МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

КУРСОВАЯ РАБОТА

по дисциплине: Разработка клиент-серверных приложений
по профилю: Разработка программных продуктов и проектирование информационных систем
направления профессиональной подготовки: 09.03.04 «Программная инженерия»

Тема: Проектирование и разработка клиент-серверного фуллстек-приложения для доставки продуктов

Студент: Фомичев Роман Алексеевич
Группа: ИКБО-20-21
Работа представлена к защите _____(дата)_____/Фомичев Р.А./
(подпись и ф.и.о. студента)

Руководитель: Коваленко Михаил Андреевич, ст.преп.

Работа допущена к защите_____(дата)_____/Коваленко М.А./
(подпись и ф.и.о. рук-ля)

Оценка по итогам защиты: _____
_____ / _____ /
_____ / _____ /
(подписи, дата, ф.и.о., должность, звание, уч. степень двух преподавателей, принявших защиту)

М. РТУ МИРЭА. 2024 г.

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

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

ЗАДАНИЕ
на выполнение курсовой работы

по дисциплине: Разработка клиент-серверных приложений
по профилю: Разработка программных продуктов и проектирование информационных систем
направления профессиональной подготовки: Программная инженерия (09.03.04)
Студент: Фомичев Роман Алексеевич
Группа: ИКБО-20-21
Срок представления к защите: 20.05.2024
Руководитель: ст.преп. Коваленко Михаил Андреевич

Тема: «Проектирование и разработка клиент-серверного фуллстек-приложения для доставки продуктов»

Исходные данные: HTML5, CSS3, Java, Spring, PostgreSQL, Maven, Git, YandexCloud.

Перечень вопросов, подлежащих разработке, и обязательного графического материала: 1. Провести анализ предметной области для выбранной темы, обосновать выбор клиент-серверной архитектуры и дать её детальное описание с помощью UML для разрабатываемого приложения; 2. Выбрать программный стек для реализации фуллстек разработки, ориентируясь на мировой опыт и стандарты в данной области; 3. Реализовать фронтенд и бекенд части клиент-серверного приложения, обеспечивать авторизацию и аутентификацию пользователя; 4. Разместить готовое клиент-серверное приложение в репозитории GitHub с указанием ссылки в тексте отчёта; 5. Развернуть клиент-серверное приложение в облаке. 6. Провести пользовательское тестирование функционирования минимально жизнеспособного продукта. 7. Разработать презентацию с графическими материалами.

Руководителем произведён инструктаж по технике безопасности, противопожарной технике и правилам внутреннего распорядка.

Зав. кафедрой ИиППО: _____/ Болбаков Р. Г. /, «25» 02 2024 г.
Задание на КР выдал: _____/Коваленко М.А. /, «26» 02 2024 г.
Задание на КР получил: _____/ Фомичев Р.А. /, «26» 02 2024 г.

УДК 004.4

Руководитель курсовой работы: ст.прер. Коваленко Михаил Андреевич

Фомичев Р.А., Курсовая работа направления подготовки «Программная инженерия» на тему «Разработка клиент-серверного фуллстек-приложения для доставки продуктов»: М. 2024 г., МИРЭА – Российский технологический университет (РТУ МИРЭА), Институт информационных технологий (ИИТ), кафедра инструментального и прикладного программного обеспечения (ИиППО) – 41 стр., 29 рис., 15 источн.

Ключевые слова: клиент-серверная архитектура, RESTful-приложение, доставка продуктов, Java, Spring Framework, MVC, PostgreSQL, Vue, Docker.

Целью работы является проектирование и разработка клиент-серверного фуллстек-приложения для доставки продуктов. Спроектирована информационная система, разработано решение для клиентских и серверных частей, разработана база данных и настроен удаленный сервер.

Fomichev R.A., Course work in the field of training "Software Engineering" on the topic "Development of a client-server full-stack application for products delivery": M. 2024, MIREA – Russian Technological University (RTU MIREA), Institute of Information Technology (IIT), Department of Instrumental and Applied Software (IiPPO) – 41 p., 29 fig., 15 sources.

