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

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

**ПРАКТИЧЕСКИЕ РАБОТЫ №5-7**

по дисциплине «Разработка серверных частей интернет-ресурсов»

**Студент группы** ИКБО-20-21 Сидоров С.Д.

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

(подпись студента)

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

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

(подпись руководителя)

Работа представлена « » 2023 г.

Допущен к работе « » 2023 г.

Москва 2023

**Содержание**

[Практическая работа №5 3](#_Toc153365070)

[Теоретическое введение. 3](#_Toc153365071)

[Задание 6](#_Toc153365072)

[Выполнение задания 7](#_Toc153365073)

[Выводы 15](#_Toc153365074)

[Ответы на вопросы 16](#_Toc153365075)

[Практическая работа №6 18](#_Toc153365076)

[Теоретическое 18](#_Toc153365077)

[Задание 20](#_Toc153365078)

[Выполнение задания 21](#_Toc153365079)

[Выводы 29](#_Toc153365080)

[Ответы на вопросы 30](#_Toc153365081)

[Практическая работа №7 31](#_Toc153365082)

[Теоретическое введение. 31](#_Toc153365083)

[Задание 33](#_Toc153365084)

[Выполнение задания 34](#_Toc153365085)

[Выводы 39](#_Toc153365086)

[Ответы на вопросы 40](#_Toc153365087)

[Список использованных источников 42](#_Toc153365088)

# Практическая работа №5

## Теоретическое введение.

Java — это строго типизированный объектно-ориентированный язык программирования. Приложения на Java транслируются в байт-код, поэтому их можно запускать на любом устройстве, где установлена виртуальная машина Java (JVM).

Язык Java отличается завидной универсальностью. На нём можно писать что угодно: серверные и десктопные программы, мобильные и веб-приложения, игры любой сложности.

Чаще всего на Java пишут Backend (бэкенд). А вот теперь самое время узнать, что это такое и за что отвечает бэкенд-разработчик.

Благодаря своей надёжности и прекрасной обратной совместимости Java прекрасно подходит для корпоративных разработок. Например, на нём написана Jira — пожалуй, самая популярная платформа для управления проектами. По тем же причинам Java очень любят в финтехе, а такие банки, как «Сбер», «Альфа» и «Райффайзен» регулярно хантят джавистов.

У Java есть прекрасные технологии и в десктопе — например, OpenJFX для создания локальных приложений с графическим интерфейсом. Среды разработки ПО (о них — ниже) Eclipse IDE и IntelliJ IDEA тоже написаны на Java.

Бэкендер должен знать преимущества и недостатки технологий, чтобы выбрать ту, которая лучше всего подходит для текущего проекта. Он должен хорошо разбираться в выбранной технологии или иметь возможность быстро её изучить. Рассмотрим основные инструменты, с которыми работают бэкендеры на Java.

JDK, или Java Development Kit, — это набор инструментов для разработки ПО на языке Java. В него входят компилятор Javac, виртуальная машина Java, стандартная библиотека модулей, классов и функций и документация.

Стандартный дистрибутив JDK поддерживает компания Oracle, но помимо него существует и ряд других. Например, Red Hat, SAP, Google разрабатывают свои инструменты и вносят вклад в развитие свободной версии JDK — OpenJDK. В России, с учётом курса на замещение продуктов зарубежных вендоров, имеет смысл использовать отечественный рантайм Axiom JDK — это единственный JDK, внесённый в реестр российского ПО.

Фреймворк — это шаблон, на основе которого создаются приложения. Самый популярный фреймворк для Java — Spring, но недавно у него появились конкуренты Micronaut и Quarkus, заточенные под создание облачных приложений и микросервисов.

В повседневной работе любой разработчик активно пользуется библиотеками, которые содержат готовые решения для многих распространённых задач программирования. Помимо стандартной библиотеки Java, существует много дополнительных полезных проектов — например, Maven, JHipster, Apache Commons и так далее. Вы так или иначе познакомитесь с этими инструментами в процессе обучения и увидите, насколько они облегчают жизнь разработчика.

Все практики на Java в этом курсе будут выполняться с помощью библиотеки Spring. Поговорим про нее подробнее. По сути Spring Framework представляет собой просто контейнер внедрения зависимостей, с несколькими удобными слоями (например: доступ к базе данных, прокси, аспектно-ориентированное программирование, RPC, веб-инфраструктура MVC). Это все позволяет вам быстрее и удобнее создавать Java-приложения.

Вся работа со Spring ведется с помощью подключенных зависимостей, к которым программист может обращаться с помощью специальных аннотаций. В рамках этой практической задачи мы познакомимся с такими зависимостями как:

1. @Controller - эта аннотация создаст бин, с помощью которого Spring создаст класс контроллер, который будет принимать запросы;
2. @ResponseBody - аннотация, которая дает фреймворку понять, что результат метода - это неконечное представление, которое нужно отправить клиенту, нужно результат предварительно прогнать через класс HttpMessageConverter;
3. @RestController - это составная аннотация, которая сама мета-аннотируется аннотациями @Controller и @ResponseBody;
4. @GetMapping - обрабатывает get запросы;
5. @PostMapping - обрабатывает post запросы;
6. @PathVariable - позволяет получать значения параметров метода из запроса на сервер;
7. @RequestParam - это аннотация, которая перехватывает параметр запроса;
8. @RequestMapping - с помощью этой аннотации можно разделять запросы по путям;
9. . @SpringBootApplication - аннотация, которая создает начальную конфигурацию приложения (@EnableAutoConfiguration, @ComponentScan, @Configuration).

## Задание

Студенту предлагается создать свое серверное CRUD приложение. Приложение должно стать основанием для создания 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.

## Выполнение задания

Для создания образа, был использован Dockerfile, изображенный на рисунке 1.

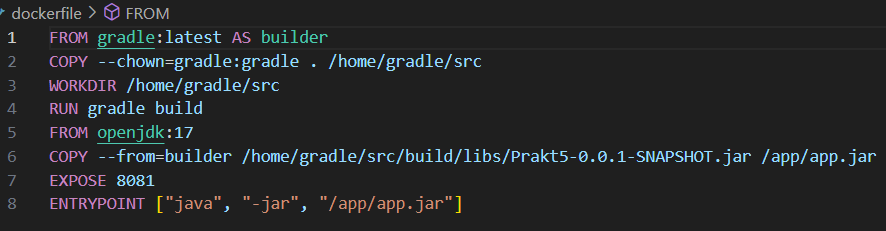


Рисунок 1 – Dockerfile

Для связи приложения и базы данных мы будем использовать docker-compose. Его содержимое показано на рисунке 2.

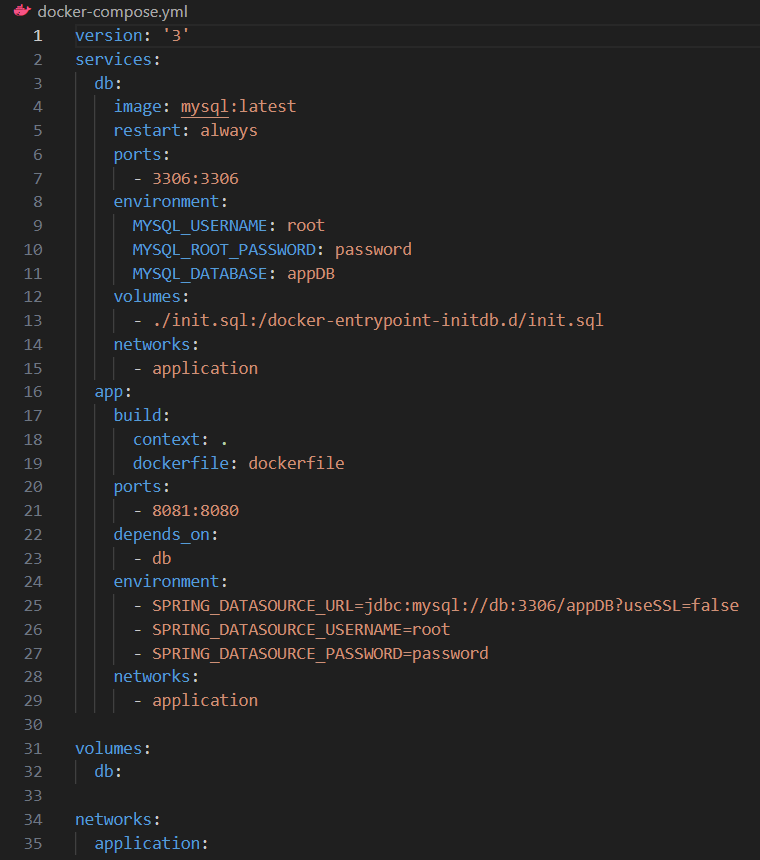


Рисунок 2 – docker-compose.yml

Создадим модели книги, клиента сайта, телефона и стиральной машины. Пример модели показан на рисунке 3.

