**Лабораторная работа №2**

**Цель работы:** реализовать Web API приложение на языке C#, реализующее интерфейс манипулирования данными предметной области, согласно варианту.

**Теория:**

1. **CRUD**

CRUD — акроним, обозначающий четыре базовые функции, используемые при работе с базами данных: создание (англ. create), чтение (read), модификация (update), удаление (delete). Введён Джеймсом Мартином (англ. James Martin) в 1983 году как стандартная классификация функций по манипуляции данными.

В SQL этим функциям, операциям соответствуют операторы Insert (создание записей), Select (чтение записей), Update (редактирование записей), Delete (удаление записей). В некоторых CASE-средствах использовались специализированные CRUD-матрицы или CRUD-диаграммы, в которых для каждой сущности указывалось, какие базовые функции с этой сущностью выполняет тот или иной процесс или та или иная роль. В системах, реализующих доступ к базе данных через API в стиле REST, эти функции реализуются зачастую (но не обязательно) через HTTP-методы PUT, POST, GET, PATCH, DELETE.

Хотя традиционно оперирование в стиле CRUD применяется к базам данных, такой подход может быть распространён на любые хранимые вычислительные сущности (файлы, структуры в памяти, объекты). Шаблон проектирования ActiveRecord обеспечивает соответствие функций CRUD объектно-ориентированному подходу, и широко используется в различных фреймворках для доступа к базам данных из объектно-ориентированных языков программирования.

1. **Методы HTTP запроса**

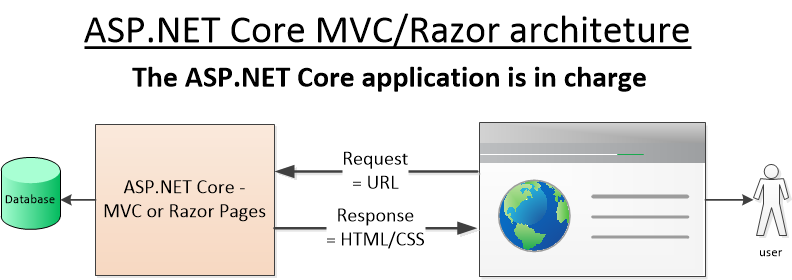
HTTP определяет множество методов запроса, которые указывают, какое желаемое действие выполнится для данного ресурса. Несмотря на то, что их названия могут быть существительными, эти методы запроса иногда называются HTTP глаголами. Каждый реализует свою семантику, но каждая группа команд разделяет общие свойства: так, методы могут быть безопасными, идемпотентными или кешируемыми.

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| GET | Запрашивает представление ресурса. Запросы с использованием этого метода могут только извлекать данные. |
| HEAD | Запрашивает ресурс так же, как и метод GET, но без тела ответа. |
| POST | Используется для отправки сущностей к определённому ресурсу. Часто вызывает изменение состояния или какие-то побочные эффекты на сервере. |
| PUT | Заменяет все текущие представления ресурса данными запроса. |
| PATCH | Используется для частичного изменения ресурса. |
| DELETE | Удаляет указанный ресурс. |
| CONNECT | Устанавливает "туннель" к серверу, определённому по ресурсу. |
| OPTIONS | Используется для описания параметров соединения с ресурсом. |
| TRACE | Выполняет вызов возвращаемого тестового сообщения с ресурса. |

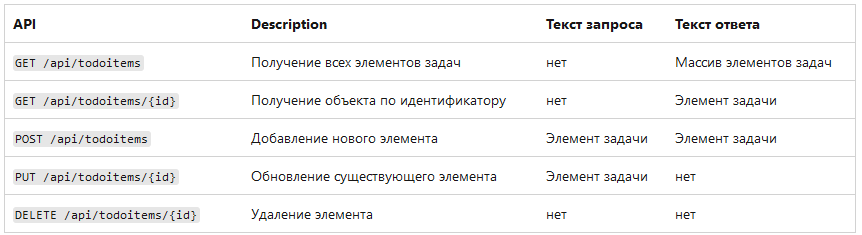
1. **Web API**

Web API представляет способ построения приложения в стиле REST (Representation State Transfer или "передача состояния представления"). REST-архитектура предполагает применение следующих методов или типов запросов HTTP для взаимодействия с сервером:

