>
  <ItemGroup />
  <ItemGroup>
    <Content Include="College\Disciplines.sql" />
    <Content Include="College\Flows.sql" />
    <Content Include="College\Groups.sql" />
    <Content Include="College\LectureTime.sql" />
```

```
          <Content
Include="College\Teachers.sql" />
          <Content Include="SQL\Flows.sql" />
          <Content   Include="SQL\Groups.sql"
/>
          <Content
Include="College\Rooms.sql" />
          <Content Include="SQL\Rooms.sql" />
          <Content Include="SQL\Teachers.sql"
/>
          <Content
Include="SQL\LectureTime.sql" />
          <Content
Include="SQL\Disciplines.sql" />
          <Content Include="Viewer.html" />
        </ItemGroup>
        <Import
Project="$(MSBuildToolsPath)\Microsoft.CSharp
.targets" />
       <!-- To modify your build process,
add your task inside one of the targets below
and uncomment it.
            Other similar extension points
exist, see Microsoft.Common.targets.
        <Target Name="BeforeBuild">
        </Target>
        <Target Name="AfterBuild">
        </Target>
        -->
      </Project>
```

Файл Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.IO;
using ektu.Timetable;
using ektu.Timetable.Timetable;
using System.Dynamic;
using ektu.Timetable.Restrict;

namespace TestingApp
{
    class Program
    {
        private       static       void
OutTables(List<TimetableData> tables)
        {
            int num = 1;
            const    string    st1    =
"┌──────┬──────────────────┬──────────────┬──────
──────┬──────────┬──────────┬──────────────┬──────
──────┐";
            const    string    st2    =
"├──────┼──────────────────┼──────────────┼──────
──────┼──────────┼──────────┼──────────────┼──────
──────┤";
            const    string    st3    =
"└──────┴──────────────────┴──────────────┴──────
──────┴──────────┴──────────┴──────────────┴──────
──────┘";
            Console.WriteLine(st1);

Console.WriteLine(String.Format("│ {0,-3} │
{1,-13} │ {2,-12} │ {3,-12} │ {4,-6} │ {5,-
10} │ {6,-16} │",
                    "num", "value", "hard",
"soft", "defect", "generation", "type"));
            Console.WriteLine(st2);
            foreach (TimetableData t in
tables)
            {

                Console.WriteLine(

String.Format("│{0,4} │ {1,-11:0.0000000} │
{2,-11:0.000000} │ {3,-11:0.0000000} │ {4,6}
│ {5,10} │ {6,-16} │",
                    num++,
t.RestrictValue,         t.HardRestrictValue,
t.SoftRestrictValue,   t.Defected().Sum(f   =>
f.DefectedCount),               t.Generation,
t.Type.ToString()));
            }
            Console.WriteLine(st3);
        }

        public       static       void
Main(string[] args)
        {
            for (int v = 200; v < 2000;
v += 200)
            {
                Generator   gen   =   new
GeneratorImpl(v);

                #region       Настройка
логирования
                int stepNumber = 0;
                string      path      =
DateTime.Now.ToString("yyyyMMdd_HHmm")      +
"\\";

Directory.CreateDirectory(path);
                gen.StartStep +=
                    num =>
                    {

Console.WriteLine("Начало    шага    №"    +
num.ToString());
                        stepNumber    =
num;
                    };
                gen.InitialInfo +=
                    t =>
                    {
                        int id = 1;
                        foreach
(TimetableData item in t)
                        {

item.GetXml().Save(path    +    "Start_"    +
(id++).ToString() + ".xml");
                        }
                        OutTables(t);
                        return false;
                    };
                gen.FinishExecute +=
                    t =>
                    {

Console.WriteLine("   Заверешние   мутаций   и
скрещивания");

Console.WriteLine("     Всего   особей:   "   +
t.Count.ToString());
                        OutTables(t);
                        return false;
                    };
                gen.FinishAlive +=
                    t =>
                    {

Console.WriteLine("     Выжило   особей:   "   +
t.Count.ToString());
                        OutTables(t);
                        string   np   =
path + stepNumber.ToString() + "\\";

Directory.CreateDirectory(np);
                        for (int i = 0;
i < t.Count; i++)
                        {

t[i].GetXml().Save(np + (i + 1).ToString() +
".xml");
                        }
```

```
                                return
stepNumber == 100;
                            };
                        gen.InfoAction += msg
=> Console.WriteLine(msg);
                        #endregion

                        gen.CountTimetable    =
10;
                        gen.CountTimetableCalc
= 20;
                        gen.FitnessValue      =
gen.Hard.Sum(f => f.Factor) + gen.Soft.Sum(f
=> f.Factor) * 2;
                        gen.Excluder    =    new
MyExcluder();

                        TimetableData    d    =
gen.GenerateTimetable(null);
                    }

                }
            }
        }
```

Файл MyExcluder.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml;
using ektu.Timetable.Dict;
using ektu.Timetable.Flow;
using ektu.Timetable.Timetable;

namespace TestingApp
{
    public class MyExcluder: Excluder
    {
        private   List<Tuple<int,   int,
int?, int?>> data = new List<Tuple<int, int,
int?, int?>>();
        private   List<Tuple<int,   int,
int>>  rooms  =  new  List<Tuple<int,  int,
int>>()
        {
            new     Tuple<int,     int,
int>(202, 6, 4),
            new     Tuple<int,     int,
int>(205, 5, 7),
            new     Tuple<int,     int,
int>(205, 5, 8),
            new     Tuple<int,     int,
int>(212, 4, 9),
            new     Tuple<int,     int,
int>(212, 4, 10),
            new     Tuple<int,     int,
int>(232, 4, 7),
            new     Tuple<int,     int,
int>(232, 4, 8),
            new     Tuple<int,     int,
int>(232, 4, 10),
            new     Tuple<int,     int,
int>(347, 5, 9),
            new     Tuple<int,     int,
int>(347, 5, 10),
            new     Tuple<int,     int,
int>(347, 5, 11),
            new     Tuple<int,     int,
int>(355, 1, 8),
            new     Tuple<int,     int,
int>(355, 1, 9),
            new     Tuple<int,     int,
int>(426, 4, 7),
            new     Tuple<int,     int,
int>(518, 5, 7),
            new     Tuple<int,     int,
int>(518, 5, 8),
            new     Tuple<int,     int,
int>(524, 6, 2),
            new     Tuple<int,     int,
int>(524, 6, 3),
            new     Tuple<int,     int,
int>(781, 1, 7),
            new     Tuple<int,     int,
int>(781, 1, 8),
            new     Tuple<int,     int,
int>(781, 5, 7),
            new     Tuple<int,     int,
int>(781, 5, 8),
            new     Tuple<int,     int,
int>(838, 3, 5),
            new     Tuple<int,     int,
int>(838, 3, 6),
            new     Tuple<int,     int,
int>(838, 3, 7),
            new     Tuple<int,     int,
int>(838, 5, 7),
            new     Tuple<int,     int,
int>(838, 5, 8)
        };
        public       override       bool
Exclude(TimetableItem   item,   Room   room,
DayItem day, LectureTime lt)
        {
            if    (rooms.Exists(f    =>
(f.Item1  ==  room.RoomID)  &&  (f.Item2  ==
day.ID    %    100)    &&    (f.Item3    >=
lt.NumberLecture)     &&     (f.Item3     <=
lt.NumberLecture  +  item.Flow.CountHours  -
1)))
            {
                return true;
            }

            if  (item.Flow.Linked.Count
== 1)
            {
                TimetableItem   ni    =
item.Owner[item.Flow.Linked[0]];
                if ((ni.Day != null) &&
(ni.Time != null))
                {
                    return   (day.ID   /
100    !=    ni.Day.ID    /    100)    ||
(Math.Abs(ni.Time.NumberLecture             -
lt.NumberLecture) != 1) || (room != ni.Room);
                }
            }

            return   data.Exists(f   =>
(f.Item1 == day.ID) &&

(f.Item2 >= lt.NumberLecture) && (f.Item2 <=
(lt.NumberLecture  +  item.Flow.CountHours  -
1)) &&