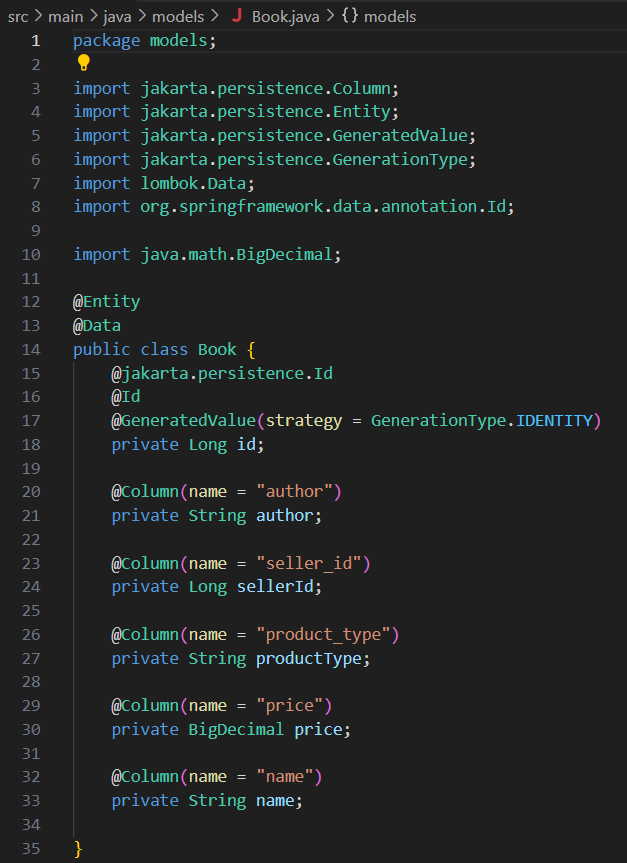


Рисунок 3 – Код модели книги

Для работы с репозиторием будем использовать JpaRepository. На рисунке 4 показан пример репозитория.

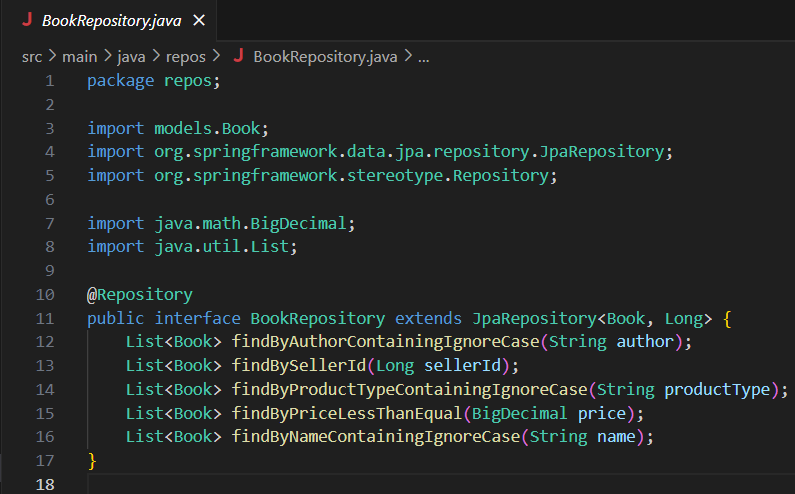


Рисунок 4 – Код репозитория книги

Для написания методов работы с моделями используем контроллеры, прописав CRUD операции – добавления, удаления, вывода, изменения модели. Пример контроллера показан на рисунке 5.

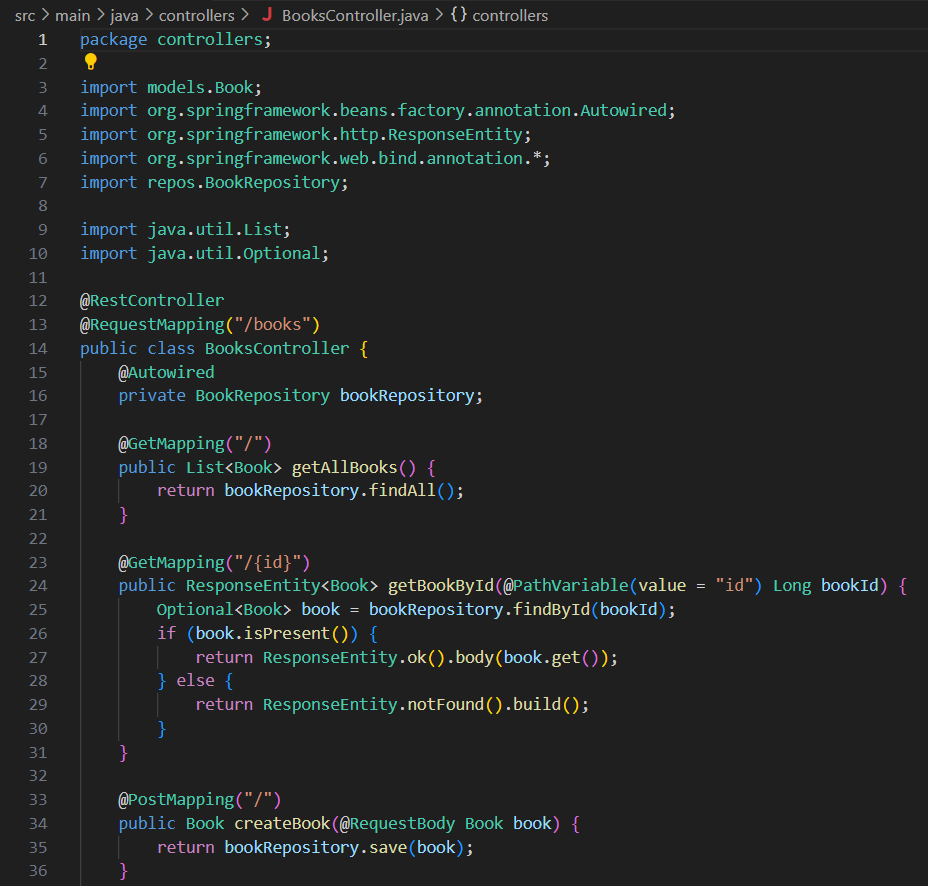


Рисунок 5 – Код контроллера книги

Результат работы программы представлен на рисунках 6 – 9.

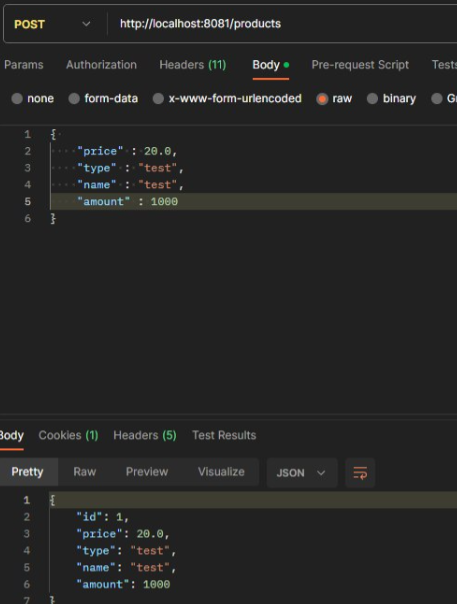


Рисунок 6 – Добавление продукта

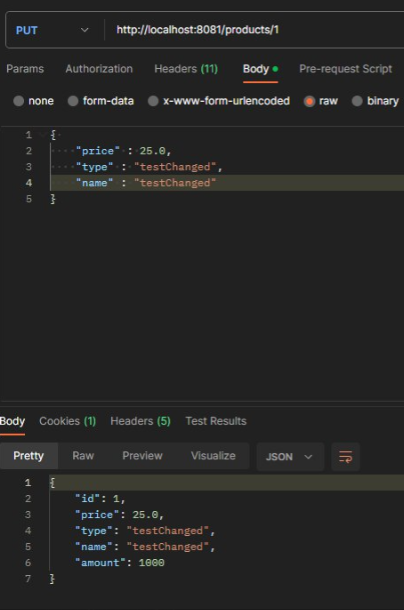


Рисунок 7 – Изменение продукта

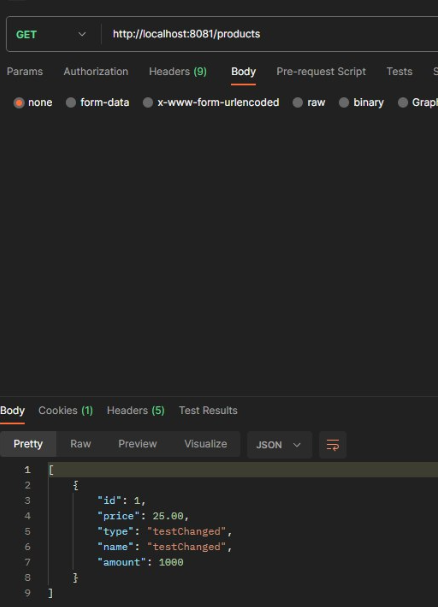


Рисунок 8 – Вывод продукта

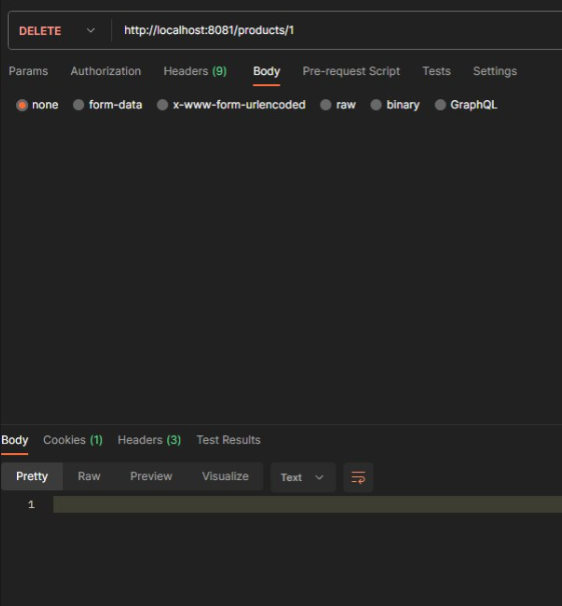


Рисунок 9 – Удаление продукта

## Выводы

Таким образом, было написано свое серверное CRUD приложение, запускающееся с помощью docker-compose. Приложение было протестировано с помощью Postman, все тесты были успешно пройдены.