Keywords: client-server architecture, RESTful application, products delivery, Java, Spring Framework, MVC, PostgreSQL, Vue, Docker.

The purpose of the work is to design and develop a web application for products delivery. An information system has been designed, a solution for client and server parts has been developed, a database has been developed and a remote server has been configured.

РТУ МИРЭА: 119454, Москва, пр-т Вернадского, д. 78

кафедра инструментального и прикладного программного обеспечения (ИиППО)

Тираж: 1 экз. (на правах рукописи)

Файл: «ПЗ_РКСП_ИКБО_20_21_ФомичевРА.pdf», исполнитель Фомичев Р.А.

СОДЕРЖАНИЕ

ГЛОССАРИЙ

В настоящем отчете применяют следующие термины:

Серверная часть – это компонент приложения, который выполняет обработку сложных данных, взаимодействует с базой данных и может передавать данные клиенту в заранее установленном формате.

Клиентская часть – это компонент приложения, отвечающий за отображение данных, полученных от сервера, пользователю. Она обеспечивает взаимодействие пользователя с приложением, принимая от него информацию и отправляя её на сервер.

Архитектура приложения – это набор принципов и правил для разработки, которые упрощают создание, поддержку и расширение приложения.

База данных – это структурированное хранилище данных, которое позволяет удобно хранить, искать и обновлять информацию.

СУБД – это программное обеспечение для создания, управления и обработки баз данных.

Контейнеризация – это процесс упаковки приложения вместе с его зависимостями в контейнеры. Эти контейнеры легко перемещаемы и запускаемы в разных средах, обеспечивая изоляцию и независимость приложения от конкретной операционной системы и окружения.

ВВЕДЕНИЕ

В последние годы наблюдается значительное увеличение спроса на услуги доставки продуктов. Это обусловлено несколькими факторами, включая стремительный рост числа занятых людей, желающих экономить время на походах в магазин, и глобальные изменения в поведении потребителей, вызванные развитием технологий и повсеместным доступом к интернету. В условиях быстро развивающегося рынка онлайн-торговли, создание эффективной и надежной системы для доставки продуктов становится все более актуальной задачей. Именно поэтому, темой работы было выбрано фуллстек приложение для доставки продуктов.

Цель данной курсовой работы – разработка комплексного приложения, включающего серверную часть на Java с использованием Spring Framework и принципов REST, базу данных PostgreSQL для хранения данных и клиентскую часть с использованием Vue. Это приложение позволит пользователям удобно заказывать продукты. Для обеспечения высокой отказоустойчивости и масштабируемости приложение будет развернуто на облачном сервере с использованием Docker.

Для достижения поставленной цели необходимо решить следующие задачи:

1) провести анализ предметной области разрабатываемого веб-приложения;
2) сформировать функциональные требования к приложению;
3) обосновать выбор средств ведения разработки;
4) разработать архитектуру веб-приложения;
5) реализовать слой серверной логики веб-приложения с использованием фреймворка Spring, выбранной технологии и инструментария;
6) реализовать слой логики базы данных;
7) разработать слой клиентского представления веб-приложения;

8) оформить пояснительную записку по курсовой работе;

9) подготовить презентацию выполненной курсовой работы.

В ходе выполнения работы были использованы следующие методы: сравнение, анализ, классификация, обобщение, описание и моделирование.

Работа состоит из введения, оглавления, аннотации, глоссария, семи основных разделов, заключения и списка использованных источников.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Описание предметной области

Фуллстек приложение для доставки продуктов – это комплексная система, позволяющая пользователям заказывать продукты из различных магазинов и получать их доставку до двери. Такое приложение объединяет серверную и клиентскую части для обеспечения эффективного взаимодействия между пользователями, магазинами и курьерами.