(item.Flow.Teachers.ContainsKey(f.Item4 ?? 0)
||        item.Flow.Items.Exists(ff        =>
ff.Group.GroupID == (f.Item3 ?? 0))));

        }
        public MyExcluder()
        {

        }
    }
}
```

Файл DayTimeSetting.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable.Dict;
using ektu.Timetable.Flow;
```

```
using ektu.Timetable;
using System.Xml;
using System.Data;

namespace TestingApp
{
    /// <summary>
    /// Установки для дней недели и времени занятий
    /// </summary>
    public sealed class DayTimeSetting
    {
        /// <summary>
        /// Преподаватель
        /// </summary>
        private Teacher Teacher { get; set; }
        /// <summary>
        /// Группа
        /// </summary>
        private Group Group { get; set; }
        /// <summary>
        /// День недели
        /// </summary>
        private DayItem WeekDays { get; set; }
        private TimeSpan? StartTime { get; set; }
        private TimeSpan? FinishTime { get; set; }

        /// <summary>
        /// Создание установки для преподавателя
        /// </summary>
        /// <param name="teacher">Преподаватель</param>
        /// <param name="type">Вид установки</param>
        private DayTimeSetting(Teacher teacher)
        {
            Teacher = teacher;
        }

        /// <summary>
        /// Создание установки для группы
        /// </summary>
        /// <param name="group"></param>
        /// <param name="type"></param>
        private DayTimeSetting(Group group)
        {
            Group = group;
        }

        private void Excule(List<SelectDataWeekTime> times, int hours)
        {
            if ((StartTime == null) && (FinishTime == null))
            {
                // Удалить весь день
                times.RemoveAll(f => f.Day == WeekDays);
            }
            else
            {
                // Удалить занятия в заданный промежуток
                times.RemoveAll(f => (f.Day == WeekDays) &&

(((f.Time.StartTime >= StartTime) && (f.Time.NextHours(hours).FinishTime <= FinishTime)) ||

((f.Time.FinishTime >= StartTime) && (f.Time.NextHours(hours).FinishTime <= FinishTime))));
            }
        }

        private void Exec(FlowData flow)
        {
            if ((Teacher != null) && (flow.Teachers.ContainsValue(Teacher) || flow.Par.Exists(f => f.Teachers.ContainsValue(Teacher))))
            {
                Excule(flow.DayTimes, flow.CountHours);
            }
            if ((Group != null) && (flow.Items.Exists(f => f.Group == Group)))
            {
                Excule(flow.DayTimes, flow.CountHours);
            }
        }

        private static List<DayTimeSetting> data;
        private static Generator gen;
        private static Tuple<int, TimeSpan?, TimeSpan?> ReadItem(string data)
        {
            string[] d = data.Split(' ');
            int de = 0;
            switch (d[0])
            {
                case "Понедельник":
                    de = 1;
                    break;
                case "Вторник":
                    de = 2;
                    break;
                case "Среда":
                    de = 3;
                    break;
                case "Четверг":
                    de = 4;
                    break;
                case "Пятница":
                    de = 5;
                    break;
                case "Суббота":
                    de = 6;
                    break;
                case "Воскресение":
                    de = 7;
                    break;
            }
            if (d.Length == 2)
            {
                string[] ts = d[1].Split('-');
                TimeSpan start = TimeSpan.Parse(ts[0]);
                TimeSpan finish = TimeSpan.Parse(ts[1]);
                return new Tuple<int, TimeSpan?, TimeSpan?>(de, start, finish);
            }
            return new Tuple<int, TimeSpan?, TimeSpan?>(de, null, null);
        }
        private static List<Tuple<int, TimeSpan?, TimeSpan?>> ReadExcluding(object data)
        {
```

```
                List<Tuple<int, TimeSpan?, TimeSpan?>> rs = new List<Tuple<int, TimeSpan?, TimeSpan?>>();
                if (data is List<string>)
                {
                    foreach(string e in (List<string>)data)
                    {
                        rs.Add(ReadItem(e));
                    }
                }
                else
                {
                    rs.Add(ReadItem((string)data));
                }
                return rs;
            }
            public static void Set(Generator generator)
            {
                data = new List<DayTimeSetting>();
                gen = generator;

                foreach (Group gr in generator.Groups.Values)
                {
                    if (gr.HasAttribute("Исключение"))
                    {
                        foreach(Tuple<int, TimeSpan?, TimeSpan?> eg in ReadExcluding(gr["Исключение"]))
                        {
                            foreach (var r in generator.WeekDays.Where(f => ((WeekDay2Item)f).DayID == eg.Item1))
                            {
                                DayTimeSetting day = new DayTimeSetting(gr);
                                day.WeekDays = r;
                                day.StartTime = eg.Item2;
                                day.FinishTime = eg.Item3;
                                data.Add(day);
                            }
                        }
                    }
                }

                foreach (Teacher th in generator.Teachers.Values)
                {
                    if (th.HasAttribute("Исключение"))
                    {
                        foreach (Tuple<int, TimeSpan?, TimeSpan?> eg in ReadExcluding(th["Исключение"]))
                        {
                            foreach (var r in generator.WeekDays.Where(f => ((WeekDay2Item)f).DayID == eg.Item1))
                            {
                                DayTimeSetting day = new DayTimeSetting(th);
                                day.WeekDays = r;
                                day.StartTime = eg.Item2;
                                day.FinishTime = eg.Item3;
                                data.Add(day);
                            }
                        }
                    }
                }
            }

            public static void Filter(FlowData flow)
            {
                foreach(DayTimeSetting d in data)
                {
                    d.Exec(flow);
                }
            }
        }
    }
```

Файл GeneratorImpl.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ektu.Timetable;
using ektu.Timetable.Dict;
using ektu.Timetable.Restrict;
using ektu.Timetable.Flow;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
using System.Xml;

namespace TestingApp
{
    /// <summary>
    /// Реализация класса генератора расписания
    /// </summary>
    internal class GeneratorImpl : Generator
    {
        private Dictionary<int, TypeFlowBase> mFlowTypes;
        /// <summary>
        /// Временные интервалы
        /// </summary>
        private Dictionary<int, LectureTime> mTimes;
        /// <summary>
        /// Группы
        /// </summary>
        private Dictionary<int, Group> mGroups;
        /// <summary>
        /// Аудитории
        /// </summary>
        private Dictionary<int, Room> mRooms;
        /// <summary>
        /// ППС
        /// </summary>
        private Dictionary<int, Teacher> mTeachers;
        /// <summary>
        /// Дисциплины
        /// </summary>
        private Dictionary<int, Discipline> mDisciplines;
        /// <summary>
        /// Учебные потоки
        /// </summary>
        private List<FlowData> mFlows;
        /// <summary>
        /// Жесткие ограничения
        /// </summary>
        private List<IRestrict> mHard;
        /// <summary>
```

```
            /// Мягкие ограничения
            /// </summary>
            private List<IRestrict> mSoft;
            /// <summary>
            /// Дни
            /// </summary>
            private           List<DayItem>
mWeekDays;
            public  override  List<DayItem>
WeekDays
            {
                get { return mWeekDays; }
            }
            public   RestrictFollowNextPred
resNext { get; private set; }

            /// <summary>
            /// Выполнение чтения данных из
БД
            /// </summary>
            ///                         <param
name="con">Соединение с БД</param>
            ///   <param   name="cmd">Строка
запроса</param>
            ///                         <param
name="action">Процедура чтения</param>
            private                       void
ReadDB(SqlConnection    con,    string    cmd,
Action<IDataRecord> action)
            {
                SqlCommand      cmds      =
con.CreateCommand();
                cmds.CommandText = cmd;
                cmds.CommandTimeout = 3600;
                using (SqlDataReader reader
= cmds.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        action(reader);
                    }
                }
            }