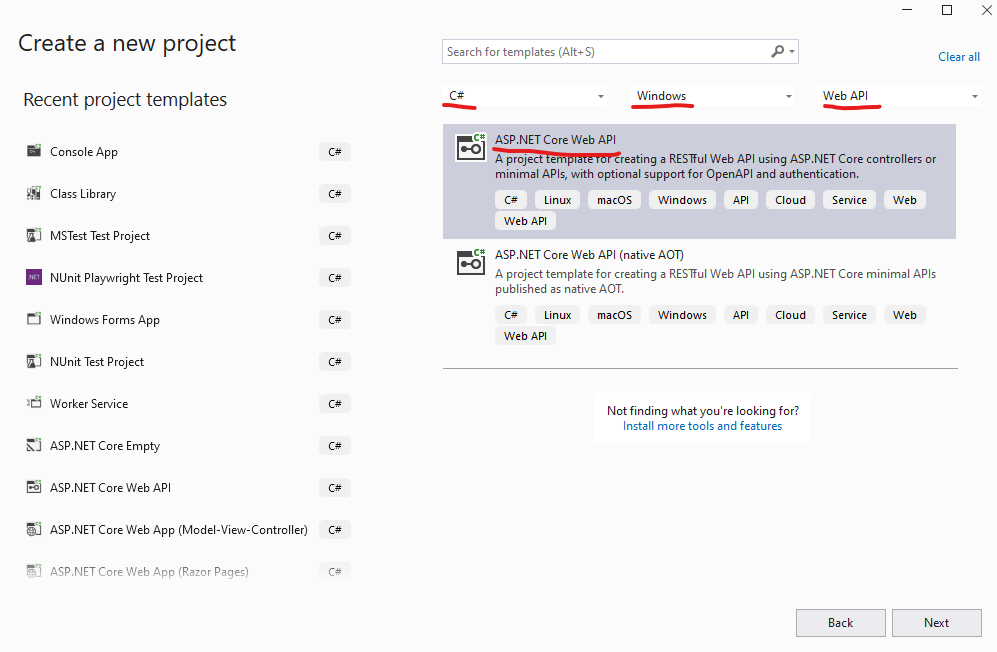
* GET (получение данных)
* POST (добавление данных)
* PUT (изменение данных)
* DELETE (удаление данных)



1. **Пример Web API интерфейса**



Шаблон проекта в Visual Studio 2022\*:



\* - обратите внимание что в русскоязычной Visual Studio может отличаться. Воспользуйтесь поиском в этом окне.