Основная цель фуллстек приложения для доставки продуктов – упростить процесс заказа и доставки продуктов, обеспечивая удобство и экономию времени для пользователей. Пользователи могут выбирать из широкого ассортимента продуктов, доступных в различных магазинах, оформлять заказы и отслеживать их выполнение в режиме реального времени.

1.2 Анализ ниши веб-приложений для доставки продкутов

Для установления необходимого функционала создаваемого приложения были проанализированы 3 веб-приложения сервисов доставки продуктов: Сбермаркет [1], Яндекс Еда [2], Самокат [3]

На следующих Рисунках 1.1 – 1.3 показаны главные страницы анализируемых сайтов.

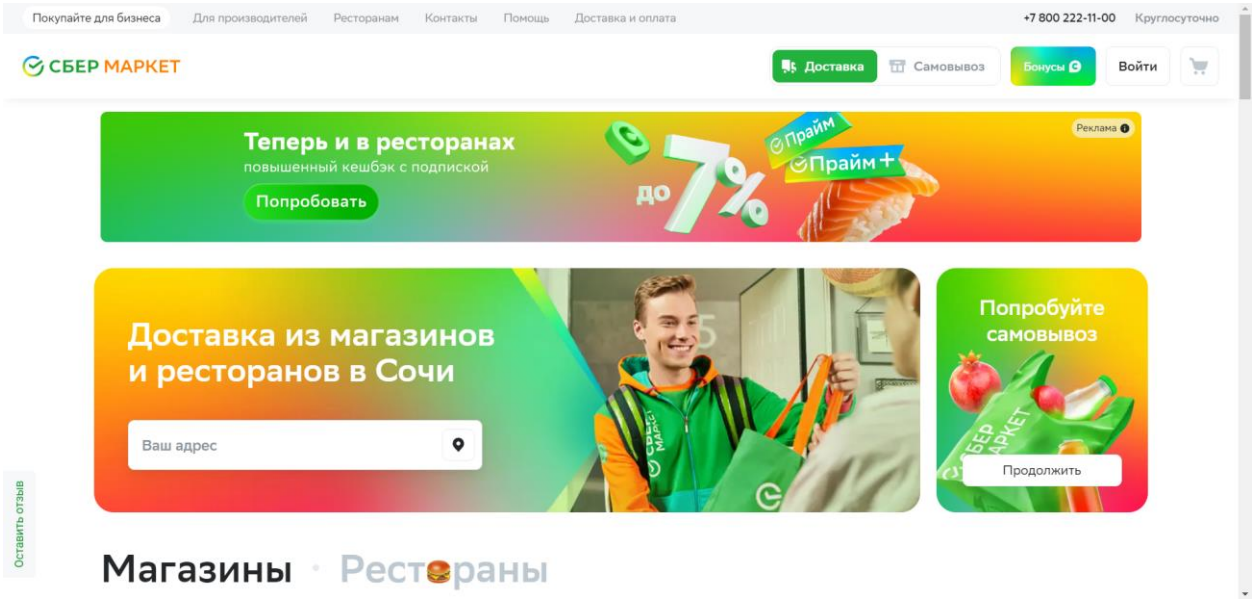

Рисунок 1.1 – Сайт сервиса «Сбермаркет»

