**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное автономное образовательное учреждение

высшего образования «Уральский федеральный университет

имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий - РТФ

Центр ускоренного обучения

Отчет по лабораторной работе № 6

по дисциплине «Технологии Программирования»

Старший преподаватель: Н.А. Архипов

Студент гр. РИВ-220938у М.С. Ермаков

Екатеринбург 2023

**1. Цель работы:**

**Цель:** разработать REST full service

**Учебные вопросы:**

1) Разработка Spring Boot 2 приложения.

2) Подключение базы данных h2, создание таблицы и наполнение данными.

3) Настройка IntelliJ IDEA.

4) Описание REST full сервиса.

5) Оформить отчет и прикрепить его на сайте https://edu.itlearn.ru/ в соответствующем курсе, в соответствующем разделе.

**2. Описание задачи:**

Реализация дополнительного функционала

Необходимо самостоятельно доработать приложение:

1. Доработайте методы, возвращающие значение, чтобы возвращался ответ об успехе или неуспехе завершения операции.

2. Создайте вторую таблицу в базе данных «Учебный дисциплины» и опишите для нее необходимые контроллер, сущность, DAO и сервисный слой.

3. Прикрепите скриншоты выполненного ручного тестирования в Postman.

Вопросы

Письменно ответьте на вопросы, если ваш номер в списке группы четный, то нужно ответить на четные вопросы, если нечетный, то нужно ответить на нечетные вопросы.

1. Чем отличаются реляционные БД от документно-ориентированных, приведите примеры СУБД обоих типов.

2. Что из себя представляют подходы API-First и Code-First?

3. Что такое Swagger?

4. Какие существуют виды интеграция информационных систем? Кратко опишите каждый.

5. Чем отличаются SOAP и REST?

6. Что такое gRPC, и какая разница между ним и REST?