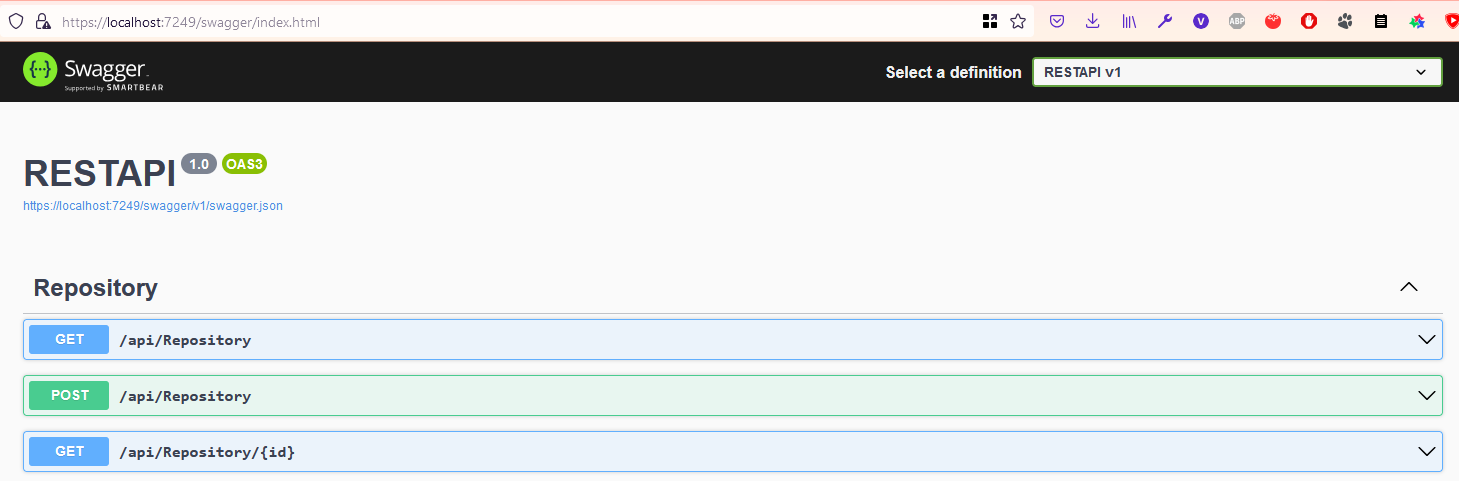
1. **Swagger**

Swagger — это инструмент, который помогает разработчикам создавать, документировать и проверять API. API — это набор правил и протоколов, которые позволяют различным системам обмениваться информацией между собой.

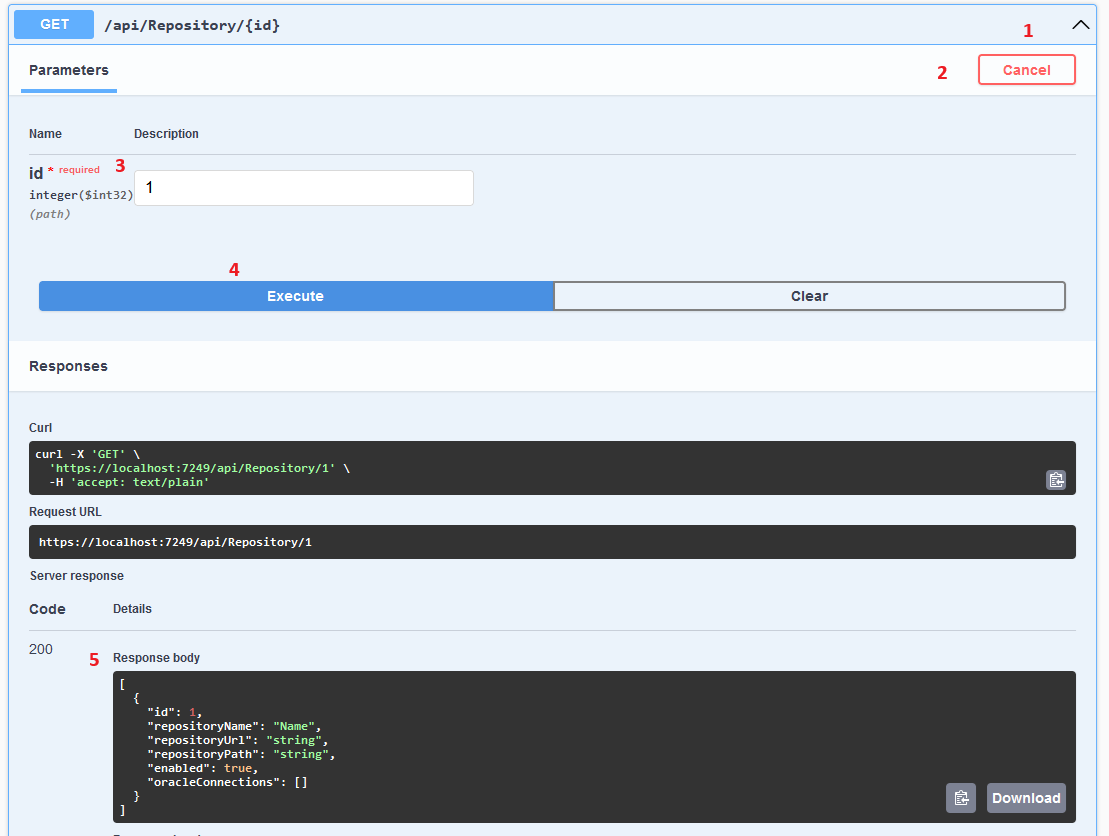
API — это общий термин, который применяется в контексте совершенно разных систем. Есть более узкий термин, REST API — это конкретный API, который используется для обмена данными между клиентами и серверами в интернете.