            /// <summary>
            /// Чтение временных интервалов
            /// </summary>
            ///                         <param
name="connection">Подключение к БД</param>
            ///  <param  name="ManagerID">ИД
расписания</param>
            private                      void
InitTimes(SqlConnection    connection,    int
ManagerID)
            {
                mTimes        =        new
Dictionary<int, LectureTime>();
                const   string   ltSql   =
"SELECT * FROM Timetable.LectureTime LT WHERE
LT.ManagerID     =     {0}     ORDER     BY
LT.NumberLecture";
                ReadDB(connection,
String.Format(ltSql, ManagerID),
                    r =>
                    {
                        LectureTime   lt   =
new LectureTime();
                        lt["LectureTimeID"]
= r["LectureTimeID"];
                        lt["StartTime"]   =
r["StartTime"];
                        lt["FinishTime"]  =
r["FinishTime"];
                        lt["NumberLecture"]
= r["NumberLecture"];

mTimes.Add(lt.LectureTimeID, lt);
                    });
                LectureTime next = null;
                foreach  (LectureTime  lt  in
Times.Values.Where(f   =>   f.NumberLecture   <
20).OrderByDescending(f => f.NumberLecture))
                {
                    lt.Next = next;
                    next = lt;
                }
                foreach  (LectureTime  lt  in
Times.Values.Where(f   =>   f.NumberLecture   >
20).OrderByDescending(f => f.NumberLecture))
                {
                    lt.Next = next;
                    next = lt;
                }

                foreach  (LectureTime  lt  in
Times.Values)
                {

                    lt.Cross = Times.Where(
                        f                 =>
(((f.Value.StartTime >= lt.StartTime) &&

(f.Value.StartTime <= lt.FinishTime)) ||

((f.Value.FinishTime >= lt.StartTime) &&

(f.Value.FinishTime   <=   lt.FinishTime)))   &&
(f.Value != lt)
                    ).Select(f            =>
f.Value).ToList();
                }
            }

            /// <summary>
            /// Чтение данные по группам
            /// </summary>
            ///                         <param
name="connection">Подключение к БД</param>
            ///  <param  name="ManagerID">ИД
расписания</param>
            private                      void
InitGroups(SqlConnection    connection,    int
ManagerID)
            {
                mGroups        =        new
Dictionary<int, Group>();
                const   string   grSql   =
"SELECT  *  FROM  Timetable.Groups  G  WHERE
G.ManagerID = {0}";
                const  string  grSqlAttr  =
"SELECT        Name,        Value        FROM
Timetable.GroupAttributes               WHERE
TimetableGroupID = {0}";
                ReadDB(connection,
String.Format(grSql, ManagerID),
                    r =>
                    {
                        Group   gr   =   new
Group();
                        gr["GroupID"]     =
r["GroupID"];
                        gr["Code"]        =
r["Code"];
                        ReadDB(connection,
String.Format(grSqlAttr,
r["TimetableGroupID"]),
                            rs =>
                            {
                                string name
= (string)rs[0];
                                if
(gr.HasAttribute(name))
                                {
                                    if
(!(gr[name] is IList<string>))
                                    {
```

```
List<string> lst = new List<string>();

lst.Add((string)gr[name]);

gr[name] = lst;
                                    }

(gr[name]                                   as
List<string>).Add((string)rs[1]);
                                }
                                else
                                {

gr[name] = rs[1];
                                }
                            });

mGroups.Add(gr.GroupID, gr);
                    });
            }

            /// <summary>
            /// Чтение данных по аудиториям
            /// </summary>
            ///                         <param
name="connection">Поключение к БД</param>
            ///  <param  name="ManagerID">ИД
расписания</param>
            private                     void
InitRooms(SqlConnection    connection,    int
ManagerID)
            {
                mRooms        =        new
Dictionary<int, Room>();
                const   string   sqlRoom   =
"SELECT  *  FROM  Timetable.Rooms  R  WHERE
R.ManagerID = {0}";
                const string sqlRoomAttr =
"SELECT       Name,       Value       FROM
Timetable.RoomAttributes               WHERE
TimetableRoomID = {0}";
                ReadDB(connection,
String.Format(sqlRoom, ManagerID),
                    r =>
                    {
                        Room   rm   =   new
Room();
                        rm["RoomID"]      =
r["RoomID"];
                        rm["Name"]        =
r["Name"];
                        rm["Places"]      =
r["Places"];
                        rm["RoomTypeID"]  =
r["RoomTypeID"];
                        ReadDB(connection,
String.Format(sqlRoomAttr,
r["TimetableRoomID"]),
                            rs =>
                            {
                                string name
= (string)rs[0];
                                if
(rm.HasAttribute(name))
                                {
                                    if
(!(rm[name] is IList<string>))
                                    {

List<string> lst = new List<string>();

lst.Add((string)rm[name]);

rm[name] = lst;
                                    }

(rm[name]                                   as
List<string>).Add((string)rs[1]);
                                }
                                else
                                {

rm[name] = rs[1];
                                }
                            });

mRooms.Add(rm.RoomID, rm);
                        });
            }

            /// <summary>
            /// Чтение данных по ППС
            /// </summary>
            ///                         <param
name="connection">Подключение к БД</param>
            ///  <param  name="ManagerID">ИД
расписания</param>
            private                     void
InitTeachers(SqlConnection   connection,   int
ManagerID)
            {
                mTeachers       =       new
Dictionary<int, Teacher>();
                const  string  sqlTeacher  =
"SELECT  *  FROM  Timetable.Teachers  WHERE
ManagerID = {0}";
                const string sqlTeacherAttr
=     "SELECT     Name,     Value     FROM
Timetable.TeacherAttributes            WHERE
TimetableTeacherID = {0}";
                ReadDB(connection,
String.Format(sqlTeacher, ManagerID),
                    r =>
                    {
                        Teacher  th  =  new
Teacher();
                        th["TeacherID"]   =
r["TeacherID"];
                        th["Name"]        =
r["Name"];
                        ReadDB(connection,
String.Format(sqlTeacherAttr,
r["TimetableTeacherID"]),
                            rs =>
                            {
                                string name
= (string)rs[0];
                                if
(th.HasAttribute(name))
                                {
                                    if
(!(th[name] is IList<string>))
                                    {

List<string> lst = new List<string>();

lst.Add((string)th[name]);

th[name] = lst;
                                    }

(th[name]                                   as
List<string>).Add((string)rs[1]);
                                }
                                else
                                {

th[name] = rs[1];
                                }
                            });

mTeachers.Add(th.TeacherID, th);
                        });
            }

            /// <summary>
            /// Чтение данных по дням
```

```
            /// </summary>
            ///                         <param
name="connection">Подключение к БД</param>
            /// <param name="ManagerID">ИД
расписания</param>
            private                     void
InitDays(SqlConnection    connection,    int
ManagerID)
            {
                mWeekDays       =       new
List<DayItem>();

mWeekDays.AddRange(WeekDay2Item.Monday);

mWeekDays.AddRange(WeekDay2Item.Tuesday);

mWeekDays.AddRange(WeekDay2Item.Wednesday);

mWeekDays.AddRange(WeekDay2Item.Thursday);

mWeekDays.AddRange(WeekDay2Item.Friday);

mWeekDays.AddRange(WeekDay2Item.Saturday);
            }

            private                     void
InitTypeFlow(SqlConnection  connection,  int
ManagerID)
            {
                mFlowTypes      =       new
Dictionary<int, TypeFlowBase>();

mFlowTypes.Add(TypeFlowLecture.Item.ID,
TypeFlowLecture.Item);

mFlowTypes.Add(TypeFlowPractice.Item.ID,
TypeFlowPractice.Item);

mFlowTypes.Add(TypeFlowLaboratory.Item.ID,
TypeFlowLaboratory.Item);
            }