**3. Ход выполнения:**

**CREATE TABLE cards (**

**id int NOT NULL AUTO\_INCREMENT,**

**name varchar(30),**

**race varchar(30),**

**radtype varchar(30),**

**rarity varchar(30),**

**cost int,**

**PRIMARY KEY (id)**

**);**

Это скрипт для создания бд CARDS

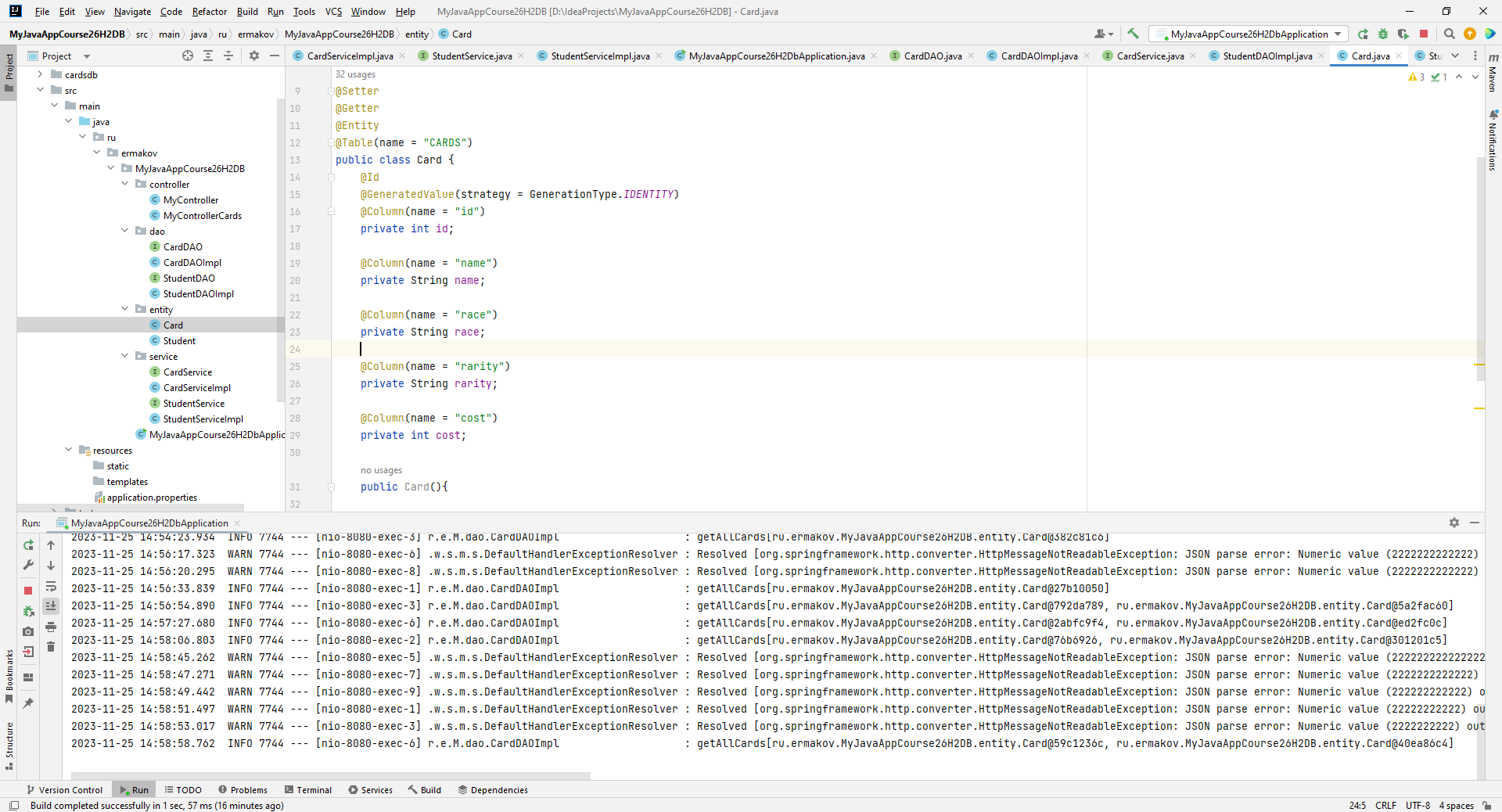


Рисунок 1 – Структура проекта

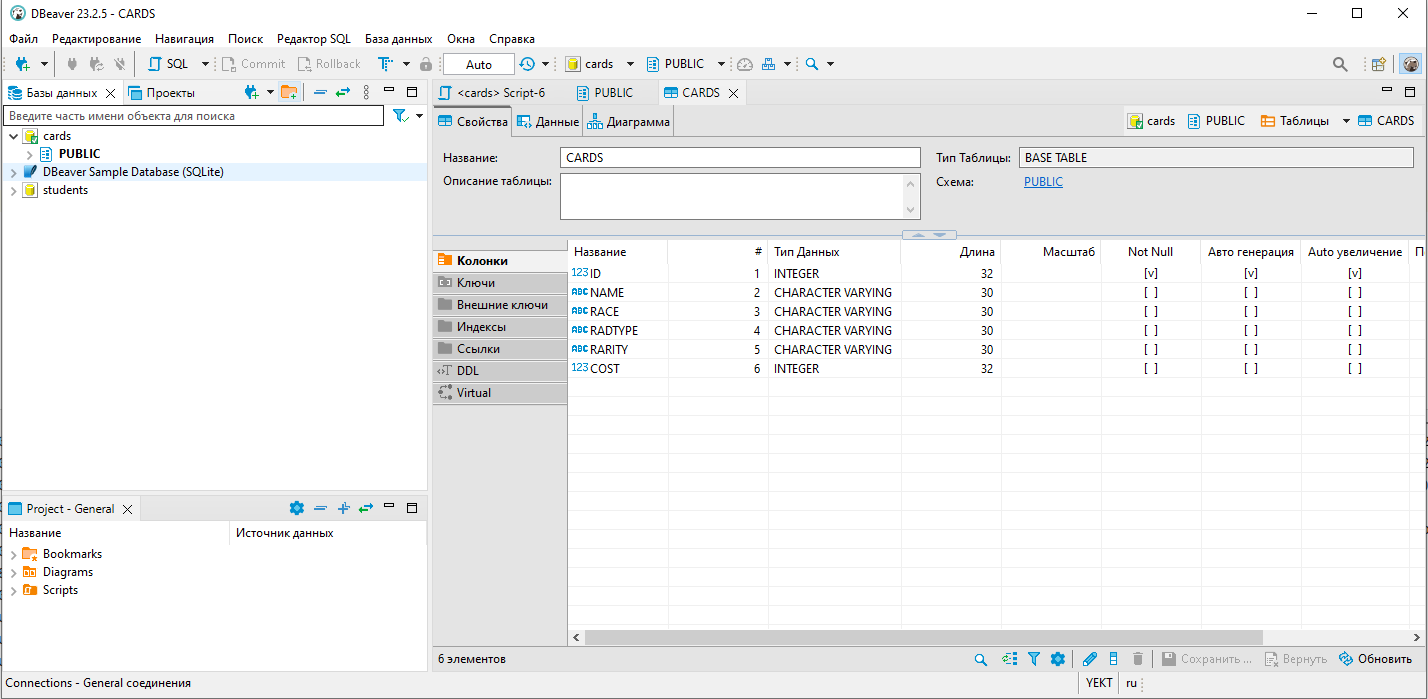


Рисунок 2 – Создание своей базы данных cards в DBveaver через бд H2