Часть работы с REST API — это создание описаний работы API: информации о ресурсах, параметрах запросов, возвращаемых данных, конечных точках и других важных вещах. Чтобы автоматизировать это описание, сделать его структурированным и прозрачным, разработчики используют Swagger. А системные аналитики, в свою очередь, с его помощью формируют требования к IT-системам.

Пример отображения доступных интерфейсов:



Возможность отправки запроса к API:



1 – Выбор метода

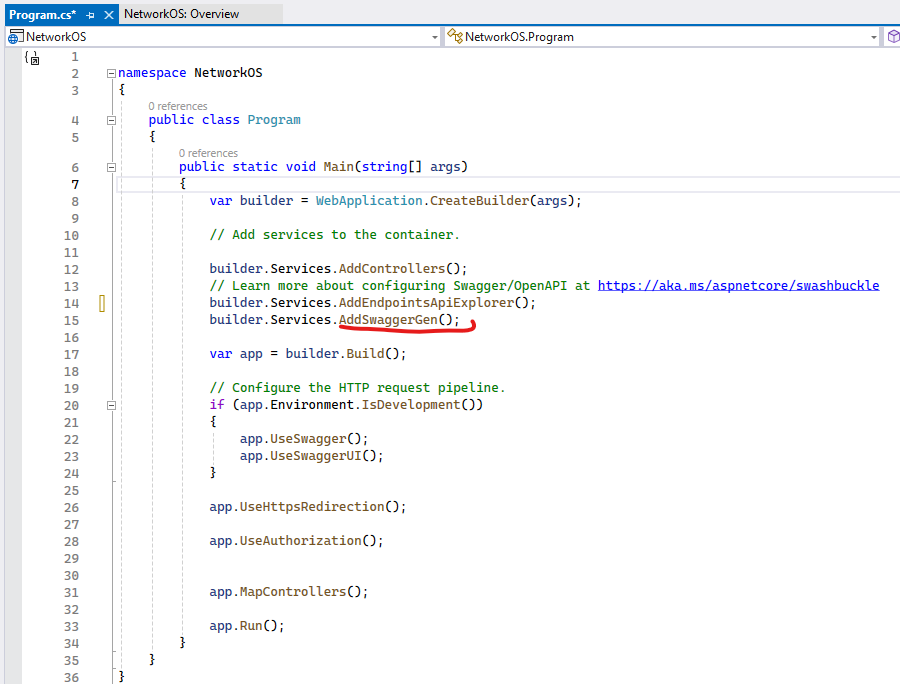
2 – открытие окна для отправки запроса

3 – ввод значений (если есть в интерфейсе)

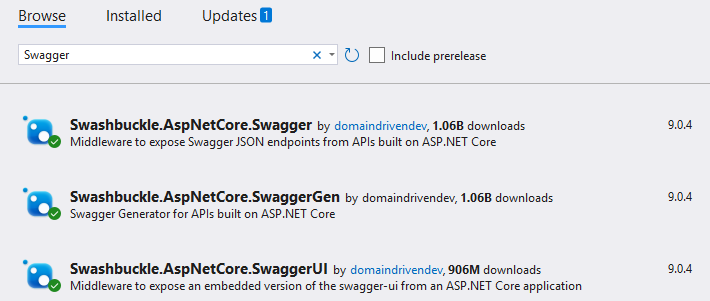
4 – отправка запроса

5 – ответ выполнения запроса.