            /// <summary>
            ///     Чтение     данных     по
дисциплинам
            /// </summary>
            ///                         <param
name="connection">Подключение к БД</param>
            /// <param name="ManagerID">ИД
расписания</param>
            private                     void
InitDisciplines(SqlConnection connection, int
ManagerID)
            {
                mDisciplines     =     new
Dictionary<int, Discipline>();
                const  string  sqlDisc  =
"SELECT  *  FROM  Timetable.Discipline  WHERE
ManagerID = {0}";
                ReadDB(connection,
String.Format(sqlDisc, ManagerID),
                    r =>
                    {
                        Discipline ds = new
Discipline();
                        ds["DisciplineID"]
= r["DisciplineID"];
                        ds["Name"]        =
r["Name"];

mDisciplines.Add(ds.DisciplineID, ds);
                    });
            }


            /// <summary>
            /// Сортировка потоков
            /// </summary>
            private void SortFlows()
            {
                Dictionary<int, int> disc =
new Dictionary<int, int>();
                mFlows.ForEach(
                    f =>
                    {
                        int      id      =
f.Items.Min(fx                          =>
fx.Discipline.DisciplineID);
                        if
(!disc.ContainsKey(id))
                        {
                            disc[id] = 0;
                        }
                        disc[id]         +=
f.CountStudent;
                    });

                ForelClassificator
classificator             =             new
ForelClassificator(mFlows, 0.8, 0.1, 0.1);
                List<List<FlowData>>
clusters = classificator.GetClusters(0.5);
                mFlows.Clear();
                clusters.Sort((f1,  f2)  =>
f2.Sum(f  =>  f.CountStudent)  -  f1.Sum(f  =>
f.CountStudent));
                while (clusters.Count > 0)
                {

mFlows.AddRange(clusters[0].OrderByDescending
(f => f.Items.Sum(fx => fx.CountStudent)));
                    clusters.RemoveAt(0);
                }
            }

            /// <summary>
            ///   Чтение   аудиторий   для
дисциплин
            /// </summary>
            ///                         <param
name="connection">Подключение к БД</param>
            /// <param name="ManagerID">ИД
расписания</param>
            private                     void
ReadDiscRooms(SqlConnection  connection,  int
ManagerID)
            {
                ReadDB(connection,  "SELECT
*  FROM  Timetable.DisciplineRooms  D  WHERE
D.ManagerID = " + ManagerID.ToString(),
                    r =>
                    {
                        Discipline   ds   =
mDisciplines[(int)r["DisciplineID"]];
                        TypeFlowBase  t  =
mFlowTypes[(int)r["TypeLecture"]];
                        string val = null;
                        if (r["RoomTypeID"]
!= DBNull.Value)
                        {
                            val          =
"RoomTypeID=" + r["RoomTypeID"].ToString() +
"," + r["PrioritySelect"].ToString();
                        }
                        if  (r["RoomID"]  !=
DBNull.Value)
                        {
                            val = "RoomID="
+    r["RoomID"].ToString()    +    ","    +
r["PrioritySelect"].ToString();
                            if
(r["TeacherID"] != DBNull.Value)
                            {
                                val      +=
",TeacherID=" + r["TeacherID"].ToString();
                            }
                        }
```

```
                        if
(!ds.Rooms.ContainsKey(t))
                        {
                            ds.Rooms.Add(t,
new List<string>());
                        }

ds.Rooms[t].Add(val);
                    });
            }

            /// <summary>
            /// Чтение потоков из БД
            /// </summary>
            ///                         <param
name="connection">Подключение к БД</param>
            ///  <param  name="ManagerID">ИД
расписания</param>
            private                     void
ReadFlows(SqlConnection    connection,    int
ManagerID, int countFlow)
            {
                const   string   sqlFlow   =
"SELECT TOP {1} * FROM Timetable.Flows WHERE
ManagerID = {0}";
                const string sqlFlowAttr =
"SELECT        Name,        Value        FROM
Timetable.FlowAttributes WHERE FlowID = {0}";
                const string sqlFlowTeacher
= "SELECT * FROM Timetable.FlowTeachers WHERE
FlowID = {0}";
                int fid = 0;
                ReadDB(connection,
String.Format(sqlFlow, ManagerID, countFlow),
                    r =>
                    {
                        int     FlowID     =
(int)r["FlowID"];
                        int   HoursCount   =
(int)r["HoursCount"];
                        int CountElements =
(int)r["CountElements"];
                        TypeFlowBase type =
mFlowTypes[(int)r["TypeLecture"]];

                        // атрибуты потока
                        Dictionary<string,
object>   attr   =   new   Dictionary<string,
object>();
                        ReadDB(connection,
String.Format(sqlFlowAttr, FlowID),
                            rs =>
                            {
                                string name
= (string)rs[0];
                                if
(attr.ContainsKey(name))
                                {
                                    if
(!(attr[name] is IList<string>))
                                    {

List<string> lst = new List<string>();

lst.Add((string)attr["name"]);
                                    }

(attr[name]                                 as
List<string>).Add((string)rs[1]);
                                }
                                else
                                {

attr[name] = rs[1];
                                }
                            });


                        // ППС
                        Dictionary<int,
List<int>>  teachers  =  new  Dictionary<int,
List<int>>();
                        ReadDB(connection,
String.Format(sqlFlowTeacher, FlowID),
                            rt =>
                            {
                                int   s   =
(int)rt["SubGroup"];
                                if
(!teachers.ContainsKey(s))
                                {

teachers.Add(s, new List<int>());
                                }

teachers[s].Add((int)rt["TeacherID"]);
                            });

                        // Группы
                        Dictionary<int,
List<FlowItem>> groups = new Dictionary<int,
List<FlowItem>>();
                        ReadDB(connection,
"SELECT  *  FROM  Timetable.FlowGroups  WHERE
FlowID = " + FlowID.ToString(),
                            rt =>
                            {
                                int   s   =
(int)rt["SubGroup"];
                                if
(!groups.ContainsKey(s))
                                {

groups.Add(s, new List<FlowItem>());
                                }
                                if
(Groups.ContainsKey((int)rt["GroupID"]))
                                {

FlowItem item = new FlowItem()
                                    {

SubGroup = s,

CountStudent = (int)rt["CountStudent"],

Group = Groups[(int)rt["GroupID"]],

Discipline                                  =
Disciplines[(int)rt["DisciplineID"]]
                                    };
                                    string
sm = (string)rt["SubGroupMask"];

item.SubGroupMask = new BitArray(sm.Length);
                                    for(int
i = 0; i < sm.Length; i++)
                                    {

item.SubGroupMask.Set(i, sm[i] == '1');
                                    }

groups[s].Add(item);
                                }
                            });


                        for (int i = 0; i <
CountElements; i++)
                        {
                            Dictionary<int,
int>  teachersCount  =  new  Dictionary<int,
int>();
                            List<FlowData>
flows = new List<FlowData>();
                            foreach
(KeyValuePair<int,  List<FlowItem>>  gr  in
groups)
```

```
                                {
                                    FlowData
flowData = new FlowData()
                                    {
                                        FlowId
= (++fid).ToString(),

TypeFlow = type,

CountHours = HoursCount
                                    };

foreach(KeyValuePair<string,  object>  v  in
attr)
                                    {

flowData.FlowParam.Add(v.Key, v.Value);
                                    }

flowData.Items.AddRange(gr.Value);
                                    if
(teachers.ContainsKey(gr.Key))
                                    {
                                        foreach
(int tid in teachers[gr.Key])
                                        {

flowData.Teachers.Add(tid, Teachers[tid]);
                                            if
(!teachersCount.ContainsKey(tid))
                                            {

teachersCount.Add(tid, 0);
                                            }

teachersCount[tid]++;
                                        }
                                    }

mFlows.Add(flowData);

flows.Add(flowData);
                                }
                                if
((flows.Count > 1) && (teachersCount.All(f =>
f.Value == 1)))
                                {
                                    foreach
(FlowData fd in flows)
                                    {
                                        foreach
(FlowData fi in flows)
                                        {
                                            if
(fd != fi)
                                            {

fd.Par.Add(fi);
                                            }
                                        }
                                    }
                                }
                            }
                        });
                }