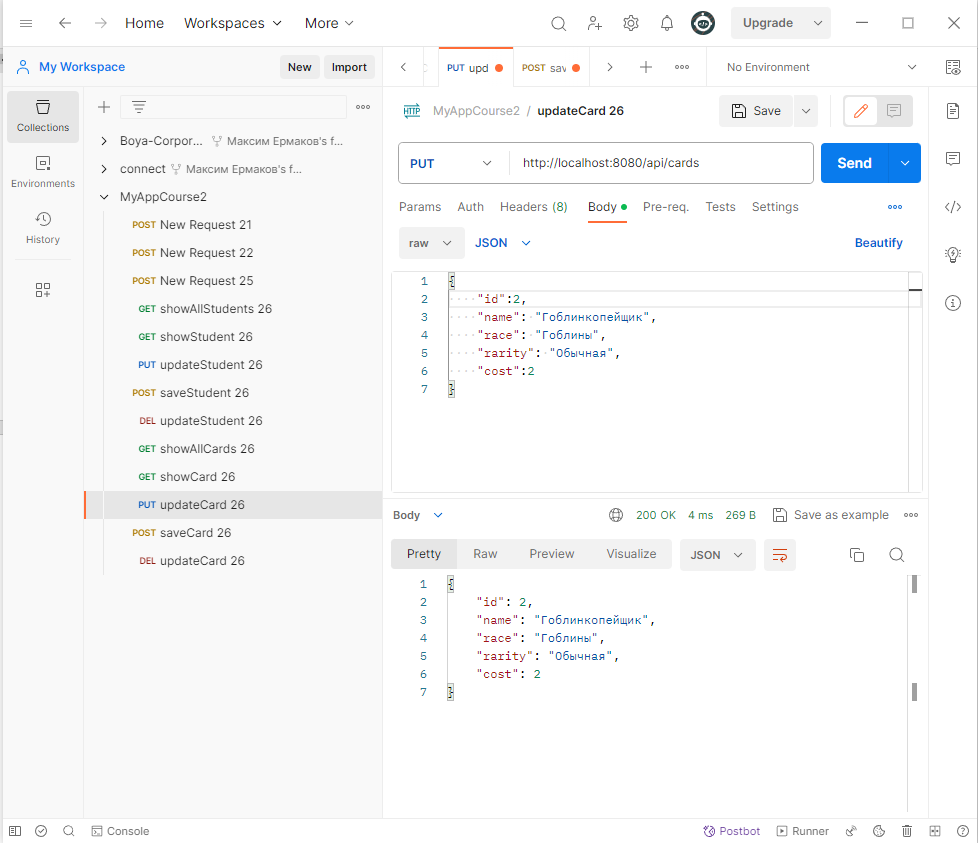


Рисунок 3 – Список запросов через Postman для командованием базой данных

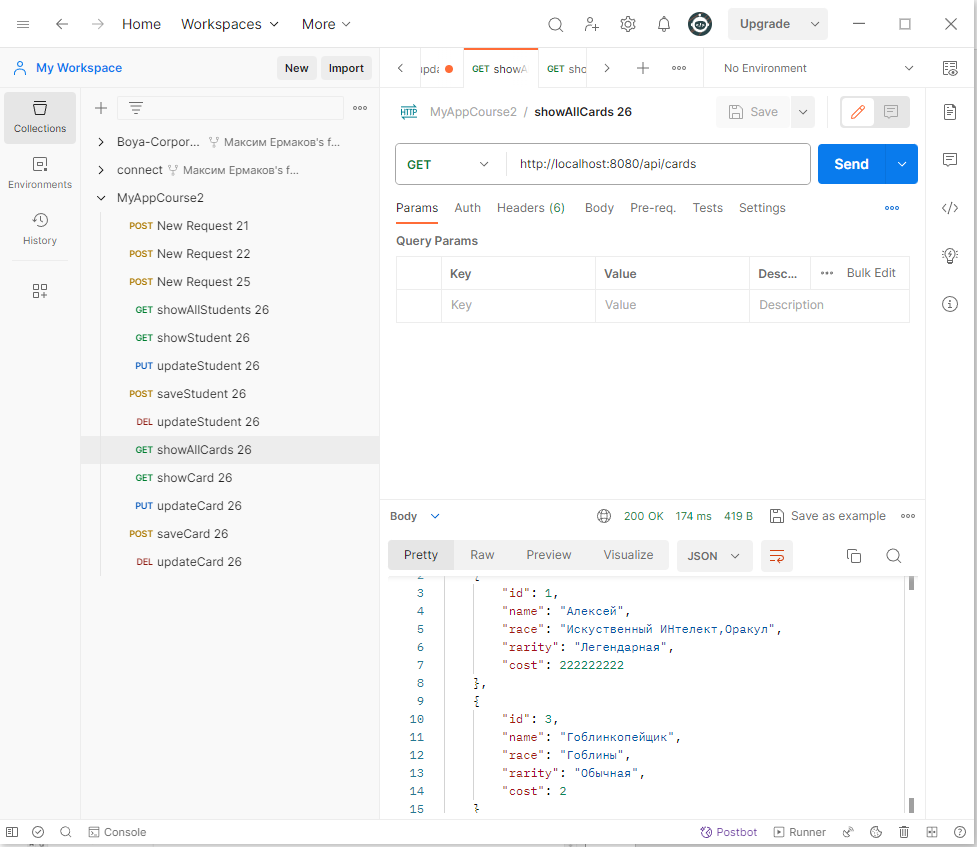


Рисунок 4 – Тестирование через Postman #1 – показать все объекты в БД