В шаблоне Web API в файле Program.cs мы видим, что по умолчанию Swagger включен, дополнительных действий не требуется.



**Обратите внимание: начиная с версии .Net 9 для Web API проектов Swagger не включен в шаблон проекта. Для его добавления необходимо вручную добавить требуемые зависимости через nuget, и добавить Swagger в сервисы в файле Program.cs.**

****

Требуемые nugget пакеты для подключения Swagger в проект

1. **Postman**

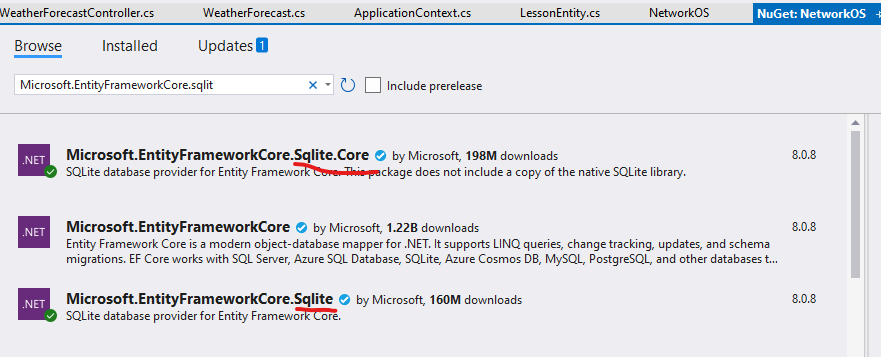
Postman — одна из самых популярных утилит, позволяющая обращаться к API.

Ключевые преимущества Postman:

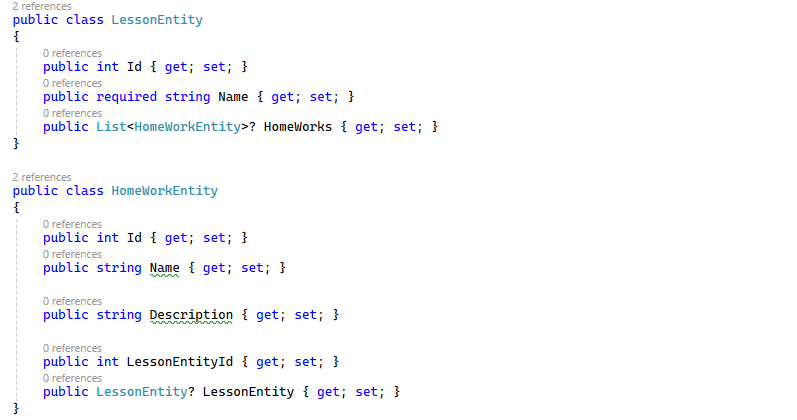
* Цена. Бесплатная версия программы имеет весь необходимый функционал. Платная стоит от 14 долларов за пользователя, но многим разработчикам достаточно и базового набора возможностей.
* Удобство. У Postman понятный графический интерфейс, в котором несложно разобраться, есть подсветка синтаксиса, возможность создавать и хранить автотесты.
* Большое комьюнити. Этот сервис — один из старейших, поэтому весь функционал хорошо задокументирован, в нем работают миллионы пользователей, которые помогают друг другу.
* Многофункциональность. Postman поддерживает все популярные типы архитектуры для API: REST, GraphQL и SOAP. Также есть возможность интегрировать его в Swagger, RAML и другие инструменты для тестирования.