                /// <summary>
                /// Фильтрация дней и временных
интевалов для потока
                /// </summary>
                ///                         <param
name="day">День</param>
                ///  <param  name="lt">Временной
интервал</param>
                ///                         <param
name="flow">Поток</param>
                ///          <returns>Приоритет
выбора</returns>
                private                     int
FilterDayTime(WeekDay2Item  day,  LectureTime
lt, FlowData flow)
                {

                    if
(((string)flow.FlowParam["Недели"]   ==   "Все
недели")      &&      (day.WeekNumber      !=
WeekDay2Item.WeekType.Both))
                    {
                        return -1;
                    }
                    if
(((string)flow.FlowParam["Недели"] == "1 раз
в   2   недели")   &&   (day.WeekNumber   ==
WeekDay2Item.WeekType.Both))
                    {
                        return -1;
                    }

                    if
(flow.Teachers.ContainsKey(3043))
                    {
                        return 1;
                    }

                    if
(flow.FlowParam.ContainsKey("Онлайн"))
                    {
                        if  (day.DayID  ==  6)
return -1;
                        if  (lt.NumberLecture <
10) return -1;
                        return 1;
                    }

                    //  в  субботу  не  ставить,
кроме онлайн
                    if (day.DayID == 6)
                    {
                        if
(flow.Items[0].SubGroup    >=    30)    return
lt.NumberLecture <= 9 ? 1 : -1;

                        if
(flow.FirstDiscipline.DisciplineID  ==  5381)
return -1;
                        if
(flow.FirstDiscipline.DisciplineID  ==  5377)
return -1;
                        if   (flow.TypeFlow   is
TypeFlowLecture)
                        {
                            if
(flow.CountStudent > 30) return -1;
                        }
                        return lt.NumberLecture
- flow.CountHours - 1 <= 9 ? 1000 : -1;
                    }

                    int prog = flow.Items.Max(f
=>
Convert.ToInt32(f.Group.Value<string>("Програ
мма")));

                    if (prog == 1)
                    {
                        return lt.NumberLecture
<= 10 ? 1 : 1000;
                    }

                    if  (flow.Items.Exists(f  =>
f.Group.GroupID == 12817))
                    {
                        if (lt.NumberLecture <=
4) return -1;
                        if (lt.NumberLecture <=
6) return 100;
                        return 1;
```

```
                }


                if ((prog == 2) || (prog == 3))
                {
                    if (lt.NumberLecture <= 6) return -1;
                    return 1;
                }

                return 100;
            }

            /// <summary>
            ///    Определение    возможных комбинаций дней и временных интевалов для потока
            /// </summary>
            ///                         <param name="flow">Поток</param>
            private                     void SetDayTimes(FlowData flow)
            {
                foreach   (DayItem   wd   in WeekDays)
                {
                    var t = Times;
                    foreach (KeyValuePair<int,    LectureTime>    lt    in t.Take(t.Count() - flow.CountHours + 1))
                    {
                        int   priority   = FilterDayTime((WeekDay2Item)wd,     lt.Value, flow);
                        if (priority != -1)
                        {

switch(flow.CountHours)
                            {
                                case 1:

flow.DayTimes.Add(new  SelectDataWeekTime(wd, lt.Value, priority));
                                    break;
                                case 2:

flow.DayTimes.Add(new  SelectDataWeekTime(wd, lt.Value, lt.Value.NumberLecture % 2 == 1 ? priority : 1000));
                                    break;
                                case 3:

flow.DayTimes.Add(new  SelectDataWeekTime(wd, lt.Value, lt.Value.NumberLecture % 3 == 1 ? priority : 1000));
                                    break;
                                case 4:

flow.DayTimes.Add(new  SelectDataWeekTime(wd, lt.Value, lt.Value.NumberLecture % 2 == 1 ? priority : 1000));
                                    break;
                                case 5:

flow.DayTimes.Add(new  SelectDataWeekTime(wd, lt.Value, lt.Value.NumberLecture % 5 == 1 ? priority : 1000));
                                    break;
                                default:

flow.DayTimes.Add(new  SelectDataWeekTime(wd, lt.Value, priority));
                                    break;
                            }
                        }
                    }
                }

DayTimeSetting.Filter(flow);
                if  (flow.DayTimes.Count  == 0)
                {
                    throw                new Exception("Нет времени для занятий ");
                }
            }

            /// <summary>
            ///  Определение  аудиторий  для потока
            /// </summary>
            ///                         <param name="flow">Поток</param>
            private  void  SetRooms(FlowData flow)
            {
                flow.FlowParam["IsEmpSpec"] = false;
                List<int> mTypeRooms = new List<int>();
                int         count         = flow.CountStudent;

                if (flow.FlowParam.ContainsKey("Онлайн"))
                {
                    flow.Rooms.Add(new RoomSelectInfo(1, Rooms[1057]));
                    return;
                }

                if (flow.Items.Count == 1)
                {
                    string    room_roup    = flow.Items[0].Group.Value<string>("Code");
                    foreach(Room    r    in Rooms.Where(f                                => f.Value.Value<string>("Name")              == room_roup).Select(f => f.Value))
                    {
                        flow.Rooms.Add(new RoomSelectInfo(1, r));
                    }
                    if  (flow.Rooms.Count  > 0)
                    {
                        return;
                    }
                }

                if ((((string)flow.FlowParam["Кафедра"] == "ИЯ") ||

((string)flow.FlowParam["Кафедра"]          == "КиРЯ"))       &&       (flow.TypeFlow      is TypeFlowPractice))
                {
                    count = 5;
                }

                if ((flow.Items.Count == 1) && (flow.Items[0].Group.Value<string>("Программа") == "3"))
                {
                    count = 1;
                }

                bool IsEmpSpec = false;
                if (!flow.Items[0].Discipline.Rooms.ContainsKey(flow.TypeFlow))
                {
```

```
flow.Items[0].Discipline.Rooms.Add(flow.TypeF
low, new List<string>());

flow.Items[0].Discipline.Rooms[flow.TypeFlow]
.Add("RoomTypeID=1,10");
                }
                var           x           =
flow.Items[0].Discipline.Rooms[flow.TypeFlow]
;
                if     (flow.TypeFlow     is
TypeFlowLecture)
                {
                    if
(flow.Items[0].Discipline.Rooms.ContainsKey(T
ypeFlowPractice.Item))
                    {

x.AddRange(flow.Items[0].Discipline.Rooms[Typ
eFlowPractice.Item]);
                    }
                    if
(flow.Items[0].Discipline.Rooms.ContainsKey(T
ypeFlowLaboratory.Item))
                    {

x.AddRange(flow.Items[0].Discipline.Rooms[Typ
eFlowLaboratory.Item]);
                    }
                    x                    =
x.Distinct().ToList();
                }
                foreach (string room in x)
                {
                    string[]      rs      =
room.Split('=');
                    string[]     rs2     =
rs[1].Split(',');
                    int         id         =
Convert.ToInt32(rs2[0]);
                    int        pid        =
Convert.ToInt32(rs2[1]);
                    if  ((count  <=  2)  &&
(flow.Items[0].Discipline.DisciplineID     !=
30))
                    {
                        if     (rs[0]     ==
"RoomID")
                        {
                            rs[0]         =
"RoomTypeID";
                        }
                    }
                    switch (rs[0])
                    {
                        case "RoomTypeID":
                            if (IsEmpSpec)
                            {
                                continue;
                            }
                            foreach   (Room
rm in mRooms.Values)
                            {
                                bool   v   =
false;
                                string
nameRoom = rm.Value<string>("Name");
                                int
roomType = rm.Value<int>("RoomTypeID");
                                string
depName = !rm.HasAttribute("Кафедра") ? null
: rm.Value<string>("Кафедра");
                                if  ((count
<=                  2)                  &&
(flow.Items[0].Discipline.DisciplineID     !=
30))
                                {
                                    v     =
nameRoom.StartsWith("К");
                                }
                                else
                                {
                                    int
placeCount = count;
                                    if
(roomType   ==   14   &&   !(flow.TypeFlow   is
TypeFlowLecture)) placeCount = 1;
                                    if
((flow.TypeFlow   is   TypeFlowPractice)   &&
(count > 30)) count = 30;