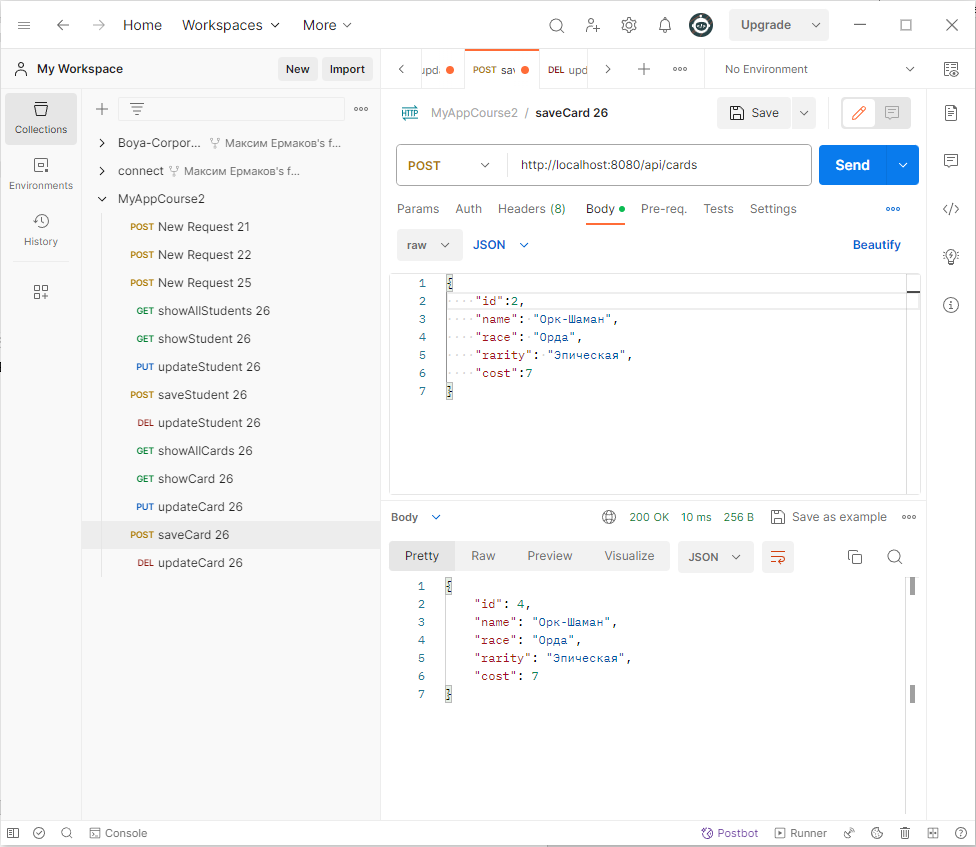


Рисунок 5 – Тестирование через Postman #2 – создать новую карту в БД

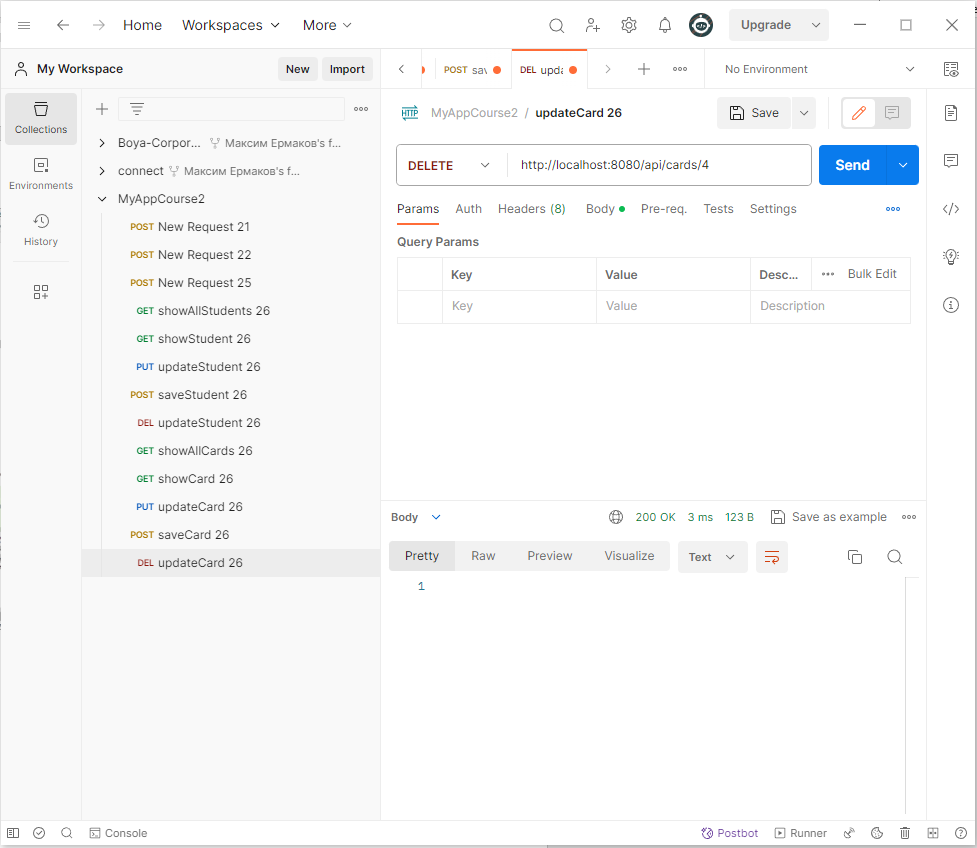
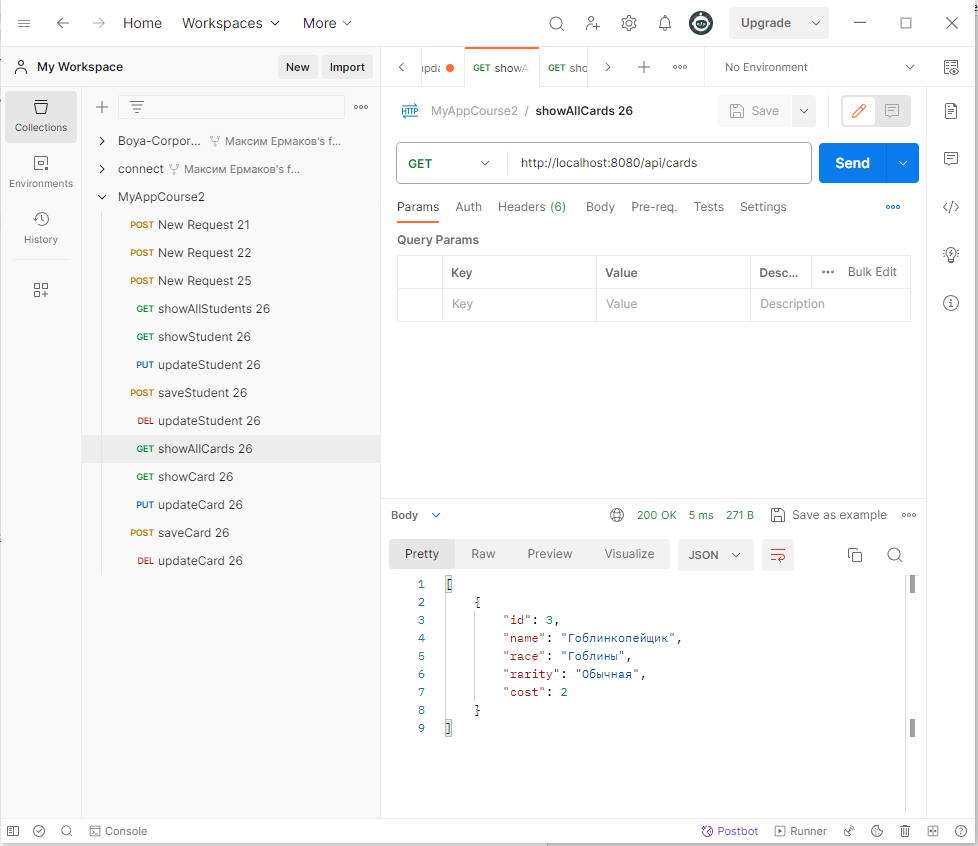
Рисунок 6 – Тестирование через Postman #3 – удаление ранее созданной карты в БД 

Рисунок 7 – Тестирование через Postman #4 – итог результата в БД

**Ответы на вопросы:**

2 Что из себя представляют подходы API-First и Code-First?