1. **\*Пример подключения к бд sql lite используя entity framework**

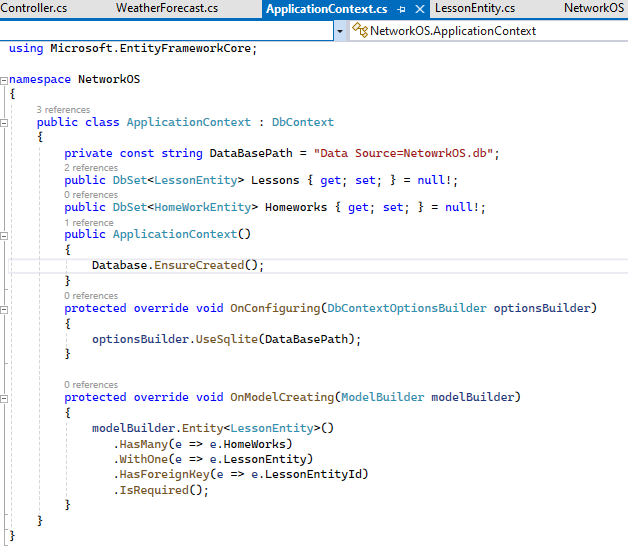
Требуется установка Nuget пакетов:



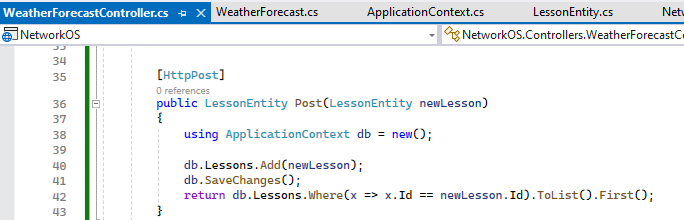
Создайте классы сущностей. Пример:



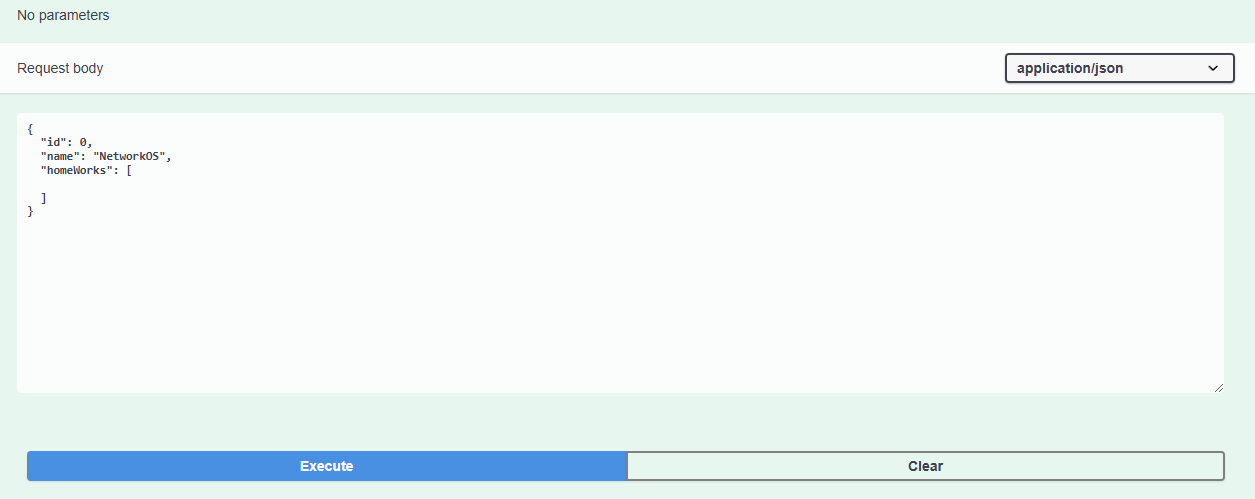
Создайте класс для работы с БД. Пример:



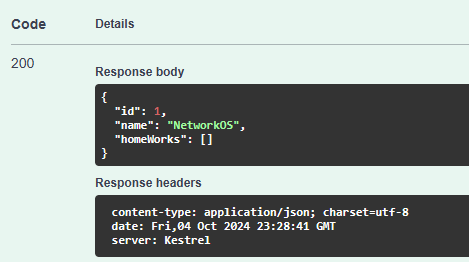
Добавьте методы обработки данных в контроллер. Пример:



Протестируйте. Пример вызова через swagger:



Ответ:



**Источники информации:**

1. <https://ru.wikipedia.org/wiki/CRUD> - CRUD
2. <https://developer.mozilla.org/ru/docs/Web/HTTP/Methods> - методы HTTP
3. <https://metanit.com/sharp/net/2.5.php> - методы HTTP
4. <https://metanit.com/sharp/aspnet6/11.1.php> - Web API Asp Net Core
5. <https://learn.microsoft.com/ru-ru/aspnet/core/tutorials/first-web-api?view=aspnetcore-8.0&tabs=visual-studio> – Пример Web API интерфейса
6. <https://practicum.yandex.ru/blog/chto-takoe-swagger/> - Swagger
7. <https://ru.hexlet.io/blog/posts/postman> - Postman
8. <https://sqlitebrowser.org/> - DB Browser for SQLite
9. <https://ru.wikipedia.org/wiki/Список_кодов_состояния_HTTP> - HTTP статусы

**Задание:**

1. Создать проект Web Api приложения используя шаблон Web API.
2. Создать подключение к базе данных используя entity framework.
3. В качестве БД предлагается использовать SQLite. Возможно использование других баз данных, при возможности демонстрации работоспособности, в том числе демонстрация данных в БД.
4. Создать сущности БД согласно варианту. Поля определить самостоятельно. Требуется предусмотреть:
   1. Хранение идентификатора сущности (Id – обязательное поле).
   2. Первая сущность должна находится со второй сущностью в связи один ко многим.
5. Реализовать обработку HTTP методов (POST, GET, PUT, **PATCH**, DELETE) для операций с данными CRUD:

|  |  |  |  |
| --- | --- | --- | --- |
| **Операция CRUD** | **Метод HTTP** | **Описание** | **Пример интерфейса** |
| Create | POST | Создание данных | POST /Lesson |
| Read | GET | Чтение данных | GET /Lesson  GET /Lesson/{id} |
| Update | PATCH / PUT | Обновление данных | PUT /Lesson/{id}  PATCH /Lesson/{id} |
| Delete | DELETE | Удаление данных | DELETE /Lesson/{id} |

1. Предусмотреть корректное возвращение статусов HTTP: в случае успешного выполнения (200), в случае возникновения ошибок (HTTP error 404, 500).
2. Продемонстрировать работу каждого из методов используя Swagger.
3. Продемонстрировать работу каждого из методов используя Postman.
4. Уметь демонстрировать работу методов используя средства разработчика в браузере.
5. Продемонстрировать изменения в БД при работе с данными (для SQL Lite можно использовать онлайн обозреватель <https://inloop.github.io/sqlite-viewer/> или утилиту DB Browser for SQLite).
6. Продемонстрировать модель данных.
7. Продемонстрировать работу каскадного удаления.

**Варианты предметной области:**

1. Библиотека и книги.
2. Аэропорты и самолеты.
3. Парковка и автомобили.
4. Университет и кафедры.
5. Папка и файлы.
6. ПК и ПО.
7. Банк и пользователи.
8. Организация и сотрудники.
9. Свой вариант.

**Контрольные вопросы**

1. Назовите CRUD операции и HTTP методы? Для чего нужен каждый из методов.
2. В чем отличие методов PUT и PATCH?
3. Назвать распространённые статусы HTTP.
4. Как в коде определяется тип метода (POST, GET, …)?
5. Объяснить структуру проекта.