|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА– Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

**Отчет по практическим работам №5-7**

по дисциплине «Разработка серверных частей интернет-ресурсов»

**Студент группы** ИКБО-03-21 Хречко Сергей Викторович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**Руководитель практической работы** ассистент Благирев М.М.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Работа представлена «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Допущен к работе «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Москва 2023

Оглавление

[Практическая работа № 1 3](#_Toc148532469)

[Цель работы 3](#_Toc148532470)

[Ход работы 4](#_Toc148532471)

[Вывод 6](#_Toc148532472)

[Ответы на вопросы к практической работе 6](#_Toc148532473)

[Использованные источники 6](#_Toc148532474)

[Ссылка на удаленный репозиторий 7](#_Toc148532475)

[Практическая работа № 2 8](#_Toc148532476)

[Цель работы 8](#_Toc148532477)

[Ход работы 9](#_Toc148532478)

[Вывод 13](#_Toc148532479)

[Ответы на вопросы к практической работе 13](#_Toc148532480)

[Использованные источники 13](#_Toc148532481)

[Ссылка на удаленный репозиторий 14](#_Toc148532482)

[Практическая работа № 3 15](#_Toc148532483)

[Цель работы 15](#_Toc148532484)

[Ход работы 16](#_Toc148532485)

[Вывод 23](#_Toc148532486)

[Ответы на вопросы к практической работе 23](#_Toc148532487)

[Использованные источники 23](#_Toc148532488)

[Ссылка на удаленный репозиторий 24](#_Toc148532489)

[Практическая работа № 4 25](#_Toc148532490)

[Цель работы 25](#_Toc148532491)

[Ход работы 27](#_Toc148532492)

[Вывод 31](#_Toc148532493)

[Ответы на вопросы к практической работе 31](#_Toc148532494)

[Использованные источники 32](#_Toc148532495)

[Ссылка на удаленный репозиторий 32](#_Toc148532496)

# Практическая работа № 5

# Цель работы

Студенту предлагается создать свое серверное CRUDприложение. Приложение должно стать основанием для создание маркетплейса. В работе должны быть реализованы такие модели:

1. Book – модель, которая описывает сущность книги, с полями: автор, номер продавца, тип продукта (подразумевается электроника, книги, сантехника и т. п.), стоимость, название.

2. Client – модель, которая описывает сущность клиента сайта. У модели должны быть поля: имя, электронная почта, логин, пароль.

3. Telephone – модель, описывающая сущность телефона. Обязательные поля: производитель, объем аккумулятора, номер продавца, тип продукта (подразумевается электроника, книги, сантехника и т. п.), стоимость, название.

4. WashingMachine – модель, которая описывает сущность стиральной машины, с такими полями: производитель, объем бака, номер продавца, тип продукта (подразумевается электроника, книги, сантехника и т. п.), стоимость, название.

Так же должны быть контроллеры, которые позволят манипулировать модели (CRUD). Вся настройка приложения должна быть реализована в директории configuration. Приложение должно запускаться с помощью docker-compose.

# Ход работы

В первую очередь установим spring. На рисунке 1 представлен процесс создания проекта.