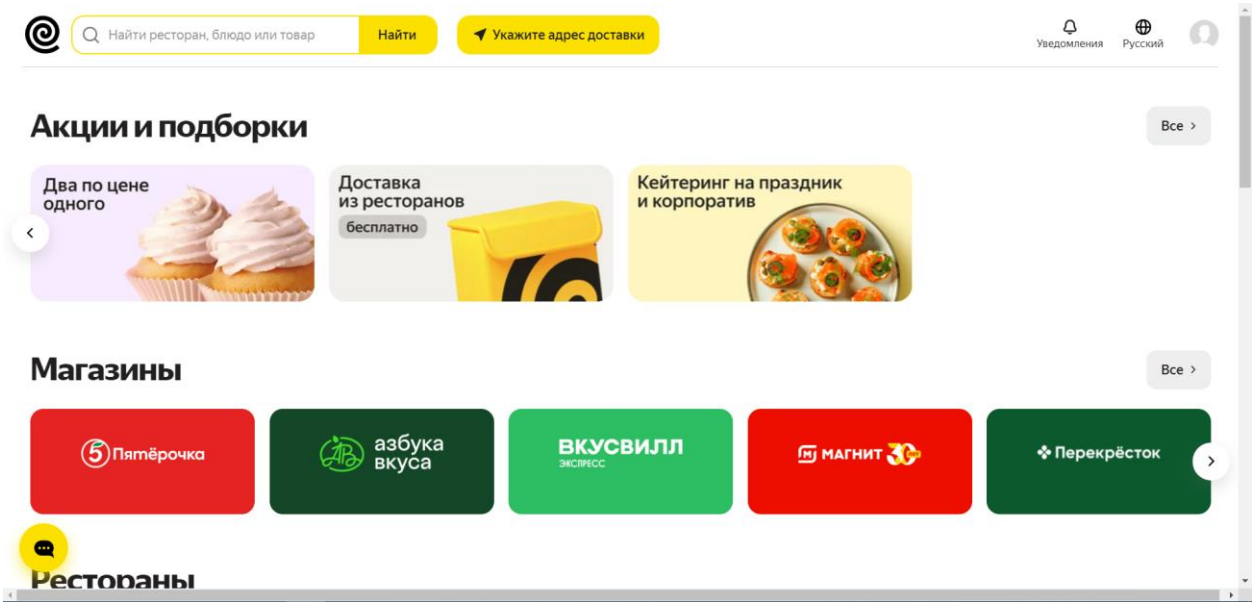

Рисунок 1.2 – Сайт сервиса «Яндекс Еда»

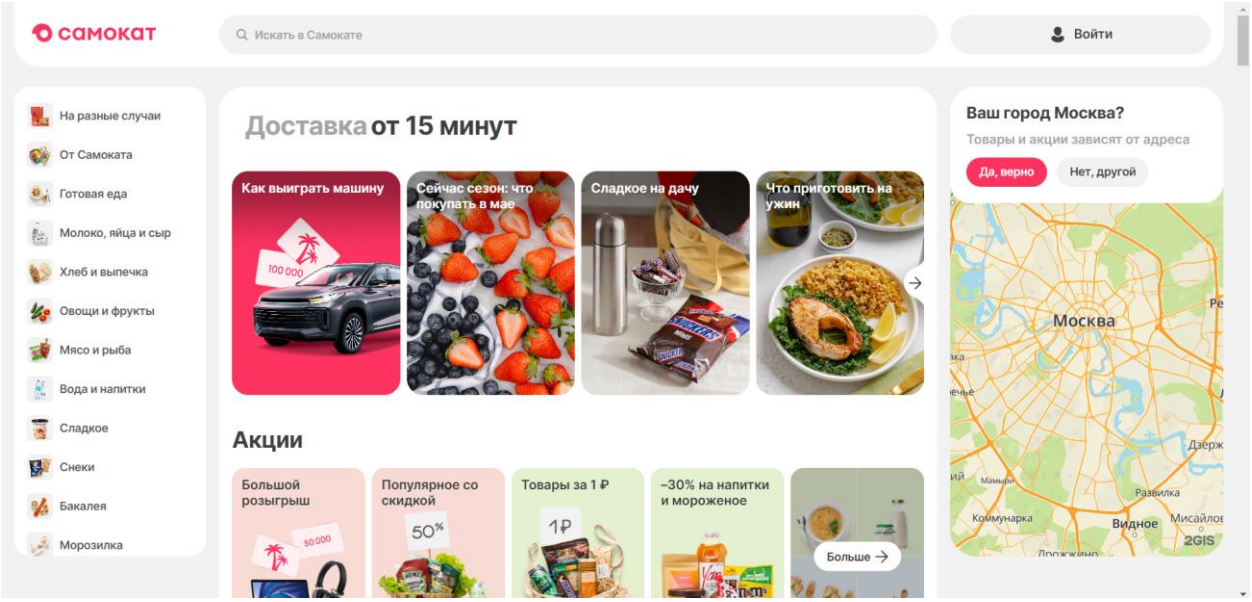

Рисунок 1.3 – Сайт сервиса «Самокат»