## Ответы на вопросы

Что такое JDBC и JPA, в чем отличия? – JDBC (Java Database Connectivity) и JPA (Java Persistence API) - это два подхода к взаимодействию с базами данных в Java. JDBC предоставляет низкоуровневый доступ к базам данных с использованием SQL, в то время как JPA - это более высокоуровневый способ работы с базами данных, использующий объектно-ориентированный подход с сущностями и запросами на языке JPQL.

Основные интерфейсы для реализации Spring Data? – Основные интерфейсы для реализации Spring Data включают в себя Repository, CrudRepository, и PagingAndSortingRepository. Они предоставляют стандартные методы для взаимодействия с базой данных, такие как создание, чтение, обновление и удаление записей.

Аннотации @Entity, @Table. В чем разница? – Аннотация @Entity используется в JPA для обозначения класса как сущности базы данных, а @Table позволяет настраивать таблицу базы данных, связанную с этой сущностью, указывая имя и другие параметры.

Что такое Lombok. Приведите несколько примеров использования. –Lombok - это библиотека для упрощения разработки в Java. Она позволяет генерировать автоматически код для геттеров, сеттеров, конструкторов и других методов, уменьшая шаблонный код. Примеры использования Lombok включают аннотации @Getter, @Setter, @NoArgsConstructor и @AllArgsConstructor для генерации кода.

Назовите способы инициализации бинов в Java. – Существуют разные способы инициализации бинов в Java, включая XML-конфигурации, аннотации (например, @Component, @Service, @Repository), Java-конфигурации (с использованием классов конфигурации с аннотацией @Configuration) и через Java API, создавая бины программно.

Что такое миграции и как они работаю в библиотеке Flyway? – Миграции - это способ управления изменениями схемы базы данных. В библиотеке Flyway миграции представляют собой набор SQL-скриптов, которые могут быть применены к базе данных последовательно, чтобы обновить ее схему. Flyway автоматически применяет миграции при запуске приложения, обеспечивая согласованность базы данных.

Что такое сущность и модуль. В чем разница? – Сущность - это объект, представляющий данные в контексте базы данных, обычно отображаемый на таблицу в реляционной базе данных. Модуль - это более широкое понятие, которое может включать в себя несколько сущностей и другие компоненты, связанные с определенной функциональностью или задачей в приложении.

# Практическая работа №6

Теоретическое **введение.**

Когда мы разрабатываем долгосрочные системы, мы должны ожидать изменяемой среды.

В целом, наши функциональные требования, фреймворки, устройства ввода-вывода и даже дизайн нашего кода могут измениться по разным причинам. Имея это в виду, чистая архитектура является ориентиром для кода с высокой степенью поддержки, учитывая все окружающие нас неопределенности**.**

В этой статье мы создадим пример API для регистрации пользователей, следуя [чистой архитектуре Роберта К. Мартина](https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html). Мы будем использовать его оригинальные слои – сущности, варианты использования, интерфейсные адаптеры и фреймворки / драйверы.

Чистая архитектура компилирует множество дизайнов кода и принципов, таких как [SOLID](https://www.baeldung.com/solid-principles), [стабильные абстракции](https://wiki.c2.com/?StableAbstractionsPrinciple) и другие. Но основная идея заключается в том, чтобы разделить систему на уровни, основанные на ценности для бизнеса**.** Следовательно, самый высокий уровень имеет бизнес-правила, причем каждый нижний уровень приближается к устройствам ввода-вывода.

Кроме того, мы можем переводить уровни в слои. В данном случае все наоборот. Внутренний слой соответствует самому высокому уровню и так далее:

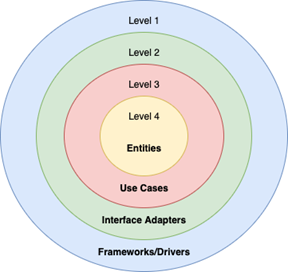


Рисунок 1 — слои в чистой архитектуре

Помня об этом,мы можем иметь столько уровней, сколько требуется нашему бизнесу. Но всегда учитывайте правило зависимости – более высокий уровень никогда не должен зависеть от более низкого.

## Задание

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.

## Выполнение задания

Dockerfile и docker-compose будем использовать с прошлой практики, рисунки 10, 11.

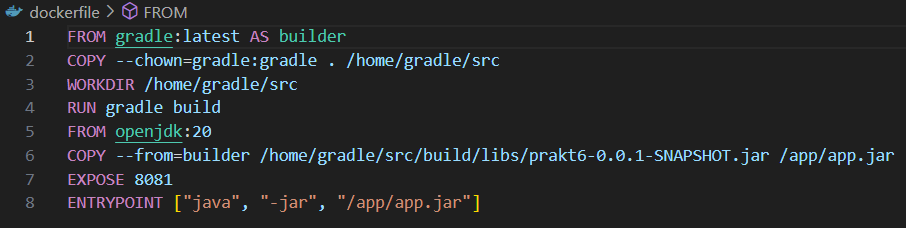


Рисунок 10 – Dockerfile проекта

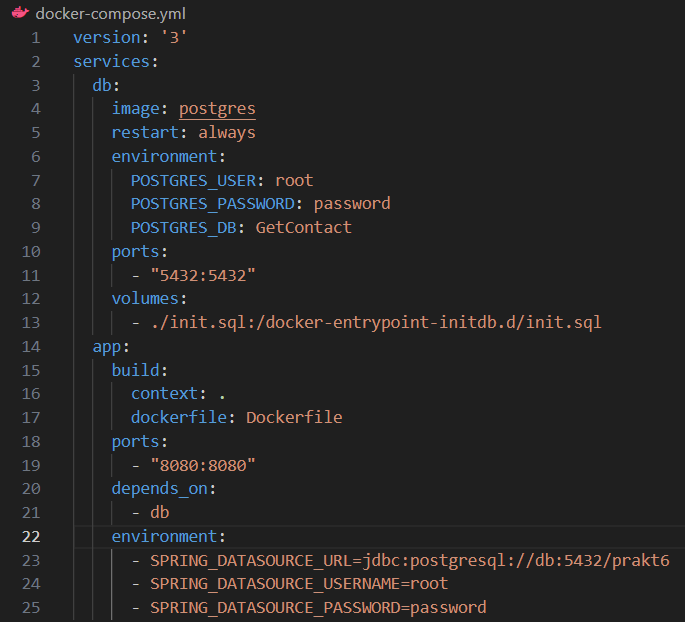


Рисунок 11 – docker-compose проекта

Для использования корзины создадим ее модель, она изображена на рисунке 12.

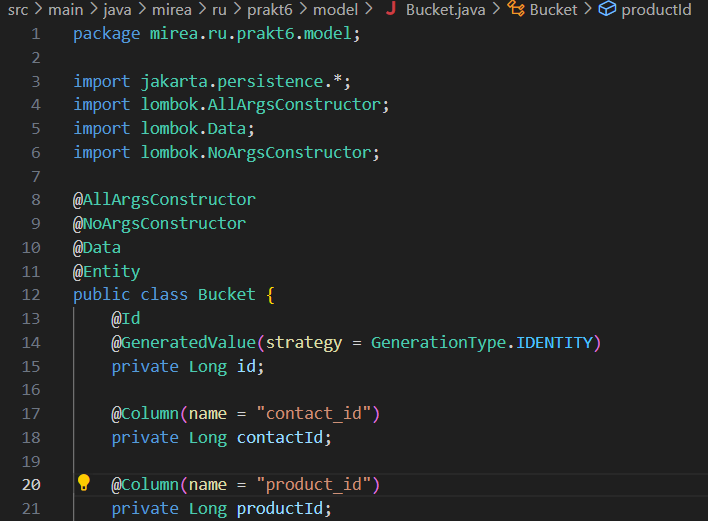


Рисунок 12 – Модель корзины

В качестве репозитория использовался JpaRepository, рисунок 13.

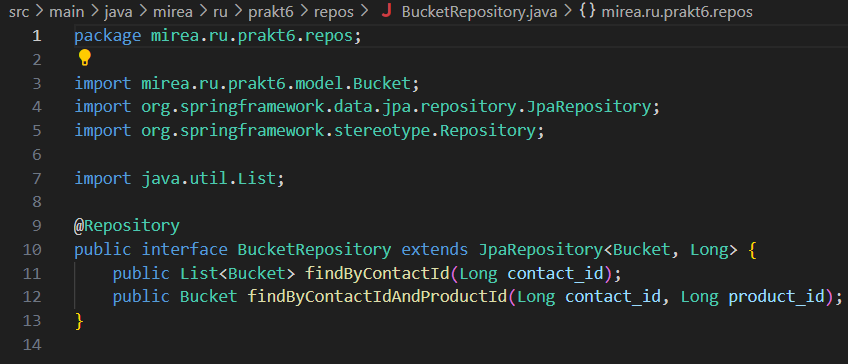


Рисунок 13 – Код репозитория деталей корзины

В сервисе написали методы просмотра всех корзин, просмотра одной корзины, изменение корзины, удаление ко­рзины по id клиента, создание корзины, просмотр корзины по id клиента, добавление товара в корзину, удаление товара из корзины,­ изменение количества товара в корзине, подсчет общей стоимости, проверка наличия всех товаров в корзине, покупка товаров, на рисунках 14-20.