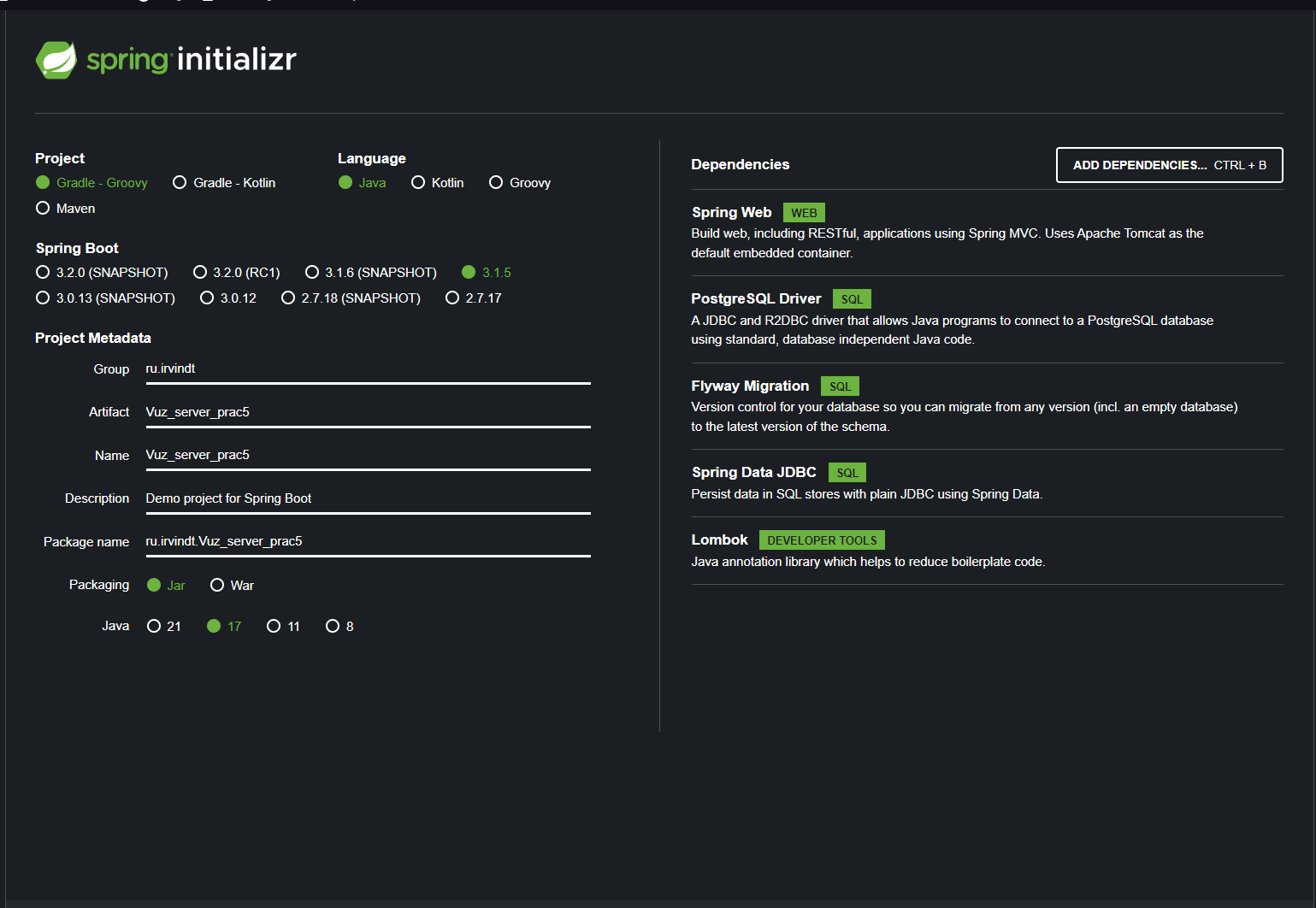


Рисунок 1 – Создания проекта spring

На рисунках 2-7 представлены результаты обработки различных CRUD запросов, а также последствия этим манипуляций.

