Неделя 1. Лаба 1. ASP.NET: запускаем сервис задач и добавляем свою функцию

Часть 1 из 2: разделы «Подготовка: что установить» — «4. Данные в памяти».

Цель: открыть проект в Visual Studio, запустить сервис и обратиться к его API из Postman, пройти запрос отладчиком через все слои и добавить новую функцию с тестами. **Время:** 120 мин. **Теория:** лекция 1. **Инструменты:** Visual Studio 2026, Postman, браузер.

Подготовка: что установить

Для этой лабы нужны только две программы. Docker, Git и остальное поставим позже — в той лабе, где они впервые понадобятся.

Программа	Зачем	Как поставить
Visual Studio 2026 Community	Открыть проект, запустить, отладить, прогнать тесты	visualstudio.microsoft.com → *Community*. В установщике отметьте набор **«ASP.NET и разработка веб-приложений»** — вместе с ним встанет .NET 10 SDK. Уже установлена? *Visual Studio Installer* → *Изменить* → отметьте этот набор
Postman	Отправлять HTTP-запросы к сервису и смотреть ответы	postman.com/downloads. Можно работать без входа в аккаунт

Требования к ПК на весь курс: Windows 10/11 x64, 16 ГБ RAM (минимум 8), 40 ГБ свободного места.

1. Проект

Распакуйте `starter.zip` в папку `C:\dev`. Внутри архива уже есть папка `demo-api`, поэтому получится `C:\dev\demo-api` — **ваш репозиторий на весь курс**. К лабе 8 в нём появятся Dockerfile, compose, CI/CD и мониторинг.

Откройте `C:\dev\demo-api\DemoApi.slnx` в Visual Studio (*Файл → Открыть → Проект или решение*). В *Обозревателе решений* раскройте `src/DemoApi` и найдите папки из лекции: `Controllers`, `Services`, `Domain`, `Data`. Откройте `STRUCTURE.md` в корне решения — там карта проекта: что в каждой папке и зачем.

2. Запуск

В выпадающем списке у зелёной кнопки запуска выберите профиль **http** и нажмите **F5**.

- Откроется консольное окно сервиса со строками `Хранилище: в памяти` и `Now listening on: http://localhost:8080`. Пока это окно открыто, сервис работает и ждёт запросов.
- Браузер откроет **http://localhost:8080/swagger** — описание API. К нему вернёмся в конце п. 3, сначала поработаем из Postman.

3. Работаем с API из Postman

Сервис задач — backend: страниц у него нет, он отвечает на HTTP-запросы. Postman — программа, которая такие запросы отправляет и показывает ответ. Так же к сервису обращались бы сайт или мобильное приложение.

Первый запрос. В Postman: **New → HTTP** (или `+` на панели вкладок).

1. Слева от адресной строки метод **GET**, в адресе `http://localhost:8080/api/projects`, **Send**.
2. Внизу — ответ: код **200 OK**, время, тело `[]` — проектов пока нет.

Запрос с телом. Новая вкладка, метод **POST**, адрес `http://localhost:8080/api/projects`. Вкладка **Body → raw**, справа тип **JSON**, тело:

```
{ "name": "Курс DevOps" }
```

Send → 201 Created, в ответе JSON нового проекта с `id`. На вкладке ответа **Headers** найдите `Location` — адрес созданного проекта. В консольном окне сервиса появилась строка `Создан проект 1 «Курс DevOps»`: запрос

из Postman дошёл до кода сервиса.

Дальше тем же способом (метод, адрес, тело):

#	Метод и адрес	Тело (Body → raw → JSON)	Ожидаемо
1	`POST http://localhost:8080/api/tasks`	`{ "projectId": 1, "title": "Написать Dockerfile" }`	**201**
2	`GET http://localhost:8080/api/tasks?projectId=1`	—	**200**, одна задача
3	`PATCH http://localhost:8080/api/tasks/1/status`	`{ "status": "InProgress" }`	**200**
4	`GET http://localhost:8080/api/projects/1`	—	**200**

Сохраните запросы в коллекцию: **Save → New Collection** `demo-api`. Она пригодится в следующих лабах, когда сервис переедет в контейнер.

Ошибки — запишите код и тело ответа:

- `POST /api/projects` с `{ "name": "" }` → **400**: что в `errors`?
- ещё раз `POST /api/projects` с `{ "name": "Курс DevOps" }` → **409**: что в `detail`?
- создайте новую задачу и сразу переведите её в `Done` → **409**: что в `detail`?
- `GET /api/tasks/999` → **404**.

Другие клиенты того же API. Сервису всё равно, кто спрашивает:

- **Swagger** (`http://localhost:8080/swagger`, открылся при F5) — описание всех эндпоинтов: что делает операция, какие поля, какие коды ответа. Раскройте `POST /api/projects` → **Try it out** → **Execute** — это тот же запрос, что вы отправляли из Postman.
- **`src/DemoApi/DemoApi.http`** в Visual Studio — запросы, которые лежат в репозитории рядом с кодом. Кнопка **Send Request** над каждым.
- Весь API сразу в Postman: **Import → Link** → `http://localhost:8080/openapi/v1.json` — появится коллекция со всеми запросами сервиса.

Дальше в курсе пользуйтесь тем клиентом, который удобнее.

4. Данные в памяти

Остановите сервис (**Shift+F5**) и запустите снова (**F5**). Выполните `GET /api/projects`.

Список пуст. **Почему?** Подсказка: строка `Хранилище: в памяти` в консоли и лекция, п. 9. Запишите ответ — в теме 3 мы это исправим.