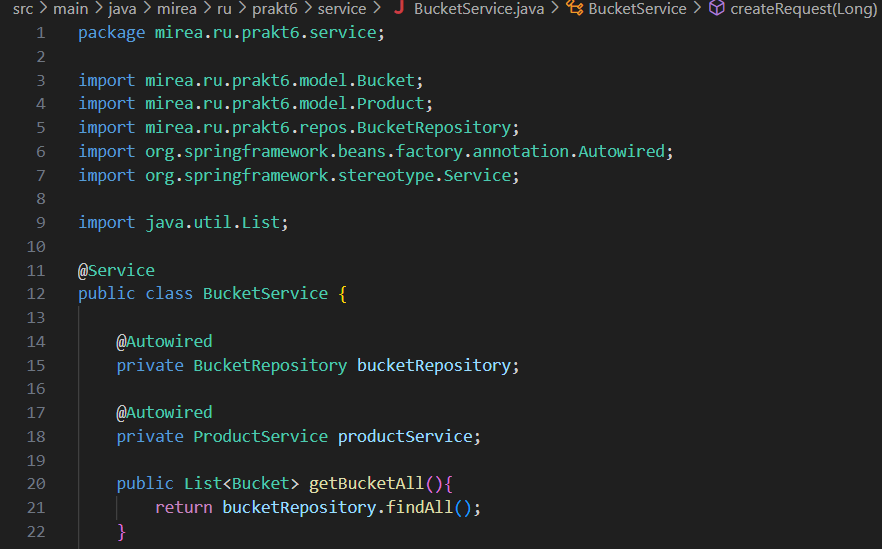


Рисунок 14 – Фрагмент кода сервиса корзины

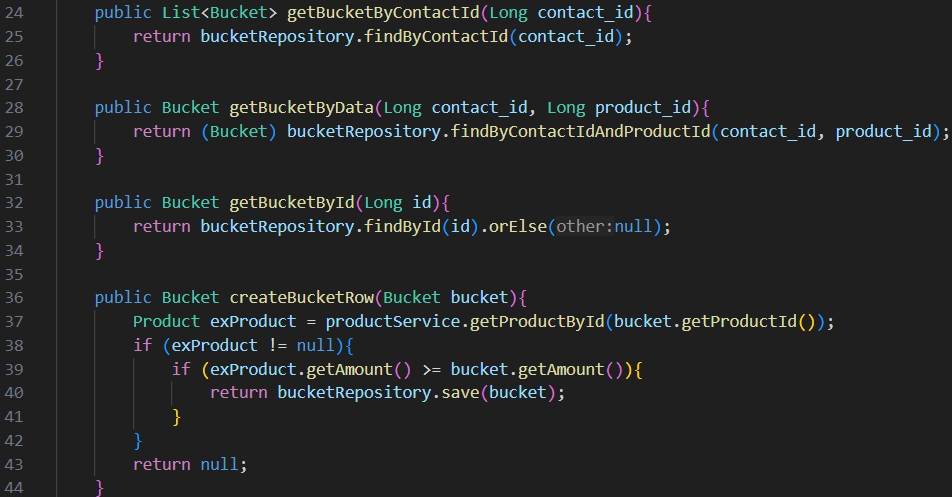


Рисунок 15 – Фрагмент кода сервиса корзины

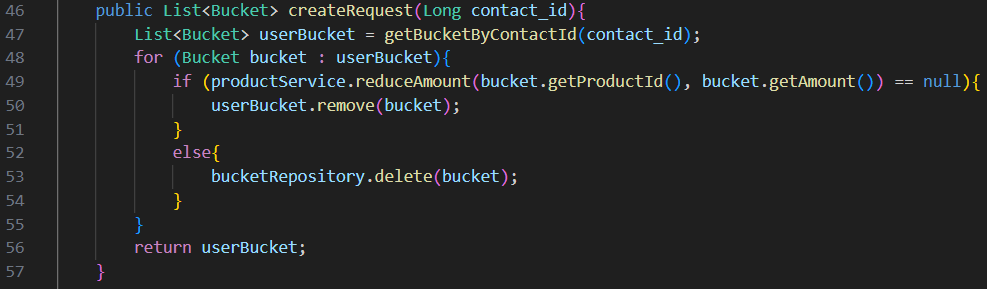


Рисунок 16 – Фрагмент кода сервиса корзины

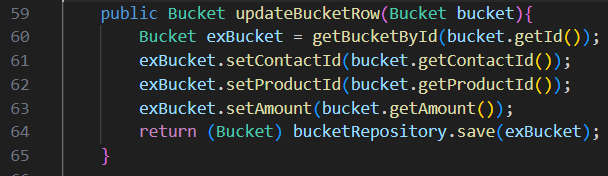


Рисунок 17 – Фрагмент кода сервиса корзины

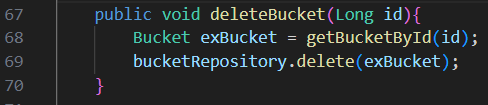


Рисунок 18 – Фрагмент кода сервиса корзины

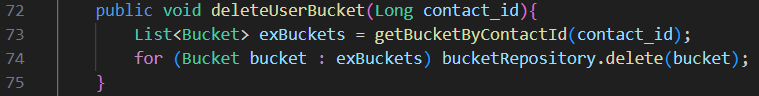


Рисунок 19 – Фрагмент кода сервиса корзины

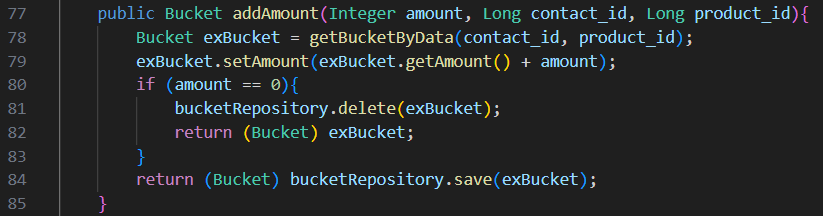


Рисунок 20 – Фрагмент кода сервиса корзины

На рисунках 21 - 23 изображены контроллеры корзины.