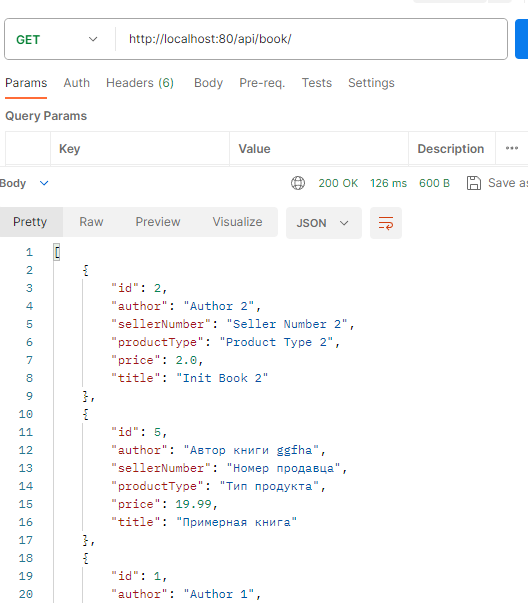


Рисунок 2 – Получение всех объектов

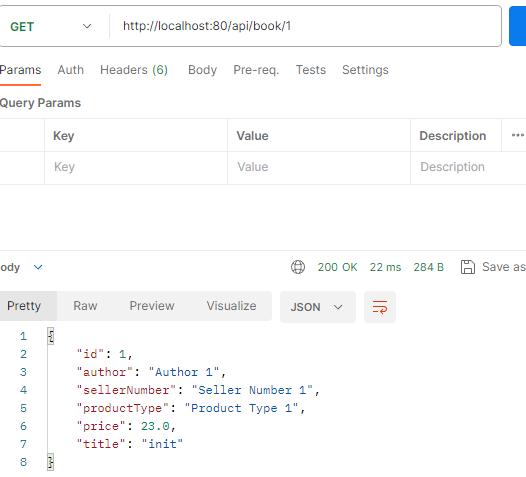


Рисунок 3 – Получение объекта по id

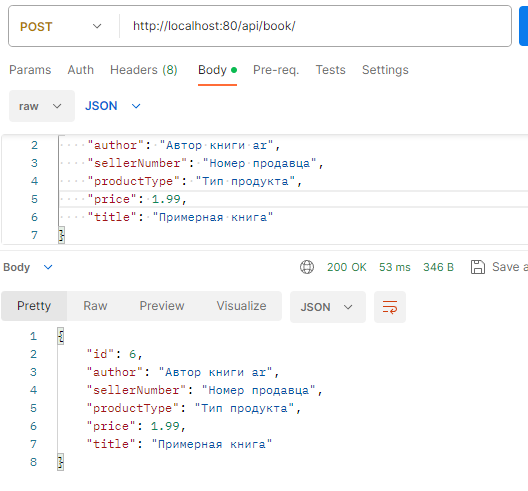


Рисунок 4 – Создание объекта

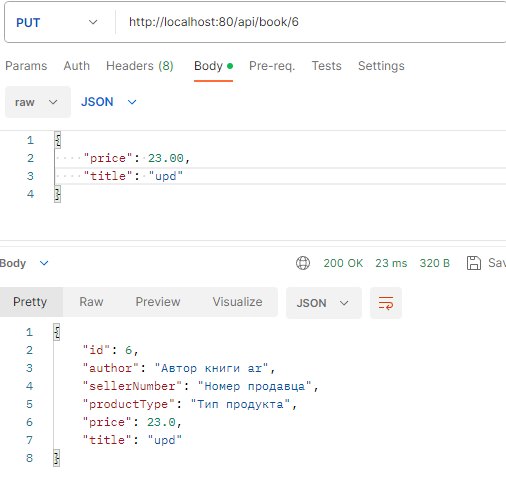


Рисунок 5 – Обновление объекта

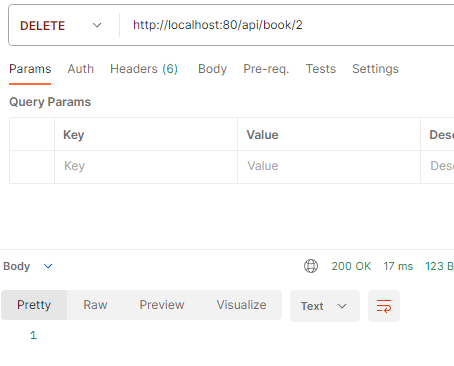


Рисунок 6 – Удаление объекта

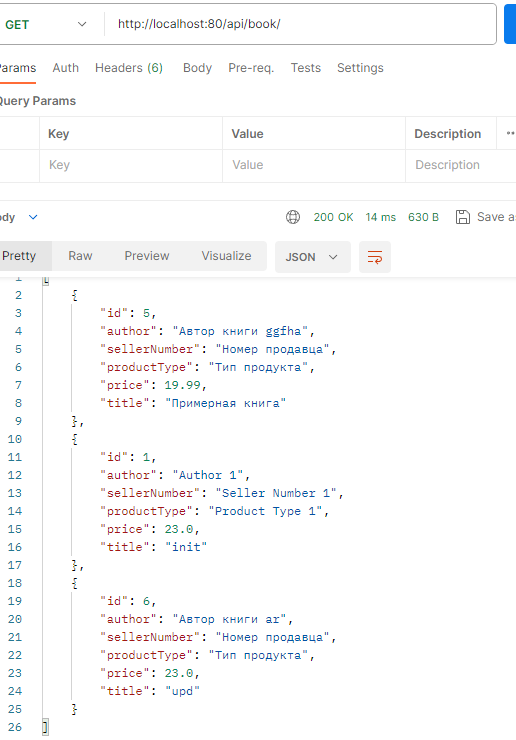


Рисунок 7 – Результат удаления объекта

# Вывод

В процессе выполнения работы были получены базовые навыки работы с фреймворком spring. Освоено создание сущностей. Освоена работа с CRUD операциями.

# Ответы на вопросы к практической работе

Что такое JDBCи JPA, в чем отличия.

