Неделя 8. Лаба 4. Git и GitHub: репозиторий сервиса и работа через PR

Часть 2 из 2: разделы «7. Шаблон PR и Dependabot» — «⭐ Дополнительно». Первая часть лабы — в материалах предыдущей недели; перед продолжением выполните подготовку ниже.

Цель: репозиторий с защищённым `main`, задачи на доске, работа через ветки и PR с ревью кода, разрешение конфликта, первый релиз. **Время:** 90 мин. **Теория:** лекция 4. **Инструменты:** Git, GitHub, Visual Studio 2026 (Git Changes), PowerShell 7, браузер.

Формат: пары, два этапа примерно по 45 минут.

- **Этап 1 — репозиторий А.** А — автор, В — ревьюер. Выполняете пп. 1–8.
- **Этап 2 — репозиторий В.** Роли меняются, повторяете пп. 1–4 и 7–8.
- **Конфликт (п. 6)** делаете один раз, в репозитории А, вдвоём.

7. Шаблон PR и Dependabot

```
git switch main; git pull
git switch -c chore/2-repo-config
New-Item -ItemType Directory -Force .github | Out-Null
Copy-Item $K\.github\pull_request_template.md, $K\.github\dependabot.yml .github\
git add .github
git commit -m "chore: add PR template and Dependabot config"
git push -u origin chore/2-repo-config
```

Прочитайте оба файла перед коммитом. PR с `Closes #2` → ревью → merge.

Зачем: шаблон напоминает автору проверить запуск и тесты; Dependabot сам предлагает обновления NuGet-пакетов и образов с уязвимостями.

8. Код через PR: переименование проекта

Задача: `PUT /api/projects/{id}` с телом `{ "name": "Новое имя" }` переименовывает проект. Пустое имя → **400**, нет проекта → **404**, имя занято → **409**.

```
git switch main; git pull
git switch -c feat/3-rename-project
```

По чек-листу из лекции 1:

1. **Data:** метод `Task<Project?> RenameAsync(int id, string name, CancellationToken ct)` в `IProjectRepository` и в обеих реализациях. `InMemoryProjectRepository`: внутри `lock (store.Sync)` найти индекс проекта по id (`FindIndex`), нет → `null`, есть → заменить копией `store.Projects[index] with { Name = name }`. `ProjectRepository` (PostgreSQL) — SQL `UPDATE projects SET name = $2 WHERE id = $1 RETURNING id, name, created_at`, по образцу `GetAsync`: строки нет → `null`.
2. **Services:** `ProjectService.RenameAsync` вызывает репозиторий и, если проект переименован, пишет в лог — по образцу `CreateAsync`.
3. **Controllers:** внизу `ProjectsController.cs` — запись `RenameProjectRequest` по образцу `CreateProjectRequest`. Действие `[HttpPut("{id:int}")]`: имя занято (`projects.NameExistsAsync`) → 409, как в `Create`; `RenameAsync` вернул `null` → `NotFound()`; иначе вернуть проект. Над действием — XML-комментарий: `/// <summary>`, `/// <param>` и `/// <response>` для 200, 400, 404, 409, чтобы операция была описана в Swagger.
4. **Tests** в `ProjectsApiTests`: успешное переименование и 409 при занятом имени (`_client.PutAsJsonAsync`). Тесты работают с PostgreSQL, поэтому сначала: `docker compose up -d --wait db`.

PR с `Closes #3`. **В ревьюит по таблице из лекции 4, п. 7:** слои, оба хранилища, тесты, SQL-параметры, секреты. Минимум один содержательный комментарий.

9. Релиз

```
git switch main; git pull
git tag v0.1.0
git push origin v0.1.0
```

Releases → *Draft a new release* → тег `v0.1.0` → **Generate release notes** → Publish.

Сдача

- [ ] Ссылка на репозиторий (преподаватель добавлен в collaborators)
- [ ] Ruleset на `main` и скриншот отклонённого прямого push
- [ ] Три issue закрыты через PR, доска со всеми в Done
- [ ] В PR есть комментарии ревьюера и исправления по ним
- [ ] Коммит разрешения конфликта в `git log --oneline --graph`
- [ ] `PUT /api/projects/{id}` работает, тесты зелёные
- [ ] Релиз `v0.1.0` с автоматическими release notes

☆ Дополнительно

Добавьте `.github/CODEOWNERS` со строкой `* @<ник-партнёра>` и включите в ruleset *Require review from Code Owners*. Проверьте, что ревьюер назначается автоматически.