1.2.1 Авторизация и аутентификация

Анализ

На всех 3-х сайтах необходима аутентификация для того, чтобы пользователь мог создавать собственные заказы.

Вывод

В разрабатываемом программном продукте должна поддерживаться аутентификация. Так область возможностей каждого пользователя будет ограничена.

1.2.2 Создание заказа

Анализ

Сервисы «Яндекс Еда» и «Самокат» позволяют создать несколько заказов в одно время, а не один, что является удобным, если пользователь забыл что-то добавить при первом заказе.

Вывод

Создаваемый программный продукт должен поддерживать возможность создания нескольких заказов.

1.2.3 История заказов

Анализ

На всех 3-х сайтах реализован функционал просмотра ожидаемых к доставке заказов, а также история когда-либо созданных.

Вывод

Создаваемое приложение должно поддерживать полноценную работу с созданием заказов, просмотром статусов заказов, возможность смотреть, как на ожидаемые доставки, так и на историю заказов.

1.2.4 Пользовательский интерфейс

Анализ

Сайты сервисов «Самокат» и «Сбермаркет» переполнены контентом.

Вывод

Интерфейс должен быть достаточно простой и интуитивно понятный.

1.2.5 Регистрация

Анализ

На сайте сервиса «Сбермаркет» нет функции регистрации нового пользователя.

Вывод

Создаваемое приложение должно поддерживать функционал регистрации новых пользователей.

1.3 Функциональные требования на основе анализа

Сайт должен быть разработан с учетом данных требований:

1. Сайт должен обеспечивать возможность регистрации и аутентификации пользователей.
2. Создаваемое приложение должно поддерживать возможность создания множества заказов для каждого пользователя
3. Создаваемое приложение должно реализовать полноценный функционал для просмотра истории заказов.
4. Должна быть реализована изменения и просмотра статуса заказа.
5. Интерфейс приложения должен быть современным и интуитивно понятным.

2 ОБОСНОВАНИЕ ВЫБОРА ТЕХНОЛОГИЙ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

На этапе, когда ясны все требования к программному продукту, необходимо определить технологический стек, который будет использоваться для достижения поставленных целей.

2.1 Основные используемые технологии

Приложение функционирует в клиент-серверной архитектуре в режиме тонкого клиента, где основная часть обработки данных выполняется на серверной стороне. Для реализации такого подхода было принято решение использовать Spring [4], PostgreSQL [5], Vue [6].

Использование стека фреймворков Spring связано с их признанием как удобного инструмента для создания серверной логики в приложениях любой сложности. Многочисленные библиотеки, доступные для Spring, ускоряют процесс разработки, а автоматическое документирование кода обеспечивает предсказуемость и удобство использования, что выгодно отличает его от других Java-фреймворков[7].

Для хранения данных выбрана система управления базами данных PostgreSQL благодаря её простоте использования, стабильности и надёжности. Эта СУБД предоставляет все необходимые функциональные возможности для разрабатываемого программного продукта и пользуется высокой популярностью в настоящее время.

Vue применяется для разработки интерактивных и масштабируемых веб-приложений. Этот фреймворк позволяет создавать компоненты, которые легко поддерживать и повторно использовать, обеспечивая плавную и отзывчивую работу пользовательского интерфейса.

2.2 Краткое обоснование всех использованных технологий

2.2.1 Язык программирования Java

Этот язык является мощным инструментом для любого разработчика. Он зарекомендовал себя как надёжный помощник в создании серверной части веб-приложений любой сложности. В проекте используется версия Java 17.