JDBC (Java Database Connectivity) и JPA (Java Persistence API) - это два различных подхода к взаимодействию Java-приложений с базами данных. JDBC предоставляет низкоуровневый интерфейс для выполнения SQL-запросов и управления соединениями с базой данных, в то время как JPA предоставляет более высокоуровневый инструментарий для объектно-реляционного отображения данных в базе, позволяя работать с объектами Java напрямую, минимизируя использование SQL. Отличие заключается в уровне абстракции: JDBC предоставляет более прямой доступ к базе данных, тогда как JPA обеспечивает более удобное объектно-ориентированное взаимодействие.

Основные интерфейсы для реализации SpringData.

CrudRepository, который предоставляет базовые методы для создания, чтения, обновления и удаления данных,

PagingAndSortingRepository, добавляющий возможности пагинации и сортировки,

а также JpaRepository, расширяющий CrudRepository и предоставляющий дополнительные методы для работы с JPA-сущностями.

Аннотации @Entity, @Table. В чем разница.

@Entity указывает на то, что класс является сущностью, а @Table позволяет настраивать параметры таблицы для хранения этой сущности.

Что такое Lombok. Приведите несколько примеров использования.

Lombok - это библиотека для языка Java, которая автоматизирует создание стандартных элементов кода.

Примеры: создать сеттеры и геттеры, конструкторы, стандартные методы, вроде toString.

Назовите способы инициализации бинов в Java.

XML-конфигурация;

@Configuration, @Bean;

@Autowired;

@PostConstruct.

Что такое миграции и как они работаю в библиотеке Flyway.

Миграции (database migration) представляют собой процесс изменения схемы базы данных или данных с течением времени.

Этот процесс включает в себя последовательное применение скриптов миграции для обновления базы данных от одной версии к другой.

Что такое сущность и модуль. В чемразница.

сущность - это объект в системе, представленный классом в коде, который может быть отображен в базе данных. Модуль, с другой стороны, является организационной структурой для группировки компонентов приложения с целью управления их разработкой и поддержкой.

# Использованные источники

1. Статья про rest приложения - https://habr.com/ru/articles/471140/
2. Статья для аннотирования контроллеров - https://javarush.com/quests/lectures/questspring.level05.lecture02#
3. Статья про аннотация @ResponseBody - https://www.baeldung.com/spring-request-response-body
4. Статья про аннотация @SpringBootApplication - https://docs.spring.io/spring-boot/docs/2.0.x/reference/html/using-boot-using-springbootapplication-annotation.html
5. Статья про аннотация @PathVariable - https://www.baeldung.com/spring-pathvariable
6. CRUDприложение на Java – https://ru.hexlet.io/courses/java-web/lessons/crud-in-db/theory\_unit
7. ПервоеCRUD приложение наJava — https://javarush.com/groups/posts/2253-znakomstvo-s-maven-spring-mysql-hibernate-i-pervoe-crud-prilozhenie-chastjh-1
8. PostgreSQL – https://ru.wikipedia.org/wiki/PostgreSQL
9. Знакомство с Flyway – https://habr.com/ru/companies/otus/articles/506788/
10. ВведениевSpring Data JDBC – https://habr.com/ru/companies/otus/articles/531332/
11. ВведениевSpring Data JPA – https://habr.com/ru/articles/435114/
12. Gson – https://habr.com/ru/companies/naumen/articles/228279/
13. Lombok – https://habr.com/ru/companies/piter/articles/676394/

# Ссылка на удаленный репозиторий

<https://github.com/sergkhr/VUZserverProg>

# Практическая работа № 6

# Цель работы

Студенту предлагается реализовать бизнес логику ко второй практике по Java. Нужно добавить к существующему приложению реализацию «корзины».

Пользователь должен иметь такие возможности:

1. Добавить товар в корзину.
2. Удалить товар из корзины.
3. Изменить количество товара в корзине.
4. Посмотреть всю корзину.
5. Оформить заказ и очистить корзину.

При добавлении товара в корзину должна быть проверка, что товар есть в наличии. Так же обработать случай, что товар добавили в корзину, после товар закончился на складе и оформить заказ невозможно.

Сервис должен с помощью docker-compose. Должно быть использовано паттерны проектирования: Чистая архитектура, MVC.

.

# Ход работы

На рисунке 8 представлено получение корзины.

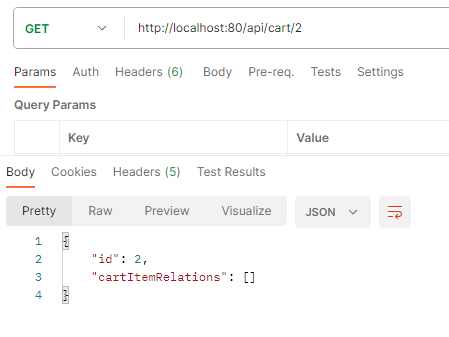


Рисунок 8 – Получение корзины

На рисунках 9-10 представлено добавление предмета в корзину.