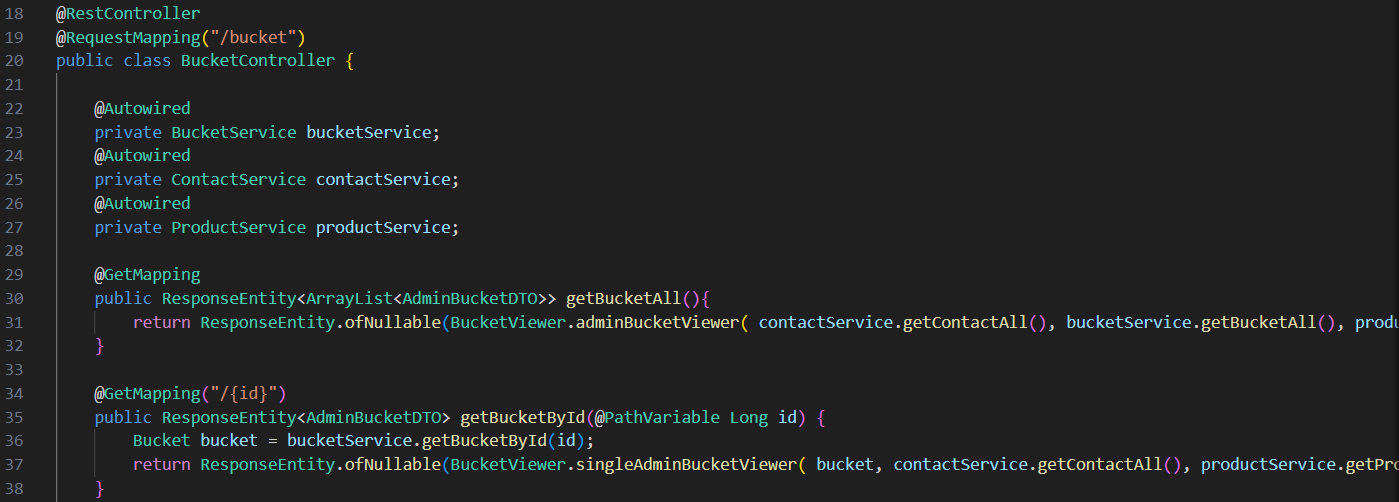


Рисунок 21 – Фрагмент кода контроллера корзины

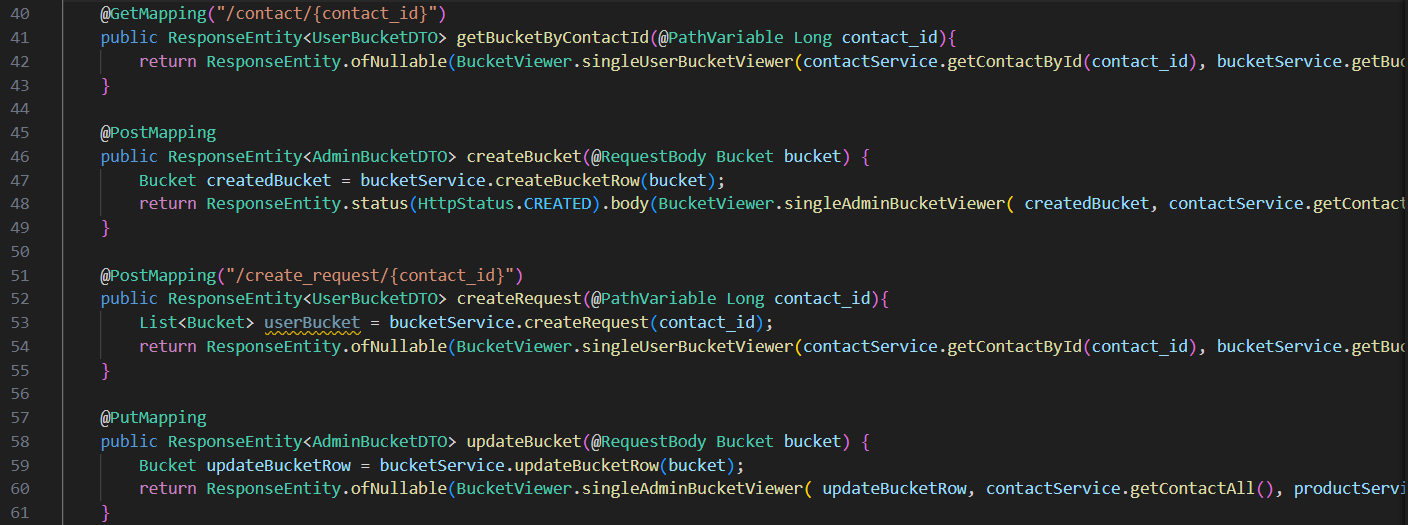


Рисунок 22 – Фрагмент кода контроллера корзины

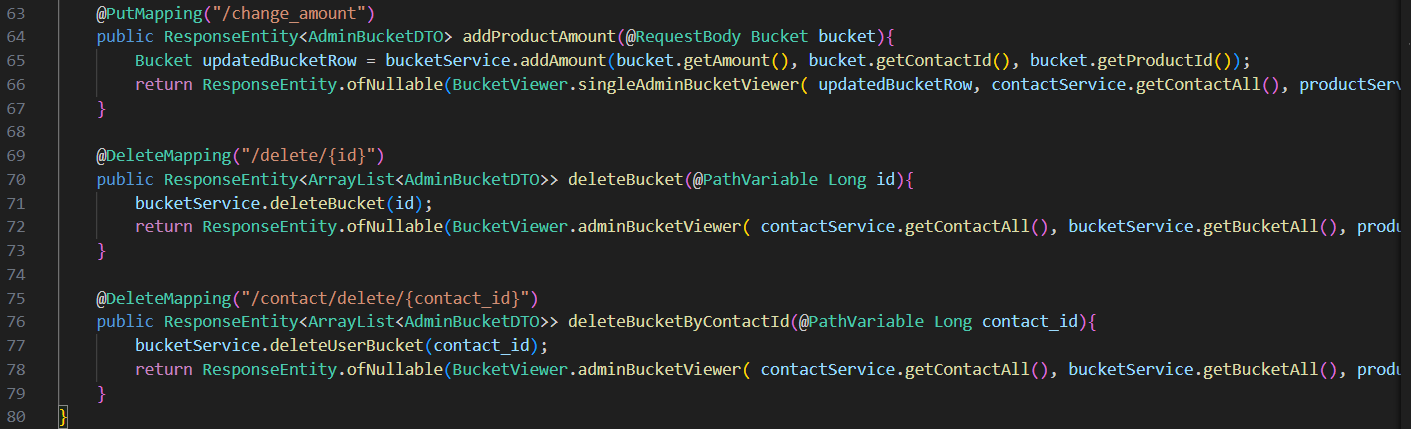


Рисунок 23 – Фрагмент кода контроллера корзины

Результат работы программы представлен на рисунках 24 – 27.

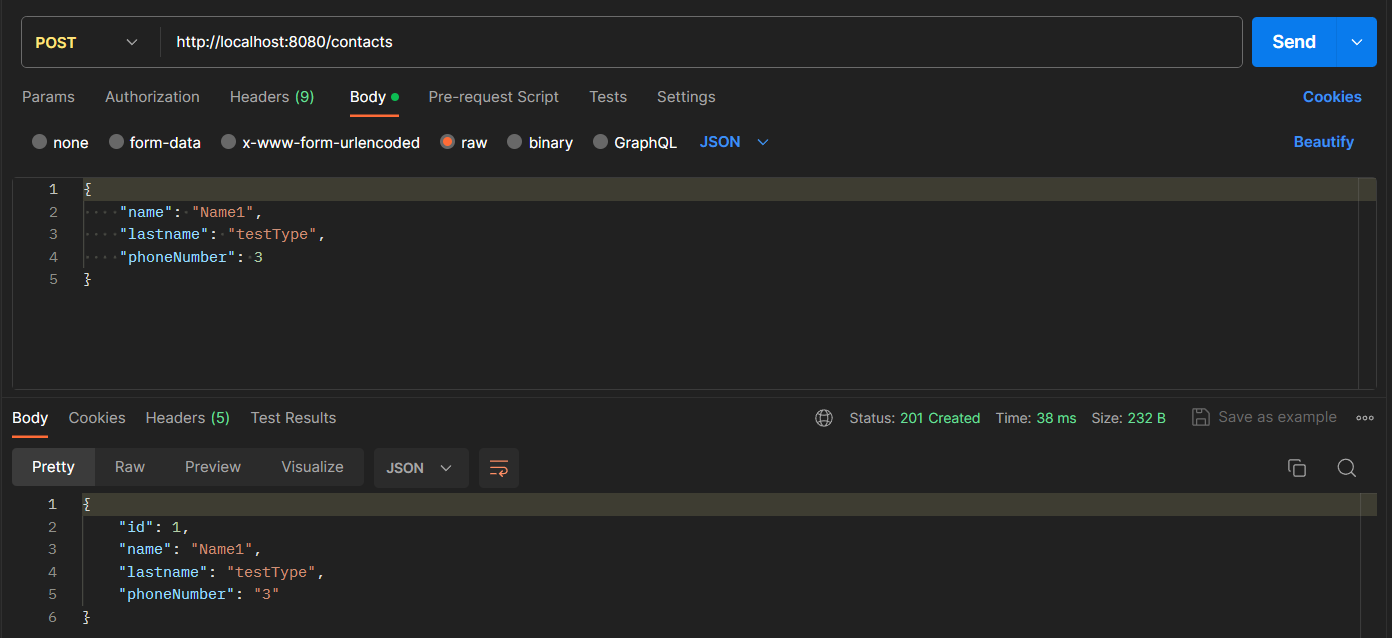


Рисунок 24 – Создание контакта

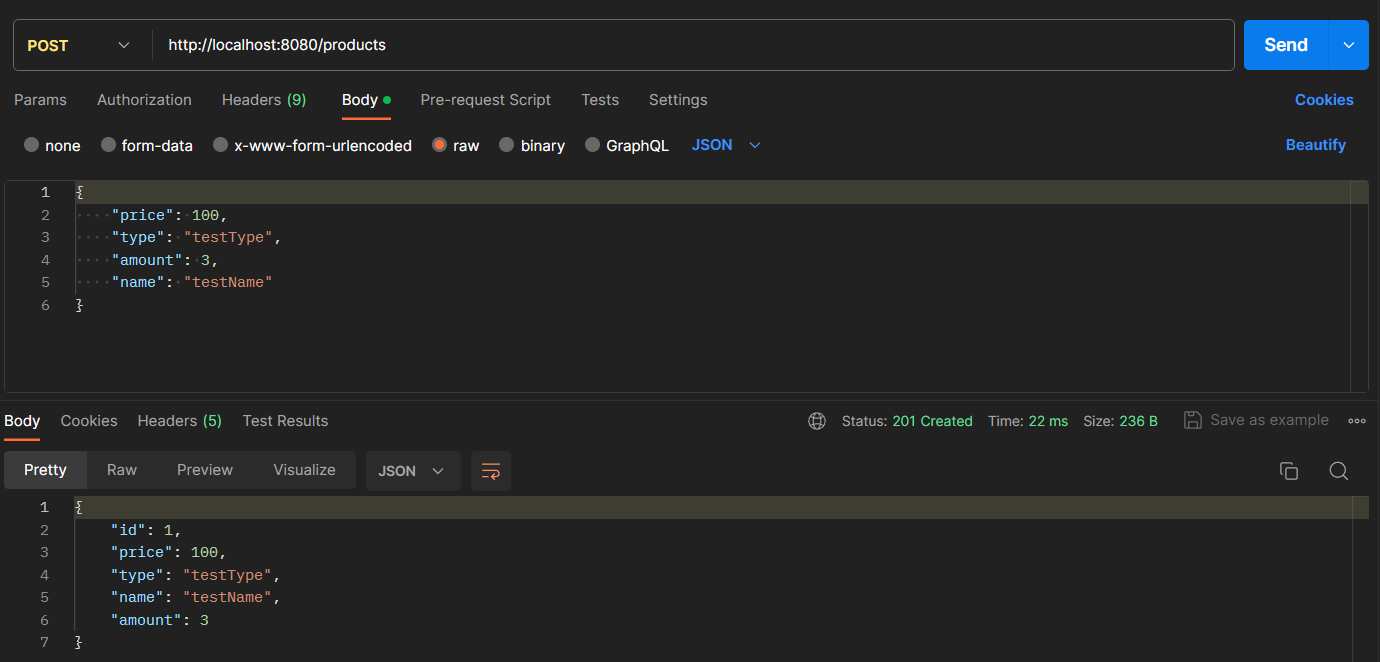


Рисунок 25 – Создание продукта.