2.2.2 Стек фреймворков Spring

Стек фреймворков Spring [8] представляет собой полный набор инструментов и библиотек, предназначенных для упрощения разработки веб-приложений и управления их инфраструктурой.

2.2.3 Spring Boot

Этот фреймворк упрощает создание автономных, готовых к использованию приложений на основе Spring, облегчая их настройку и развертывание. Он предоставляет стандартные настройки по умолчанию и автоматическую конфигурацию, что способствует быстрому запуску проектов и уменьшает объем необходимого кода.

2.2.4 Spring Web

Данный фреймворк используется для создания веб-приложений. Он включает в себя подмодули, наиболее известный – Spring MVC.

2.2.5 Hibernate

Эта библиотека используется для объектно-реляционного отображения в Java. Он упрощает взаимодействие с базами данных, позволяя разработчикам работать с объектами в коде, не беспокоясь о деталях взаимодействия с базой данных. [9]

2.2.6 Spring Data JPA

Данная технология является частью стека Spring, предоставляющая абстракции и удобные инструменты для взаимодействия с базами данных через Java Persistence API [10].

2.2.7 Spring Security

Эта технология используется для обеспечения безопасности веб-приложений. Он предоставляет механизмы аутентификации, авторизации и защиты от атак, обеспечивая надежную защиту приложения [11].

2.2.8 Maven

Система автоматизации сборки проекта, которая управляет зависимостями и создает проекты. Она используется для упрощения процесса сборки, тестирования и развертывания приложений.

2.2.9 Lombok

Библиотека для языка программирования Java, предоставляющая аннотации для автоматической генерации методов кода, таких как геттеры, сеттеры, конструкторы и другие [13].

2.2.10 Postman

Postman - это инструмент для тестирования и разработки API веб-сервисов. Он обеспечивает простой и удобный интерфейс пользователя, который упрощает создание, отправку, тестирование и отладку HTTP-запросов. С помощью Postman можно эффективно взаимодействовать с API, проверять различные конечные точки приложения, управлять параметрами запросов, а также автоматизировать и организовывать тестовые сценарии [14].

2.2.11 IntelliJ IDEA

Интегрированная среда разработки (IDE) для языка программирования Java, разработанная компанией JetBrains. IDE предоставляет мощные

инструменты для создания, отладки и управления проектами Java. Особенности IntelliJ IDEA включают интеллектуальные подсказки, автоматическое исправление кода, интеграцию с системами сборки, анализ кода и многие другие возможности, улучшающие процесс разработки.

2.2.12 Git

Система управления версиями - это инструмент, который применяется для отслеживания изменений в исходном коде в проектах программного обеспечения. Он предоставляет возможность создавать резервные копии проекта на удаленных серверах, обеспечивая надежное хранение и доступ к истории изменений.

2.2.13 GitHub

Веб-платформа, предназначенная для хостинга и совместной разработки программного обеспечения с использованием системы контроля версий Git. Он предоставляет широкий спектр инструментов для управления проектами, отслеживания ошибок, обсуждения кода и совместной работы над проектами.

2.2.14 Structured Query Language (SQL)

Язык запросов, который применяется для управления реляционными базами данных. Он предоставляет стандартные команды для создания, изменения, запросов и удаления данных в базах данных.

2.2.15 PostgreSQL

Система управления реляционными базами данных, часто используемая в веб-разработке и других сферах. Она обеспечивает эффективное хранение, обработку и извлечение данных, поддерживая стандартные SQL-операции. Отличительные особенности PostgreSQL включают высокую надежность, расширяемость и поддержку распределенных транзакций.

2.2.16 HyperText Markup Language (HTML)

Язык разметки, который применяется для создания структуры и представления контента на веб-страницах. Он предоставляет теги для определения различных элементов страницы, таких как заголовки, параграфы, изображения и ссылки.