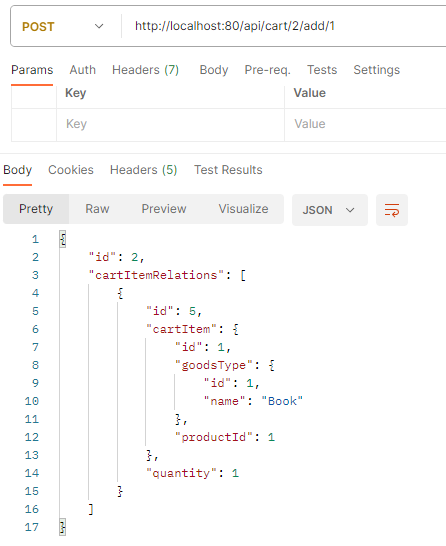


Рисунок 9 – Добавление в корзину



Рисунок 10 – Добавление объекта

На рисунке 11 представлено удаление предмета из корзины.

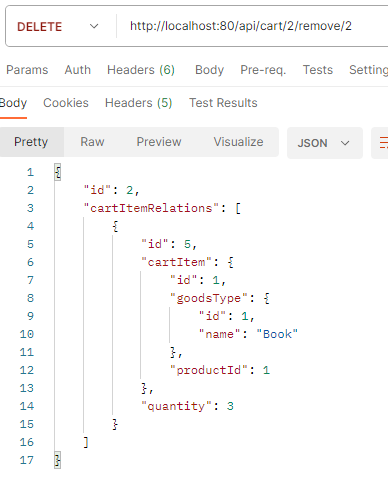


Рисунок 11 – Удаление предмета

На рисунке 12 представлено изменение количества предметов в корзине.

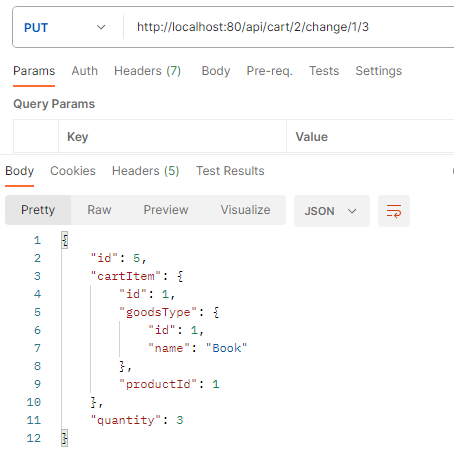


Рисунок 12 – Изменение количества предметов в корзине

На рисунке 13 представлена корзина после проведенных манипуляций.

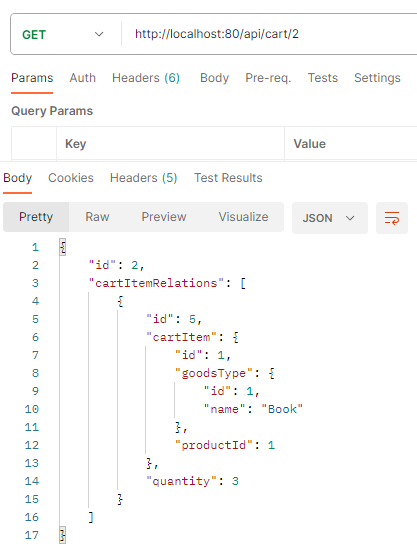


Рисунок 13 – Корзина после манипуляций

На рисунке 14 представлено количество товаров до очистки корзины, в процессе которой мы списываем часть товара, так как предполагается, что они были куплены.

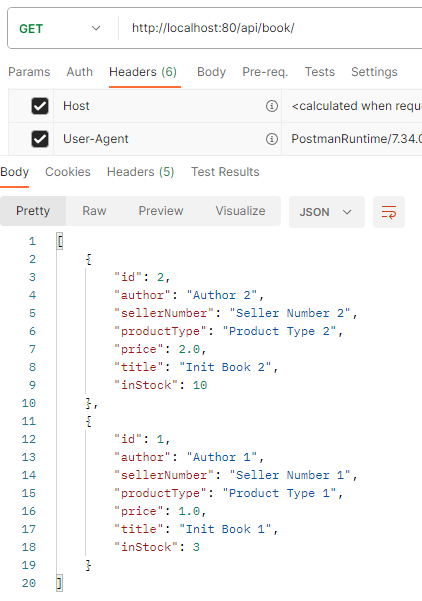


Рисунок 14 – Количество товара

Обратите внимание, что сейчас в наличие 3 книги. На рисунке 15 показано успешное очищение корзины. На рисунке 16 – пустая корзина. На рисунке 17 показано отсутствие товара, после очищения корзины.