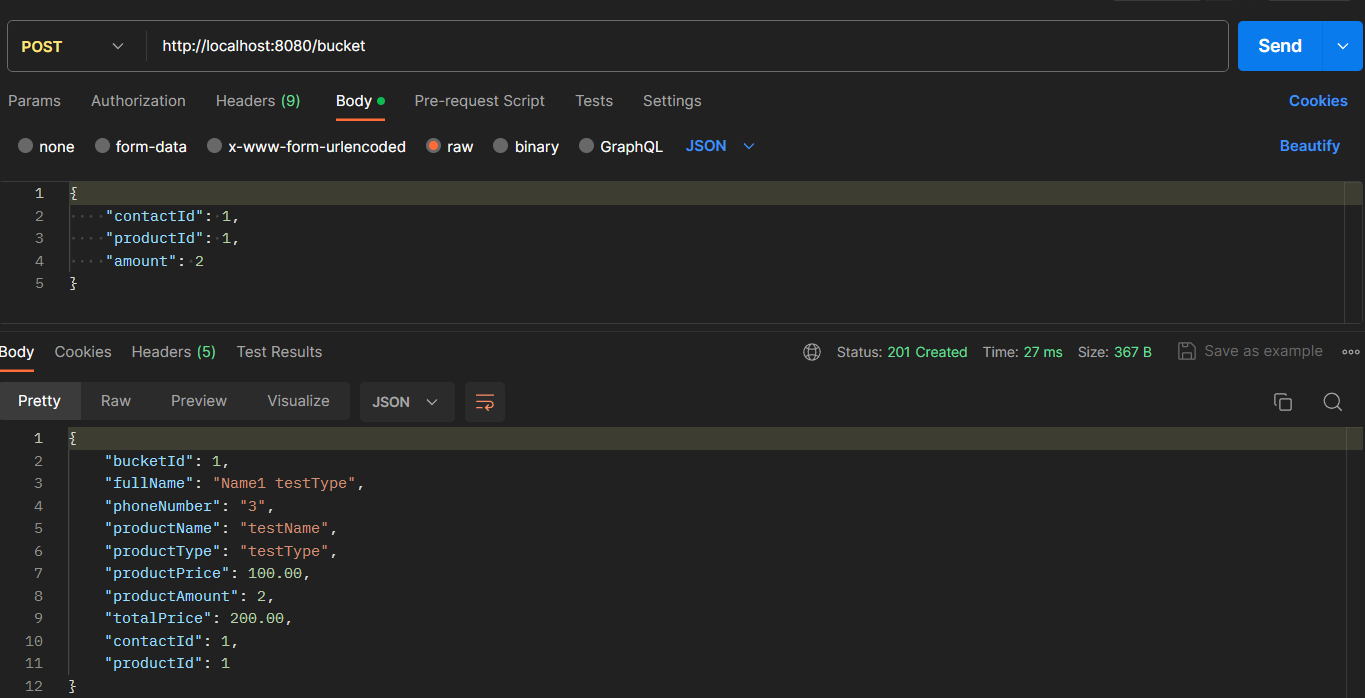


Рисунок 26 – Добавление продукта в корзину

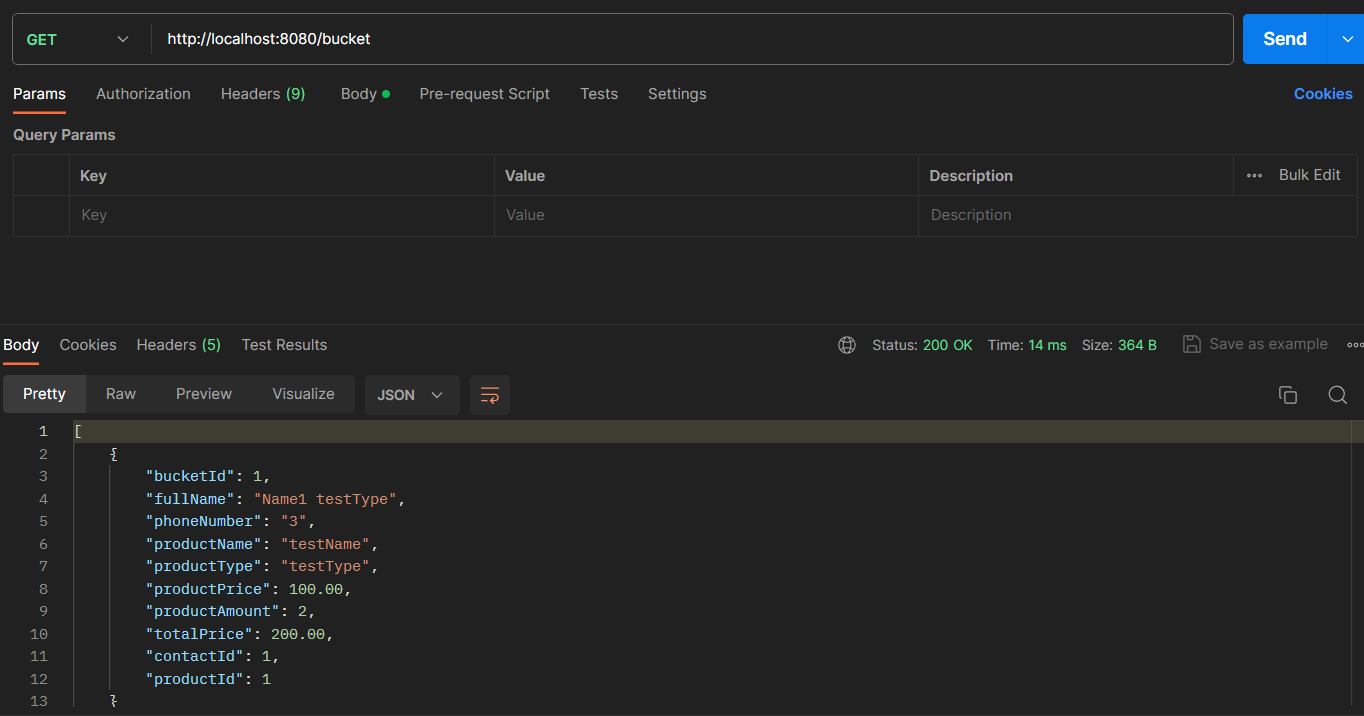


Рисунок 27 – Просмотр корзины

## Выводы

Таким образом, была реализована сущность корзины, в которую можно добавлять товары, изменять их количество, удалять товары из корзины, просматривать ее и выполнять заказ. Все тесты пройдены успешно.

## Ответы на вопросы

Аннотация Service. – Аннотация Service в программировании обозначает класс или компонент, предоставляющий конкретные сервисные функции для использования в приложении. Она часто используется в контексте фреймворков и инструментов вроде Spring в языке Java, где помеченные таким образом классы автоматически обнаруживаются и управляются внутри приложения.

Паттерн MVC. – Паттерн MVC (Model-View-Controller) разделяет приложение на три основных компонента: Модель (Model), Отображение (View) и Контроллер (Controller). Модель представляет данные и бизнес-логику, Представление отвечает за отображение информации, а Контроллер управляет взаимодействием пользователя с системой, обрабатывая входные данные и воздействуя на модель и представление.

Паттерн «Чистая архитектура». – Паттерн «Чистая архитектура» предлагает структуру приложения с легко заменяемыми компонентами, обеспечивающими высокую степень независимости от конкретных технологий и фреймворков. Он подчеркивает важность разделения бизнес-правил от деталей реализации, повышая гибкость и поддерживаемость системы.

Слои RESTful приложения. – Слои RESTful приложения включают в себя представление, приложение, домен и слой данных. Представление отвечает за отображение данных и обработку пользовательских запросов, приложение содержит бизнес-логику, домен включает в себя правила бизнес-логики, а слой данных занимается доступом к базе данных и управлением данными.

# Практическая работа №7

## Теоретическое введение.

Spring Security – это мощная и настраиваемая платформа аутентификации и контроля доступа. Это стандарт де-факто для обеспечения безопасности приложений на базе Spring.

Spring Security – это платформа, которая фокусируется на обеспечении аутентификации и авторизации для приложений Java. Как и во всех проектах Spring, реальная мощь Spring Security заключается в том, насколько легко ее можно расширить для удовлетворения пользовательских требований

Обычно настройка безопасности пишется в папке со всеми конфигурациями приложения. Настройка заключается в том, что пользователь создает роли пользователей и с помощью специального интерфейса UserDetails производит проверку имеет ли пользователь доступ к тому или иному ресурсу на сайте.