Что такое подход «API-First»?

В течение многих лет термину «API-first» не было стандартизированного определения в отрасли. Это слово означало разные вещи как для разработчиков API, так и для профессионалов. В соответствии с Отчет Postman о состоянии API за 2021 год, 42% разработчиков считали, что подход «API прежде всего» подразумевает планирование и проектирование API и базовой схемы перед созданием других зависимых компонентов API и приложений. Между тем, 31% считают, что этот термин относится к созданию API до приложений или интеграции. Хотя эти две точки зрения кажутся похожими, есть небольшая разница. **Первый рассматривает проектирование API как большую часть общего жизненного цикла разработки системы, а второй рассматривает API как основу для создания других систем.**

**По состоянию на 2023, этот подход определяется как проектирование и разработка API до написания фактического кода**. Начиная с проектирования API, разработчики могут установить четкие рекомендации и спецификации того, как различные части системы будут взаимодействовать друг с другом. Этот подход способствует созданию модульной и масштабируемой архитектуры, поскольку API выступает в качестве центральной точки контроля и координации.

4. Какие существуют виды интеграции информационных систем? Кратко опишите каждый.

Кто поддерживал и внедрял системы, а уж тем более, занимался доработкой, реинженерингом и интеграцией, тот знает, что более двух третей всех усилий в ИТ (внимания, времени и денег) уходит на «склейку» несовместимого и попытки «подружить» модули, написанные разными людьми, в разное время, на разных языках и технологиях, под разные платформы.  
  
Давайте перечислим и проанализируем **факторы**, влияющие на интеграцию:

* **Ускорение процессов**. Развитие организации требует все чаще и чаще менять структуры данных, бизнес-процессы, не говоря уже о дизайне и пользовательском интерфейсе, который просто постоянно находится в изменении. Вот, как раз в таких динамичных областях, где “изменчивость” является самой сутью и природой системы, задача интеграции усугубляется и превращается в серьезную проблему.
* **Распределенность**. Организации становятся все более крупными, а решаемые задачи все более комплексными, появляется логическая, организационная и географическая рассредоточенность.
* **Гетерогенность**. В крупном проекте, почти никогда нет возможности придерживаться платформ и инструментов от одного производителя, поэтому приходится учитывать и поддерживать особенности нескольких платформ.
* **Наследственность**. Невозможность полностью отказаться от легаси систем, морально устаревших технологий, старого аппаратного обеспечения, корторые, кстати, иногда дают вполне хорошие показатели по надежности и производительности но уж ни как не способствуют интеграции.
* **Хаотичность**. Не всегда есть возможность полностью формализовать, специфицировать и структурировать данные, и часть модели остается “слабо-связанной”, не поддающейся или слабо поддающейся машинной обработке, анализу, индексации, обсчету.
* **Обусловленность**. К сожалению, информационные системы ограничены не только техническими рамками, но и привычками людей (которых сложно переучивать), особенностями законодательства (которое просто не готово к появлению таких систем), множеством других факторов, не зависящих от разработчиков.
* **Интерактивность**. Потребитель информации постоянно повышает свои ожидания о скорости реакции системы, быстродействии и оперативности доставки информации. Большинство процессов стремятся к выполнению в реальном времени.
* **Мобильность**. Пользователь систем стал передвигаться быстрее, а взаимодействие с ним ведется через каналы связи общего пользования в транспорте, дома и на улице, в общественных местах и повсеместно.
* **Безопасность**. Пока данные хранились на носителе внутри охраняемого помещения, то особо ни кто не беспокоился о шифровании, но теперь сетевые пакеты летают в воздухе и это нельзя оставлять без внимания.
* **Высоконагруженность**. На сложность интеграции влияют: количество пользователей в системе, интенсивность потока обработки данных, объемы данных и ресурсоемкость вычислений.
* **Непрерывность цикла работы**. Интеграция и апгрейд систем почти всегда должны проводиться без остановки их функционирования, плавно, постепенно и незаметно для организации и ее клиентов.
* **Межсистемная интеграция**. Задачи стыковки не ограничены рамками организации, все чаще нужно интегрироваться с партнерами, клиентами, поставщиками, подрядчиками и даже государственными структурами.