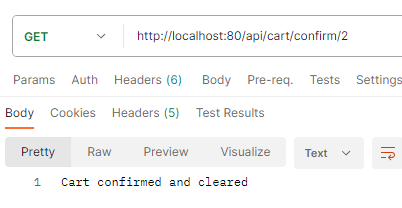


Рисунок 15 – Очищение корзины

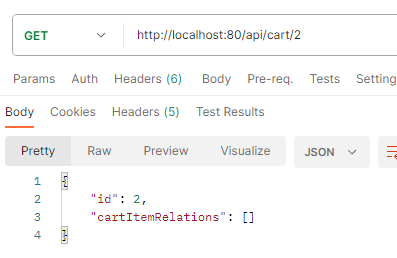


Рисунок 16 – Пустая корзина

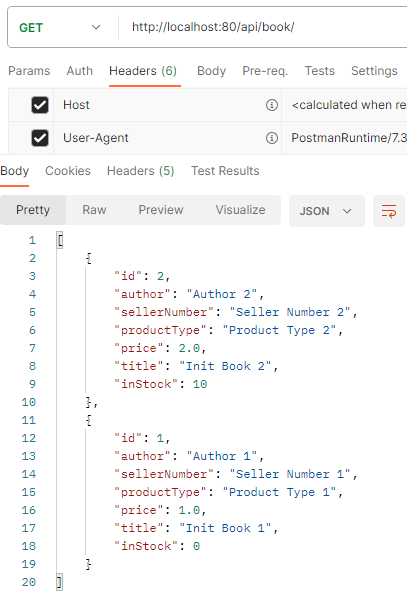


Рисунок 17 – Отсутствие книг

# Вывод

В процессе выполнения работы были получены навыки работы с фреймворком spring. Была освоена настройка связей между сущностями. Была реализована логика корзины.

# Ответы на вопросы к практической работе

Аннотация Service.

Аннотация @Service в Spring используется для пометки класса как сервиса, предоставляющего бизнес-логику. Эта аннотация позволяет Spring автоматически обнаруживать и регистрировать класс как компонент при сканировании пакетов, а также внедрять его в другие компоненты при необходимости.

Паттерн MVC.

это архитектурный шаблон проектирования, разделяющий компоненты приложения на три основных части

Модель (Model):

- Представляет собой бизнес-логику и данные приложения.

- Отвечает за обработку данных, их обновление и передачу уведомлений представлению об изменениях.

Представление (View):

- Отвечает за отображение данных пользователю и предоставляет интерфейс взаимодействия.

- Получает данные от модели и отображает их пользователю.

Контроллер (Controller):

- Обрабатывает пользовательский ввод и взаимодействует с моделью и представлением.

- Принимает запросы от пользователя, обновляет модель и обновляет представление в соответствии с изменениями в модели.

Паттерн «Чистая архитектура».

Паттерн "Чистая архитектура" представляет собой методологию построения программных систем с акцентом на разделение ответственности, легкость тестирования и независимость от конкретных технологий.

Описать все слои RestFull приложения и описать зачем каждый из них.

RESTful приложение обычно следует принципам архитектуры REST (Representational State Transfer), и его структура может быть организована в виде различных слоев. Основные слои RESTful приложения включают:

Слой ресурсов (Resource Layer):

Описание: Этот слой представляет собой ресурсы, с которыми клиенты взаимодействуют. Ресурсы могут быть представлены в виде URI (Uniform Resource Identifier) и могут включать, например, объекты, коллекции или услуги.

Цель: Обеспечивает точки входа для клиентов приложения, определяя, какие данные или услуги могут быть получены или изменены.

Слой представления (Representation Layer):

Описание: В этом слое данные представляются в виде конкретных форматов, таких как JSON или XML. Он определяет, как ресурсы будут представлены клиентам и как клиенты будут представлять данные при отправке запросов.

Цель: Предоставляет стандартизированный способ обмена данными между клиентами и сервером.

Слой управления (Controller Layer):

Описание: Контроллеры обрабатывают запросы от клиентов, вызывая соответствующие сервисы для выполнения бизнес-логики. Они принимают входные данные, валидируют их и направляют выполнение дальше по стеку.