                                    v     =
(roomType == id) &&

((rm.Places   >=   placeCount   -   5)   ||
(flow.TypeFlow is TypeFlowLaboratory)) &&

(nameRoom.StartsWith("Г-")) &&

(!nameRoom.StartsWith("Г-М")) &&

(!nameRoom.StartsWith("Г-Л-М")) &&

(!nameRoom.StartsWith("К")) &&

(!nameRoom.StartsWith("Ф"));
                                }
                                if (v)
                                {

flow.Rooms.Add(new RoomSelectInfo((depName ==
null)          ||          (depName          ==
(string)flow.FlowParam["Кафедра"])  ?  pid  :
100000, rm));
                                }
                            }
                            break;
                        case "RoomID":
                            if  ((rs.Length
==                  3)                    &&
flow.Teachers.ContainsKey(Convert.ToInt32(rs[
2])))
                            {

mTypeRooms.Add(mRooms[id].Value<int>("RoomTyp
eID"));
                                if
((flow.TypeFlow   is   TypeFlowLaboratory)   ||
(mRooms[id].Places >= count - 5))
                                {
                                    if
(!IsEmpSpec)
                                    {

flow.Rooms.Clear();

IsEmpSpec = true;

flow.FlowParam["IsEmpSpec"] = true;
                                    }

flow.Rooms.Add(new        RoomSelectInfo(pid,
mRooms[id]));
                                }
                            }
                            else
                            {
                                if
((rs.Length == 2) && (!IsEmpSpec))
                                {

mTypeRooms.Add(mRooms[id].Value<int>("RoomTyp
eID"));
                                    if
((flow.TypeFlow   is   TypeFlowLaboratory)   ||
(mRooms[id].Places >= count - 5))
                                    {
```

```
flow.Rooms.Add(new RoomSelectInfo(pid, mRooms[id]));
                                    }
                                }
                            }

                            break;
                        }
                    }
                    if (flow.Rooms.Count == 0)
                    {
                        foreach (Room rm in mRooms.Values)
                        {
                            bool v = false;
                            string nameRoom = rm.Value<string>("Name");
                            int roomType = rm.Value<int>("RoomTypeID");
                            string depName = !rm.HasAttribute("Кафедра") ? null : rm.Value<string>("Кафедра");
                            if ((flow.CountStudent == 1) && (flow.Items[0].Discipline.DisciplineID != 30))
                            {
                                v = nameRoom.StartsWith("К");
                            }
                            else
                            {
                                v = (mTypeRooms.Contains(roomType)) && ((rm.Places >= (roomType == 14 && !(flow.TypeFlow is TypeFlowLecture) ? 1 : count)) ||
                                (flow.TypeFlow is TypeFlowLaboratory)) &&
                                (nameRoom.StartsWith("Г-")) &&
                                (!nameRoom.StartsWith("Г-М")) &&
                                (!nameRoom.StartsWith("Г-Л-М"));
                            }
                            if (v)
                            {
                                flow.Rooms.Add(new RoomSelectInfo((depName == null) || (depName == (string)flow.FlowParam["Кафедра"]) ? 0 : 100000, rm));
                            }
                        }
                    }

                    var l = flow.Rooms
                        .OrderBy(f => f.Room.HasAttribute("Кафедра") &&
                        f.Room.Value<string>("Кафедра") == (string)flow.FlowParam["Кафедра"] ? 1 : 2)
                        .ThenBy(f => f.Priority)
                        .ThenBy(f => f.Room.Places)
                        .ThenBy(f => f.Room.RoomID)
                        .ToList();
                    if (!(flow.TypeFlow is TypeFlowLecture) || (flow.CountStudent < 30))
                    {
                        l.RemoveAll(f => f.Room.RoomID == 212);
                        l.RemoveAll(f => f.Room.RoomID == 347);
                    }
                    flow.Rooms.Clear();
                    flow.Rooms.AddRange(l);
                }

                /// <summary>
                /// Конструктор
                /// </summary>
                /// <param name="ManagerID">ИД менеджера расписания из БД</param>
                public GeneratorImpl(int countFlow)
                {
                    int ManagerID = 2017;
                    mFlows = new List<FlowData>();
                    mHard = new List<IRestrict>();
                    mSoft = new List<IRestrict>();

                    string conStr = ConfigurationManager.ConnectionStrings["TimetableConStr"].ConnectionString;
                    using (SqlConnection connection = new SqlConnection(conStr))
                    {
                        connection.Open();
                        InitDays(connection, ManagerID);
                        InitTypeFlow(connection, ManagerID);
                        InitTimes(connection, ManagerID);
                        InitGroups(connection, ManagerID);
                        InitRooms(connection, ManagerID);
                        InitTeachers(connection, ManagerID);
                        InitDisciplines(connection, ManagerID);
                        DayTimeSetting.Set(this);
                        ReadDiscRooms(connection, ManagerID);
                        ReadFlows(connection, ManagerID, countFlow);
                        mFlows.ForEach(
                            f =>
                            {
                                SetDayTimes(f);
                                SetRooms(f);
                            });
                        SortFlows();
                    }

                    mHard.Add(new RestrictEmployee(8000));
                    mHard.Add(new RestrictGroup(15000));

                    RestrictRoom resRoom = new RestrictRoom(7000);
                    resRoom.Filter = item =>
                    {
                        // Филиалы сами определяют аудитории
                        if (item.Value<string>("Name").StartsWith("Ф")) { return false; }
                        // Филиалы сами определяют аудитории
                        if (item.Value<string>("Name").StartsWith("К")) { return false; }
                        // онлайн-аудитория
                        if (item.RoomID == 1057) { return false; }
```

```
                    return true;
                };
                mHard.Add(resRoom);

                RestrictOneDayDisciplines
resOneDayDisc               =              new
RestrictOneDayDisciplines(5000);
                resOneDayDisc.FilterFlow  =
item =>
                {
                    if ((item.CountHours ==
1) && (item.DayTimes[0].Day.Cross.Count > 0))
return false;
                    return true;
                };
                mHard.Add(resOneDayDisc);

                RestrictGroupHours
resGroupHours = new RestrictGroupHours(5000,
8, 1);
                resGroupHours.FilterItem =
                    item =>
                    {
                        if
(item.FirstDiscipline.DisciplineID   ==   30)
return false;
                        if
(item.Items[0].SubGroup >= 30) return false;
                        return true;
                    };
                mHard.Add(resGroupHours);

                RestrictGroupHours
resGroupHours4 = new RestrictGroupHours(50,
8, 2, "RestrictGroupHours2");
                resGroupHours4.FilterItem =
                    item =>
                    {
                        if
(item.FirstDiscipline.DisciplineID   ==   30)
return false;
                        if
(item.Items[0].SubGroup >= 30) return false;
                        return true;
                    };
                resGroupHours4.Filter     =
item => item.GroupID != 12815;
                mSoft.Add(resGroupHours4);

                RestrictGroupFill
resGroupFill  =  new  RestrictGroupFill(2000,
"RestrictGroupFill");
                resGroupFill.FilterItem   =
item                                    =>
!item.FlowParam.ContainsValue("Онлайн");
                mHard.Add(resGroupFill);

                RestrictDayDiscipline
resDayDisc = new RestrictDayDiscipline(500);
                mSoft.Add(resDayDisc);

                RestrictEmployeeFill
resEmpFill   =   new   RestrictEmployeeFill(2,
"RestrictEmployeeFill");
                mSoft.Add(resEmpFill);