6) Что такое gRPC, и какая разница между ним и REST?

В чем разница между gRPC и REST?

gRPC и REST – два способа разработки API. API – это механизм, который позволяет двум программным компонентам взаимодействовать друг с другом, используя набор определений и протоколов. **В gRPC один компонент (клиент) вызывает определенные функции в другом программном компоненте (сервере). В REST вместо вызова функций клиент запрашивает или обновляет данные на сервере.**

**Что такое gRPC?**

gRPC – это архитектура и система API с открытым исходным кодом, управляемая Cloud Native Computing Foundation. Она основана на модели удаленного вызова процедур (RPC). Хотя модель RPC обширна, gRPC обладает особой реализацией.

**Что такое RPC?**

В RPC взаимодействие между клиентом и сервером происходит так, как если бы клиентский API-запрос был локальной операцией или запрос был внутренним кодом сервера.

В RPC клиент отправляет запрос процессу на сервере, который постоянно прослушивает удаленные вызовы. В запросе содержится вызываемая серверная функция и все передаваемые параметры. RPC API использует в качестве базового механизма обмена данными протокол типа HTTP, TCP или UDP.

**Чем gRPC отличается от RPC?**

gRPC – это система, реализующая традиционный RPC с несколькими оптимизациями. Например, gRPC использует Protocol Buffers и HTTP 2 для передачи данных.

Она также абстрагирует механизм обмена данными от разработчика. Например, другая широко распространенная реализация RPC API, OpenAPI, требует от разработчиков сопоставления концепций RPC с протоколом HTTP. Но gRPC абстрагирует базовую HTTP-связь. Эти оптимизации делают gRPC быстрее, проще в реализации и удобнее для работы в Интернете по сравнению с другими реализациями RPC.

**Что такое REST?**

REST – это подход к архитектуре программного обеспечения, определяющий набор правил для обмена данными между программными компонентами. Он основан на HTTP, стандартном протоколе связи в Интернете. Интерфейсы RESTful API управляют связью между клиентом и сервером с помощью HTTP-команд, таких как *POST*, *GET*, *PUT* и *DELETE*, для операций создания, чтения, обновления и удаления. Ресурс на стороне сервера идентифицируется по адресу в виде URL.

REST работает следующим образом:

1. Клиент делает запрос на создание, изменение или удаление ресурса на сервере.
2. Запрос содержит адрес ресурса, а также может включать дополнительные параметры.
3. Сервер отвечает, возвращая клиенту весь ресурс после завершения операции.
4. Ответ содержит данные в формате JSON и коды состояния.

Интерфейсы API, созданные с использованием рекомендаций REST, называются *RESTful API*или *REST API*.

**Ссылка** на репозиторий **github.com** =

<https://github.com/themakson/MyJavaAppCourse2/tree/main/MyJavaAppCourse26H2DB>

**Вывод:**

В ходе лабораторной работы 6 мы разработали REST full service с помощью фреймворка Spring Boot в языке программирования Java, подключили базы данных h2, создали таблицы и наполнили их данными, а также протестировали данный сервис с помощью API запросов (Postman).

|  |  |
| --- | --- |
| Сдал: | М.С. Ермаков |
| Проверил: | Н.А. Архипов |