Цель: Реализация бизнес-логики и оркестрация работы различных компонентов приложения.

Слой сервисов (Service Layer):

Описание: Сервисы предоставляют бизнес-логику, которую вызывают контроллеры. Они могут включать в себя операции, связанные с обработкой данных, взаимодействием с хранилищем данных и другими бизнес-операциями.

Цель: Разделение бизнес-логики от контроллеров для обеспечения более модульной и поддерживаемой структуры приложения.

Слой хранилища данных (Data Storage Layer):

Описание: Этот слой отвечает за взаимодействие с базой данных или другими механизмами хранения данных. Он обеспечивает сохранение и получение данных, необходимых для выполнения бизнес-логики.

Цель: Управление постоянными данными и обеспечение их доступности для остальных слоев приложения.

# Использованные источники

Чистая архитектура — https://www.baeldung.com/spring-boot-clean-architecture

MVC – https://javarush.com/groups/posts/2536-chastjh-7-znakomstvo-s-patternom-mvc-model-view-controller

Аннотация Service – https://www.baeldung.com/spring-service-annotation-placement

# Ссылка на удаленный репозиторий

<https://github.com/sergkhr/VUZserverProg>

# Практическая работа № 7

# Цель работы

Студенту предлагается дополнить задание шестой практики по Java. Нужно создать приложение на микросервисной архитектуре, где один сервис — это приложение шестой практики, а второй сервис — это сервис авторизации.

Сервис авторизации должен быть написан с помощью Spring Security исодержатьJWT token. В качестве СУБД во втором сервисе должен быть использован Redis. Должны быть созданы три роли: USER, SELLER, ADMINISTRATOR.

USERне имеет доступ к какому-либо методу, который связан с редактированием или просмотром информации по другому пользователю, не имеет возможности добавлять или удалять товары.

SELLERимеет те же ограничения, что и USER, за исключением, что продавец может добавлять и удалять свои товары.

ADMINISTRATORне имеет каких либо ограничений.

В сервис с приложением маркетплейса добавить бизнес логику, позволяющую пользователь с ролью USERполучить роль SELLER.

Практика должна запускаться с помощью docker-compose. Каждый микросервис должен запускаться в отдельном потоке.

.

# Ход работы

На рисунке 18 показано создание нового пользователя.

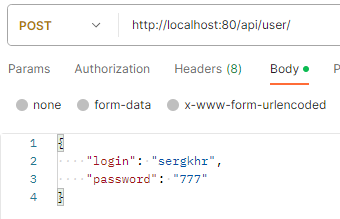


Рисунок 18 – Создания нового пользователя

На рисунке 19 показано получение всех пользователей.

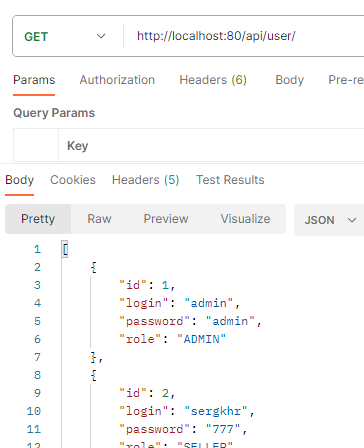


Рисунок 19 – Получение пользователей

На рисунке 20 показана авторизация. Мы получаем jwt токен, который в дальнейшем хранится в базе данных redis.

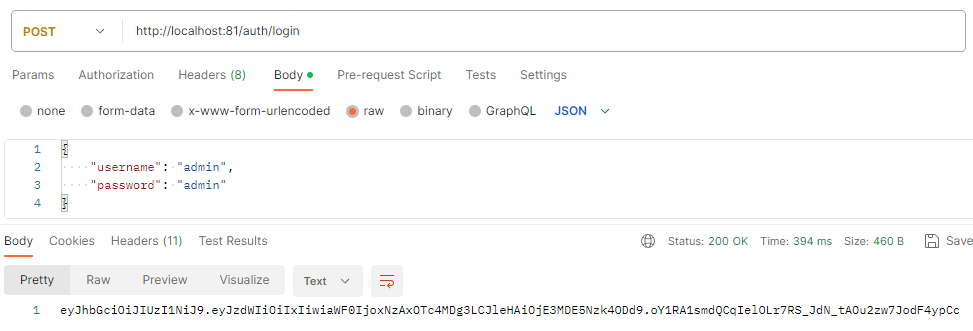


Рисунок 20 – Авторизация

На рисунке 21 показано то, как мы передаем jwt токен в запрос для работы с корзиной и товарами.