                RestrictEmployeeHours
resEmpHours = new RestrictEmployeeHours(500,
"RestrictEmployeeHours", 9);
                mSoft.Add(resEmpHours);

                List<FlowData>     df     =
Flows.Where(f   =>   (f.CountHours   ==   1)   &&
(f.DayTimes.First().Day.Cross.Count == 0) &&
(f.Items[0].Group.Value<string>("Программа")
== "1")).ToList();

                foreach   (FlowData   f   in
Flows.Where(f   =>   (f.CountHours   ==   1)   &&
(f.DayTimes.First().Day.Cross.Count  >  0)  &&
(f.Items[0].Group.Value<string>("Программа")
== "1")))
                {
                    foreach(FlowData  fx  in
df)
                    {
                        if ((fx.TypeFlow ==
f.TypeFlow) &&
                            (fx.Items.Count
== f.Items.Count) &&
(fx.Teachers.Count == f.Teachers.Count))
                            {
                                bool all_groups
= true;
                                bool
all_teachers = true;
foreach(KeyValuePair<int,   Teacher>   t   in
f.Teachers)
                                {
all_teachers      =      all_teachers      &&
(fx.Teachers.ContainsKey(t.Key));
                                }
foreach(FlowItem gi in f.Items)
                                {
                                    all_groups
=   all_groups   &&   fx.Items.Exists(fg   =>
(fg.CountStudent == gi.CountStudent) &&
(fg.Discipline == gi.Discipline) &&
(fg.Group  ==  gi.Group)  &&  (fg.SubGroup  ==
gi.SubGroup));
                                }
                                if  (all_groups
&& all_teachers)
                                {
f.Linked.Add(fx);
fx.Linked.Add(f);
                                }
                            }
                        }
                    }
                }
                public                override
IReadOnlyDictionary<int,  LectureTime>  Times
=> mTimes;
                public                override
IReadOnlyDictionary<int,  Group>  Groups  =>
mGroups;
                public                override
IReadOnlyDictionary<int,   Room>   Rooms   =>
mRooms;
                public                override
IReadOnlyDictionary<int, Teacher> Teachers =>
mTeachers;
                public                override
IReadOnlyDictionary<int,          Discipline>
Disciplines => mDisciplines;
                public override List<FlowData>
Flows => mFlows;
                public                override
IReadOnlyList<IRestrict> Hard => mHard;
                public                override
IReadOnlyList<IRestrict> Soft => mSoft;
            }
        }
```

Приложение 3 – Исходный код html-файл для просмотра расписания

```
<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="utf-8" />
    <title>Просмотр расписания</title>
    <script type="text/javascript">
        function $get(id) {
            return document.getElementById(id);
        }

        function Viewer(xmlFile) {
            var reader = new FileReader();
            var obj = this;
            this.clearItems();
            reader.onload = function () {
                var p = new DOMParser();
                obj._doc = p.parseFromString(this.result, "application/xml");
                obj._init();
            };
            reader.readAsText(xmlFile.files[0]);
        }
        Viewer.prototype = {
            _day:  new  Array("Monday",  "Tuesday",  "Wednesday",  "Thursday",  "Friday",
"Saturday"),
            _init: function () {
                var lg = $get("listGroups");
                var lr = $get("listRooms");
                var lt = $get("listTutors");
                lg.innerHTML = "";
                lr.innerHTML = "";
                lt.innerHTML = "";
                var gid = new Array();
                var rid = new Array();
                var tid = new Array();
                var s = this._doc.getElementsByTagName("s");
                for (var i = 0; i < s.length; i++) {
                    if (rid.indexOf(s.item(i).attributes.room.value) == -1) {
                        var o = document.createElement("OPTION");
                        o.value = s.item(i).attributes.room.value;

o.appendChild(document.createTextNode(s.item(i).attributes.roomNumber.value));
                            lr.options.add(o);
                            rid.push(o.value);
                        }
                        var g = s.item(i).getElementsByTagName("g");
                        for (var j = 0; j < g.length; j++) {
                            if (gid.indexOf(g.item(j).attributes.id.value) == -1) {
                                var o = document.createElement("OPTION");
                                o.value = g.item(j).attributes.id.value;

o.appendChild(document.createTextNode(g.item(j).attributes.code.value      +      '      ['      +
g.item(j).attributes.id.value + ']'));
                                lg.options.add(o);
                                gid.push(o.value);
                            }
                        }
                        var t = s.item(i).getElementsByTagName("t");
                        for (var j = 0; j < t.length; j++) {
                            if (tid.indexOf(t.item(j).attributes.id.value) == -1) {
                                var o = document.createElement("OPTION");
                                o.value = t.item(j).attributes.id.value;

o.appendChild(document.createTextNode(t.item(j).attributes.name.value));
                                lt.options.add(o);
                                tid.push(o.value);
                            }
                        }
                    }
```

```
                    this.sortSelect(lg);
                    this.sortSelect(lr);
                    this.sortSelect(lt);
                },
                sortSelect: function (s) {
                    var o = new Array();
                    while (s.options.length > 0) {
                        o.push(s.options.item(0));
                        s.options.remove(0);
                    }
                    o.sort(function (r1, r2) { return
r1.firstChild.nodeValue.localeCompare(r2.firstChild.nodeValue); });
                    for (var i = 0; i < o.length; i++) {
                        s.appendChild(o[i]);
                    }
                    s.value = o[0].value;
                },
                showGroupTimetable: function () {
                    var id = $get("listGroups").value;
                    this.clearItems();
                    var s = this._doc.getElementsByTagName("s");
                    for (var i = 0; i < s.length; i++) {
                        var g = s.item(i).getElementsByTagName("g");
                        for (var j = 0; j < g.length; j++) {
                            if (g.item(j).attributes.id.value == id) {
                                this.generateItem(s.item(i), id);
                            }
                        }
                    }
                    this.sortItems();
                },
                showTutorTimetable: function () {
                    var id = $get("listTutors").value;
                    this.clearItems();
                    var s = this._doc.getElementsByTagName("s");
                    for (var i = 0; i < s.length; i++) {
                        var g = s.item(i).getElementsByTagName("t");
                        for (var j = 0; j < g.length; j++) {
                            if (g.item(j).attributes.id.value == id) {
                                this.generateItem(s.item(i));
                            }
                        }
                    }
                    this.sortItems();
                },
                showRoomTimetable: function () {
                    var id = $get("listRooms").value;
                    this.clearItems();
                    var s = this._doc.getElementsByTagName("s");
                    for (var i = 0; i < s.length; i++) {
                        if (s.item(i).attributes.room.value == id) {
                            this.generateItem(s.item(i));
                        }
                    }
                    this.sortItems();
                },
                generateItem: function (s, gid) {
                    gid = gid || 0;
                    var lt = s.getElementsByTagName("lt");
                    var tut = s.getElementsByTagName("t");
                    var gr = s.getElementsByTagName("g");
                    var defects = s.getElementsByTagName("defect");
                    for (var i = 0; i < lt.length; i++) {
                        var e = document.createElement("div");
                        var px = lt.item(i).attributes.start.value.split(":")
                        e.tid = parseInt(px[0]) * 100 + parseInt(px[1]);
                        e.style.border = "1px dotted black";
                        e.style.margin = "3px";
                        e.style.padding = "4px";

$get("tSchedule").rows.item(0).cells.item(Math.floor(s.attributes.day.value / 100) -
1).appendChild(e);

                        var dayName = document.createElement("div");

dayName.appendChild(document.createTextNode(lt.item(i).attributes.start.value + " - " +
lt.item(i).attributes.finish.value + ", " + s.attributes.dayName.value));
                        e.appendChild(dayName);
```

```
                            var type = document.createElement("div");
                            type.appendChild(document.createTextNode(s.attributes.type.value + ", "
+ s.attributes.roomNumber.value));
                            e.appendChild(type);

                            var tut_list = new Array();
                            for (var j = 0; j < tut.length; j++) {
                                tut_list.push(tut.item(j).attributes.name.value);
                            }