2.2.17 Cascading Style Sheets (CSS)

Язык таблиц стилей, который используется для оформления веб-страниц. С его помощью определяются стили, цвета, шрифты и расположение элементов на странице, обеспечивая единообразный и привлекательный дизайн.

2.2.18 JavaScript (JS)

Язык программирования, который используется для создания интерактивных элементов на веб-страницах. Он обеспечивает возможность динамического изменения содержимого страницы, взаимодействия с пользователем и обработки событий, делая веб-приложения более динамичными и функциональными.

2.2.19 Vue

JavaScript библиотека для создания интерактивных пользовательских интерфейсов. Она применяет компонентный подход к разработке, что упрощает сопровождение и масштабирование проектов. Благодаря виртуальной DOM, Vue обеспечивает высокую производительность и отзывчивость пользовательского интерфейса, делая веб-приложения более быстрыми и эффективными.

2.2.20 Docker

Контейнеризация приложений стала стандартом развертывания на сегодняшний день, обеспечивая удобное управление и поддержку программного обеспечения. Docker является одним из наиболее

распространенных решений в этой области, обладая значительным влиянием на сферу контейнеризации приложений.

2.2.21 Docker-compose

Использование Docker-compose обеспечивает простоту и эффективность в развертывании и управлении многоконтейнерными приложениями. Он позволяет определить и запустить все контейнеры приложения одной командой через конфигурационный файл. Это способствует ускорению процесса разработки, обеспечивает консистентность окружения и упрощает масштабирование приложения.

3 АРХИТЕКТУРЫ ВЕБ-ПРИЛОЖЕНИЯ

3.1 Архитектура серверной части приложения

3.1.1 Описание архитектуры

Для построения архитектуры веб-приложения важно выявить ключевые бизнес-правила, которые будут лежать в основе разработки системы. В диаграмме вариантов использования UML, представленной на Рисунке 3.1, отражены сценарии взаимодействия, основанные на этих бизнес-правилах.