JSON Web Token (JWT) — это JSON объект, который определен в открытом стандарте [RFC 7519](https://tools.ietf.org/html/rfc7519). Он считается одним из безопасных способов передачи информации между двумя участниками. Для его создания необходимо определить заголовок (header) с общей информацией по токену, полезные данные (payload), такие как id пользователя, его роль и т.д. и подписи (signature).

Простыми словами, JWT — это лишь строка в следующем формате header.payload.signature.  
 Предположим, что мы хотим зарегистрироваться на сайте. В нашем случае есть три участника — пользователь user, сервер приложения application server и сервер аутентификации authentication server. Сервер аутентификации будет обеспечивать пользователя токеном, с помощью которого он позднее сможет взаимодействовать с приложением.

Приложение использует JWT для проверки аутентификации пользователя следующим образом:

1. Сперва пользователь заходит на сервер аутентификации с помощью аутентификационного ключа (это может быть пара логин/пароль, либо Facebook ключ, либо Google ключ, либо ключ от другой учетки).
2. Затем сервер аутентификации создает JWT и отправляет его пользователю.
3. Когда пользователь делает запрос к API приложения, он добавляет к нему полученный ранее JWT.
4. Когда пользователь делает API запрос, приложение может проверить по переданному с запросом JWT является ли пользователь тем, за кого себя выдает. В этой схеме сервер приложения сконфигурирован так, что сможет проверить, является ли входящий JWT именно тем, что был создан сервером аутентификации.

## Задание

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 и содержать JWT token. В качестве СУБД во втором сервисе должен быть использован Redis. Должны быть созданы три роли: USER, SELLER, ADMINISTRATOR.

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

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

ADMINISTRATOR не имеет каких-либо ограничений.

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

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

## Выполнение задания

Реализуем функционал для регистрации и авторизации, рисунки 28-30.

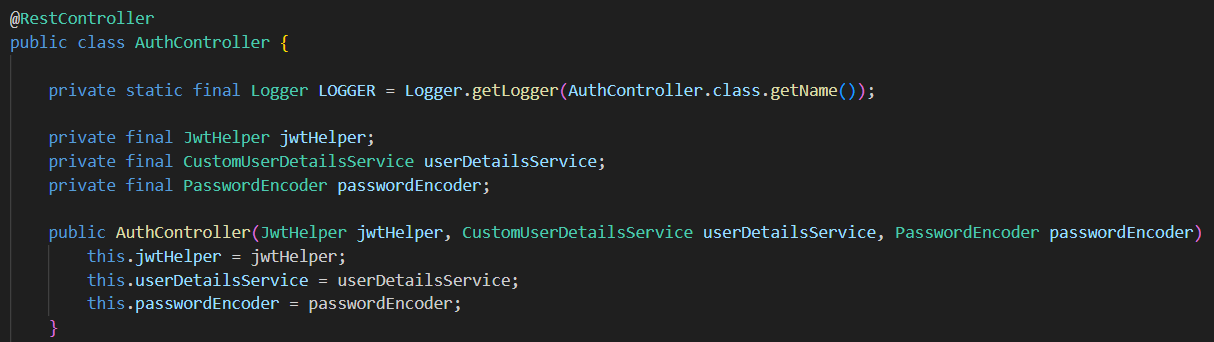


Рисунок 28 – Сервис авторизации и регистрации, 1 часть

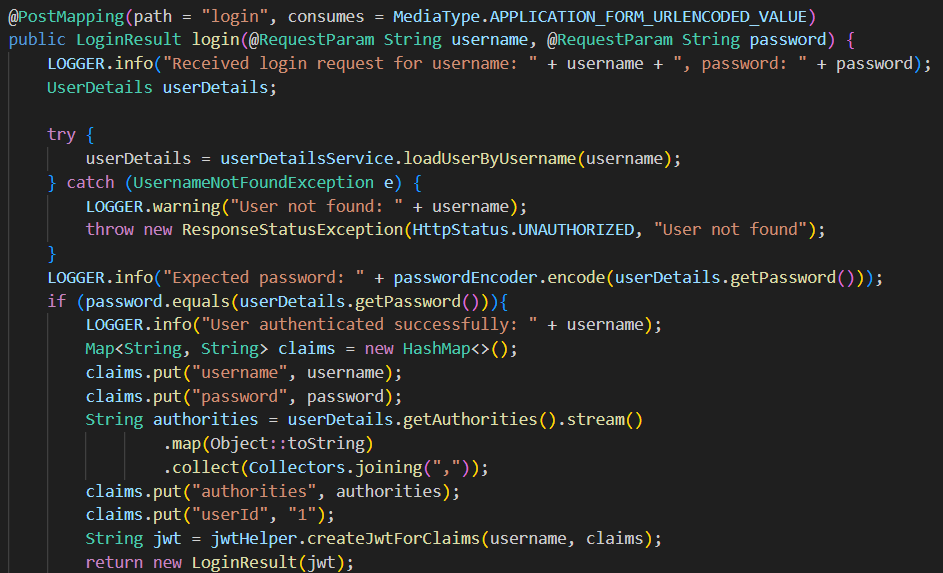


Рисунок 29 – Сервис авторизации и регистрации, 2 часть

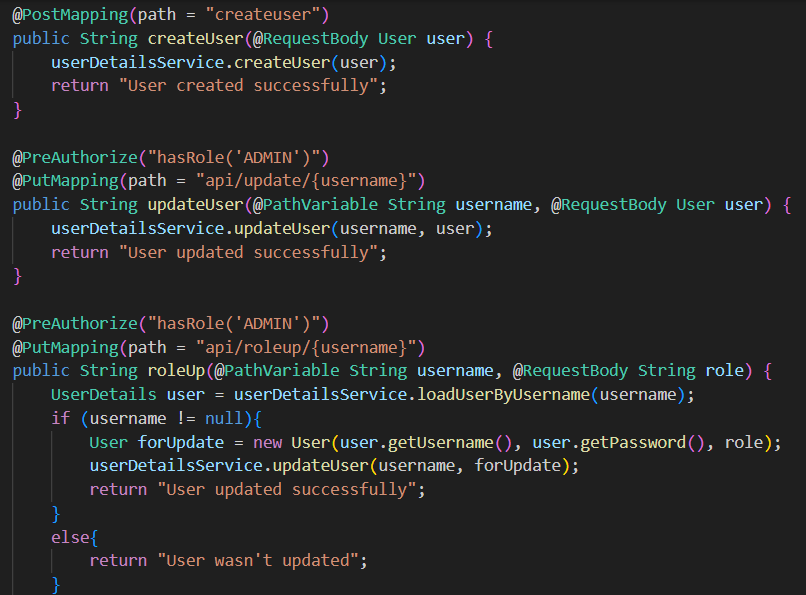


Рисунок 30 – Контроллер авторизации и регистрации

Результаты работы программы представлены на рисунках 31 – 34.