                            var tut_div = document.createElement("div");
                            tut_div.appendChild(document.createTextNode(tut_list.join(", ")));
                            e.appendChild(tut_div);

                            for (var j = 0; j < gr.length; j++) {
                                if ((gid == 0) || (gid == gr.item(j).attributes.id.value)) {
                                    var disc = document.createElement("div");
                                    var t = "";
                                    var ge = gr.item(j).attributes;
                                    if (gid != 0) {
                                        t = ge.discname.value;
                                        if (ge.sub.value != 0) {
                                            t += " [" + ge.sub.value + "]";
                                        }
                                    } else {
                                        t = ge.discname.value + " [" + ge.code.value;
                                        if (ge.sub.value != 0) {
                                            t += ", " + ge.sub.value;
                                        }
                                        t += "]";
                                    }
                                    disc.appendChild(document.createTextNode(t));
                                    e.appendChild(disc);
                                }
                            }


                            for (var j = 0; j < defects.length; j++) {
                                var def_div = document.createElement("div");
                                var di = defects.item(j);
                                if (di.attributes.hard.value == "False") {
                                    def_div.style.backgroundColor = "blue";
                                } else {
                                    def_div.style.backgroundColor = "red";
                                }
                                def_div.style.color = "white";
                                def_div.style.marginTop = "3px";
                                def_div.style.padding = "3px";

def_div.appendChild(document.createTextNode(di.attributes.name.value));
                                e.appendChild(def_div);
                            }
                        }

                    },
                    clearItems: function () {
                        var e = $get("tSchedule").rows.item(0).cells;
                        for (var i = 0; i < e.length; i++) {
                            e.item(i).innerHTML = "";
                        }
                    },
                    sortItems: function () {
                        var e = $get("tSchedule").rows.item(0).cells;
                        for (var i = 0; i < e.length; i++) {
                            var ex = new Array();
                            while (e.item(i).childNodes.length > 0) {
                                ex.push(e.item(i).childNodes.item(0));
                                e.item(i).removeChild(e.item(i).firstChild);
                            }
                            ex.sort(function (e1, e2) {
                                var r = e1.tid - e2.tid;
                                return r;
                            });
                            for (var j = 0; j < ex.length; j++) {
                                e.item(i).appendChild(ex[j]);
                            }
                        }
                    }
                }
```

```
            function Load() {
                try {
                    Viewer.current = new Viewer($get("iFile"));
                } catch (e) {
                    alert(e.message);
                }
            }
        </script>
    </head>
    <body>
        <table cellpadding="3" border="1" cellspacing="0" style="width: 100%">
            <tr>
                <td colspan="2">
                    Файл:     <input     type="file"     width="200"     id="iFile"/>     <a
href="javascript:Load()">Загрузить файл</a>
                </td>
            </tr>
            <tr>
                <td width="150">Список</td>
                <td>Расписание</td>
            </tr>
            <tr>
                <td id="tList" style="vertical-align: top; white-space:nowrap">
                    <div style="padding: 10px">
                        <div>Группы</div>
                        <div><select id="listGroups"></select></div>
                        <a       href="javascript:Viewer.current.showGroupTimetable()">Показать
расписание группы</a>
                    </div>
                    <div style="padding: 10px">
                        <div>ППС</div>
                        <div><select id="listTutors"></select></div>
                        <a       href="javascript:Viewer.current.showTutorTimetable()">Показать
расписание ППС</a>
                    </div>
                    <div style="padding: 10px">
                        <div>Аудитории</div>
                        <div><select id="listRooms"></select></div>
                        <a        href="javascript:Viewer.current.showRoomTimetable()">Показать
расписание аудитории</a>
                    </div>
                </td>
                <td style="vertical-align: top">
                    <table border="1" style="border-collapse: collapse; width: 100%">
                        <thead>
                            <tr>
                                <th style="width:16.7%">Понедельник</th>
                                <th style="width:16.7%">Вторник</th>
                                <th style="width:16.7%">Среда</th>
                                <th style="width:16.7%">Четверг</th>
                                <th style="width:16.7%">Пятница</th>
                                <th>Суббота</th>
                            </tr>
                        </thead>
                        <tbody id="tSchedule">
                            <tr>
                                <td style="vertical-align: top"></td>
                                <td style="vertical-align: top"></td>
                                <td style="vertical-align: top"></td>
                                <td style="vertical-align: top"></td>
                                <td style="vertical-align: top"></td>
                                <td style="vertical-align: top"></td>
                            </tr>
                        </tbody>
                    </table>
                </td>
            </tr>
        </table>
    </body>
    </html>
```

Приложение И – Исходный код хранимой процедуры для импорта xml-файла с расписанием в БД образовательного портала

```
    CREATE PROCEDURE [dbo].[procImportSchedule]
      @X xml,
      @id int
    AS
    BEGIN

    DECLARE @T table (dayid int, weekt int, timeid  int, roomid int, typel
nvarchar(50), flow uniqueidentifier, tid int, gid int, sub int, disc int)
    INSERT INTO @T
    SELECT CEILING(S.S.value('@day', 'int') / 100),
           S.S.value('@day', 'int') % 10,
           LT.lt,
           S.S.value('@room', 'int'),
           S.S.value('@type', 'nvarchar(50)'),
           CASE WHEN S.S.value('count(g)', 'int') > 1 THEN NEWID() ELSE NULL
END,
           T.id,
           G.id,
           G.sub,
           G.disc
      FROM @X.nodes('s[@const="False"]') AS S(S)
             OUTER APPLY (SELECT X.X.value('@id', 'int') AS id
                          FROM S.S.nodes('t') AS X(X)) T
             CROSS APPLY (SELECT X.X.value('@id', 'int') AS id,
                                 X.X.value('@sub', 'int') AS sub,
                                 X.X.value('@disc', 'int') AS disc
                          FROM S.S.nodes('g') AS X(X)) G
             CROSS APPLY (SELECT TM.T.value('@id', 'int') AS lt
                          FROM S.S.nodes('lt') AS TM(T)) LT

    BEGIN TRY
      BEGIN TRAN
      DELETE FROM dbo.LectureSchedule
        WHERE ScheduleManagerID = @id AND DisciplineID <> 30

      INSERT  INTO  dbo.LectureSchedule(DisciplineID,  EmployeeID,  GroupID,
FlowID, KindLectureID, LectureRoomID, LectureTimeID, WeekDayID, WeekTypeID,
ScheduleManagerID, SubGroupID)
        SELECT T.disc,
               T.tid,
               T.gid,
               T.flow,
               CASE WHEN T.typel = 'Lecture' THEN 4
                    WHEN T.typel = 'Practice' THEN 7
                    WHEN T.typel = 'Laboratory' THEN 6 END,
               T.roomid,
               T.timeid,
               WD.WeekDayID,
               CASE WHEN T.weekt = 0 then 5 when T.weekt = 1 then 4 when
T.weekt = 2 then 6 end,
               @id,
               CASE WHEN T.sub = 0 OR T.sub > 30 THEN 4
                    WHEN T.sub = 1 THEN 1
                    WHEN T.sub = 2 THEN 2
                    WHEN T.sub = 3 THEN 6
                    WHEN T.sub = 4 THEN 7
                    WHEN T.sub = 5 THEN 8
                    WHEN T.sub = 6 THEN 1
                    WHEN T.sub = 7 THEN 2
```

```
                    WHEN T.sub = 8 THEN 6
                    WHEN T.sub = 9 THEN 7
                    WHEN T.sub = 10 THEN 1 END
          FROM @T T INNER JOIN dbo.WeekDay WD
                 ON T.dayid = WD.DayID
    COMMIT
  END TRY
  BEGIN CATCH
    DECLARE @msg nvarchar(4000) =  N'Ошибка при импорте расписания - ' +
ERROR_MESSAGE()
    ROLLBACK TRAN;
    THROW 50200, @msg, 1;
  END CATCH
  END
```