Рисунок 21 – Передача токена

# Вывод

В процессе выполнения работы были получены базовые навыки работы с фреймворком spring. Освноена работа с безопасностью и настройкой ролей для работы с корзиной и товарами.

# Ответы на вопросы к практической работе

Spring Security.

Spring Security - это мощный фреймворк для обеспечения аутентификации, авторизации и защиты приложений на платформе Spring. Он предоставляет множество готовых решений для обеспечения безопасности веб-приложений и служит стандартным инструментом в сообществе Java для обеспечения защиты приложений.

Что такое JWTTokenи как он работает.

JWT (JSON Web Token) представляет собой стандарт RFC 7519 для создания токенов с использованием формата JSON. JWT состоит из трех частей, закодированных в формате Base64 и разделенных точками: заголовка (header), полезной нагрузки (payload) и подпись (signature).

Заголовок (Header):

Содержит информацию о типе токена и используемом алгоритме подписи.

Полезная нагрузка (Payload):

Содержит утверждения (claims) о пользователе или другую информацию. Включает в себя стандартные утверждения (например, "sub" - субъект, "exp" - срок действия), а также может содержать пользовательские утверждения.

Подпись (Signature):

Создается с использованием заголовка, полезной нагрузки и секретного ключа (если используется алгоритм с секретным ключом). Подпись позволяет проверить подлинность токена.

Микросервисная архитектура. Плюсы и минусы перед монолитной системой.

Плюсы:

Масштабируемость, гибкость и легкость изменений, разделение ответственностей, устойчивость к отказам, легкость в развертывании.

Минусы:

Сложность управления, сложность тестирования, сетевая сложность, трудность обеспечения целостности данных, дополнительные расходы на безопасность, сложность отладки.

Варианты масштабируемости.

Вертикальная масштабируемость (Scale Up):

Горизонтальная масштабируемость (Scale Out):

Подход "Микросервисы":

Описание: Разделение системы на небольшие, независимые микросервисы, каждый из которых может быть масштабирован независимо.

Что такое вертикальная масштабируемость.

Увеличение производительности путем улучшения характеристик отдельных серверов (больше процессоров, памяти, дисков).

Что такое горизонтальная масштабируемость.

Добавление дополнительных экземпляров приложений или серверов для распределения нагрузки.

Объяснить принцип сбора микросервисной архитектуры в Docker.

Принцип сбора микросервисной архитектуры в Docker основан на использовании контейнеризации для упаковки, доставки и выполнения микросервисов в изолированной среде. Docker обеспечивает легкость развертывания и управления микросервисами, устраняя зависимости от конкретных окружений и обеспечивая консистентность между различными средами.

NoSQLDB. Почему в сервисе авторизации удобнее использовать Redis.

Можно хранить jwt токены и удалять и по времени. А еще Redis быстрый.

Многопотчность — что это такое. Как оно работает в Java.

Многопоточность - это концепция, при которой несколько потоков выполнения существуют в пределах одного процесса и делят общие ресурсы, такие как память и файловые дескрипторы. Многопоточность позволяет программам эффективно использовать ресурсы процессора и улучшать отзывчивость приложений.

Классами Thread и Concurrent

Класс Thread, расскажите про его методы.

Метод start() запускает новый поток выполнения.

Метод run() содержит код, который будет выполнен внутри потока.

Метод sleep() заставляет поток "заснуть" на указанное количество миллисекунд.

Метод join() заставляет текущий поток ожидать завершения выполнения другого потока, на который он вызывается.

Метод interrupt() посылает сигнал прерывания потоку. Обработка этого сигнала должна быть реализована внутри кода потока.

Метод isAlive() возвращает true, если поток был запущен и ещё не завершил выполнение.

И другие, используемые реже.

# Использованные источники

1. RestFull приложениесоSpring Security иJWT Token – https://habr.com/ru/articles/545610/
2. JWT Token иSpring Security – https://www.baeldung.com/spring-security-oauth-jwt
3. JWT Token – https://proglib.io/p/json-tokens
4. Spring Security – https://spring.io/projects/spring-security
5. Микросервисная архитектура — https://habr.com/ru/companies/vk/articles/320962/
6. UserDetails – https://www.baeldung.com/spring-security-authentication-with-a-database

# Ссылка на удаленный репозиторий

<https://github.com/sergkhr/VUZserverProg>