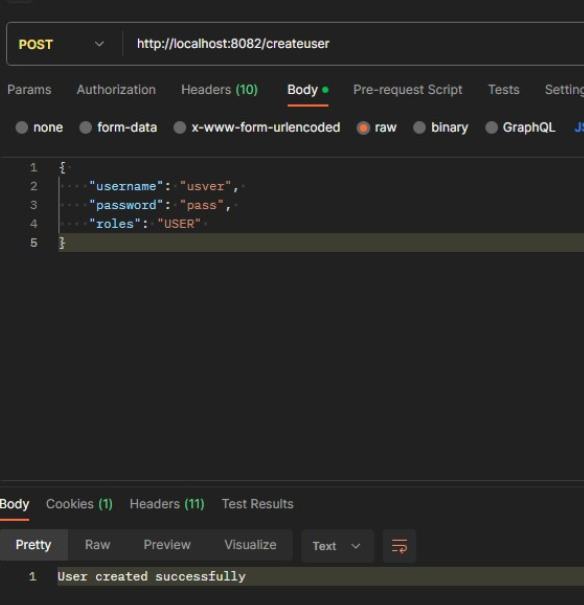


Рисунок 31 – Создание пользователя

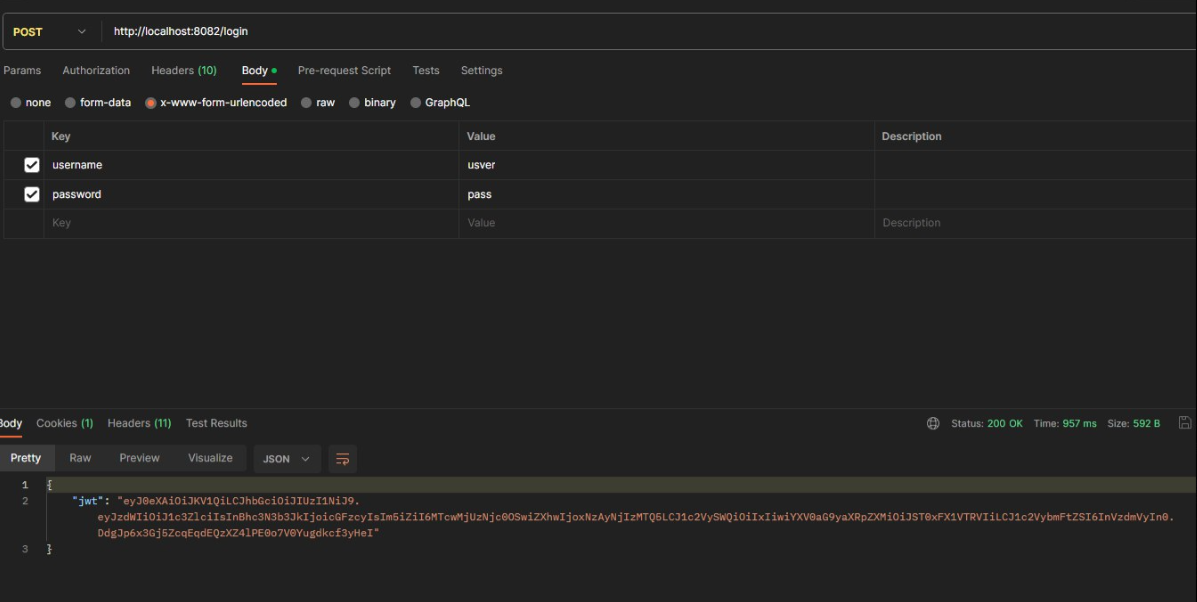


Рисунок 32 – Ввод данных

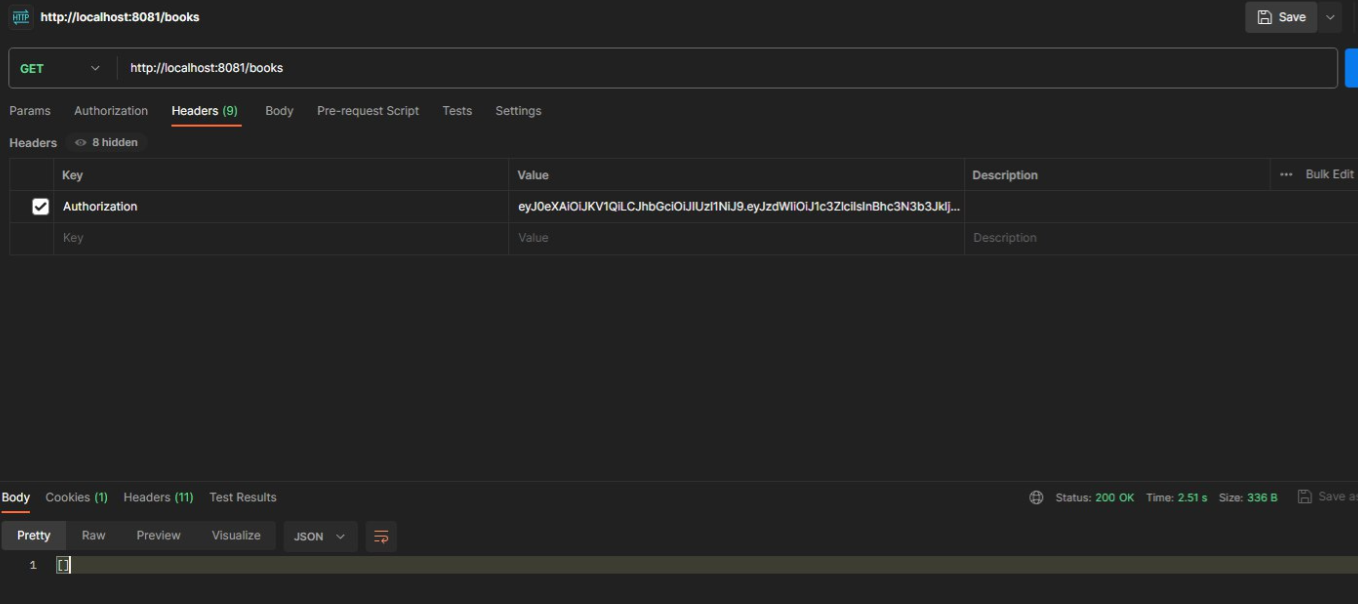


Рисунок 33 – Успешный

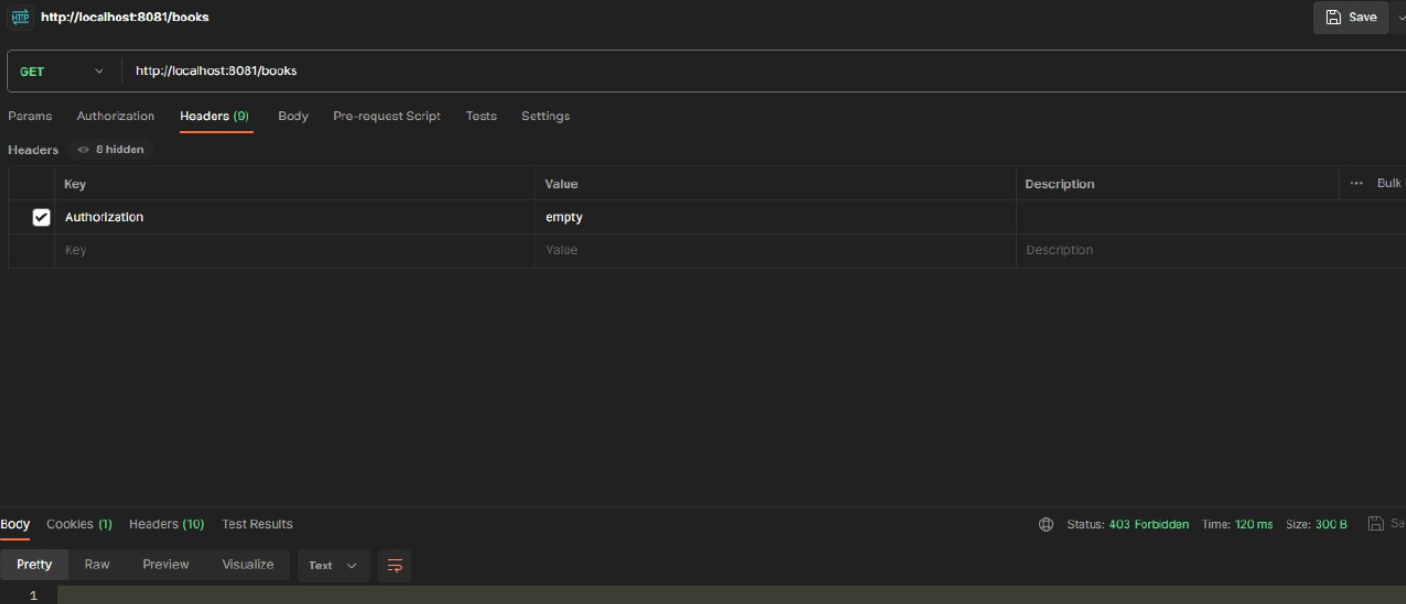


Рисунок 34 – Неуспешный запрос

## Выводы

В ходе выполнения практической работы был реализован микросервис для входа и регистрации пользователей, а также управления ролями пользователей: обычного пользователя, продавца.

## Ответы на вопросы

Spring Security. – Spring Security - это мощный фреймворк для обеспечения аутентификации и авторизации в приложениях на платформе Java. Он предоставляет различные функции, такие как управление сеансами, защита от CSRF-атак, а также поддержка для интеграции с различными системами аутентификации.

JWT Token. – JWT (JSON Web Token) - это формат токена, представляющий собой компактное и самостоятельное средство передачи информации между сторонами. Он состоит из заголовка, полезной нагрузки и подписи. Токен подписывается с использованием секретного ключа, что обеспечивает его подлинность и невозможность изменения. JWT часто используется для аутентификации и передачи данных между клиентом и сервером.

Микросервисная архитектура. – Микросервисная архитектура - это подход к построению приложений, в котором приложение разбивается на небольшие автономные службы, работающие в собственных процессах и взаимодействующие посредством сетевых вызовов. Плюсы включают легкость масштабирования и обновления, но есть и минусы, такие как сложность управления распределенной системой.

Варианты масштабируемости. – Масштабирование может быть вертикальным (увеличение ресурсов на одном сервере) или горизонтальным (увеличение числа серверов). Это также может быть ручным (при добавлении ресурсов вручную) или автоматизированным с использованием инструментов управления нагрузкой.

Вертикальная масштабируемость. – Это увеличение мощности сервера путем добавления процессоров, памяти или хранилища на одном уровне.

Горизонтальная масштабируемость. – Это увеличение мощности системы путем добавления новых серверов или узлов к распределенной сети.

Объяснить принцип сбора микросервисной архитектуры в Docker. – Docker облегчает упаковку, доставку и запуск микросервисов в контейнерах, обеспечивая изолированное окружение для каждого сервиса и упрощая их развертывание в различных средах.

NoSQL DB. Почему в сервисе авторизации удобнее использовать Redis? – Redis, как база данных NoSQL с высокой производительностью в операциях чтения/записи, может быть удобным выбором для сервиса авторизации, где важна быстрая и эффективная работа с кэшированием токенов и хранением временных данных.

Многопотчность — что это такое. Как оно работает в Java? – Многопоточность в Java позволяет одновременно выполнять несколько потоков внутри одного процесса. Она реализуется через классы в пакете java.lang, такие как Thread, и управляется с использованием методов, таких как start(), run() и join().

Класс Thread, расскажите про его методы. – Класс Thread в Java представляет поток выполнения. Некоторые из его методов включают start(), который запускает поток, run(), где размещается код для выполнения, и join(), который ожидает завершения выполнения потока.

## Список использованных источников

1. RestFull приложение со Spring Security и JWT Token – https://habr.com/ru/articles/545610/
2. JWT Token и Spring 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