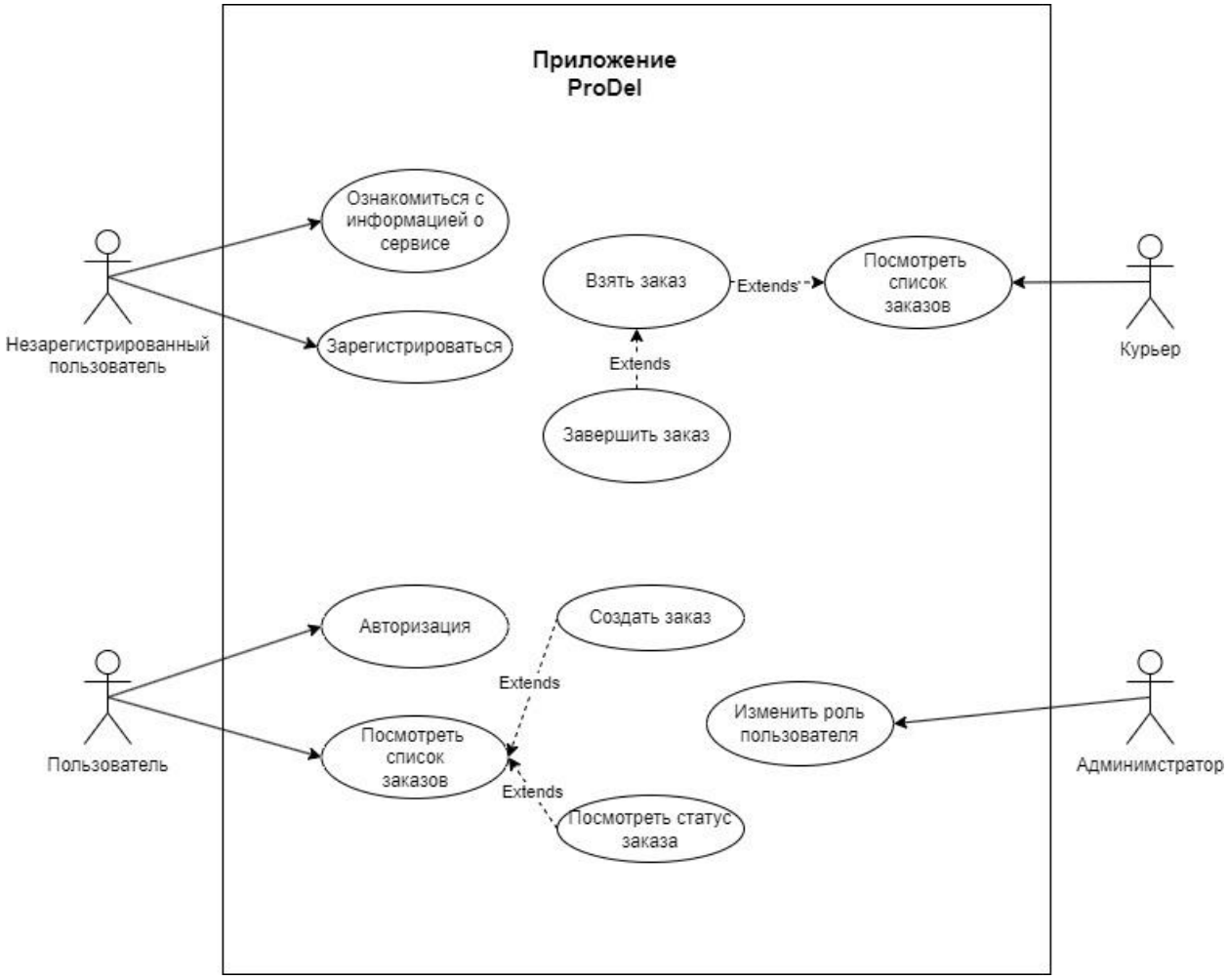

Рисунок 3.1 – Диаграмма вариантов использования веб-приложения

В проекте будем придерживаться архитектуры MVC [15]. Этот подход предполагает четкое разделение кода приложения на три основных компонента: модель, представление и контроллер.

Модели отвечают за управление данными и бизнес-логикой приложения. Они обрабатывают данные и выполняют операции, связанные с бизнес-процессами.

Представления отвечают за визуализацию данных и взаимодействие с пользователем. Они отображают информацию, полученную от сервера, и позволяют пользователю взаимодействовать с приложением.

Контроллеры являются посредниками между моделями и представлениями. Они обрабатывают запросы от пользователя, взаимодействуют с моделями для получения данных и передают эти данные представлениям для отображения.

В приложениях, где разработка клиентской и серверной частей разделены, слой представления представляет собой данные, отправляемые клиенту (например, браузеру или мобильному приложению) в формате JSON.

Для обработки RESTful запросов в нашем приложении мы используем REST-контроллеры. Они принимают данные в формате JSON и используют библиотеку Jackson для их обработки и преобразования в объекты Java.

Понятие "модель" в нашем приложении разделяется на "сервисы" и "репозитории". Сервисы представляют собой слой бизнес-логики, а репозитории взаимодействуют с базой данных через SQL-запросы.

Архитектура MVC помогает нам ясно разграничить ответственность между компонентами, что улучшает читаемость кода и облегчает его сопровождение и масштабирование.

3.1.2 Пример реализации архитектуры

Сначала запрос от клиента приходит в rest-контроллер для обработки запросов по заказам. Фрагмент кода представлен на Листинге 3.1.

Листинг 3.1 – Фрагмент контроллера для обработки запросов по заказам

```
@RestController
@RequestMapping("/api/orders")
public class OrderController {
    @Autowired
    private OrderRepo orderRepo;

    @GetMapping
    @PreAuthorize("hasAuthority('3') || hasAuthority('2')")
    public List<OrderModel> getOrderList() {
        return orderRepo.findAll();
    }
}
```

После ответственность за обработку запрошенных данных передается репозиторию, связанному с сущностью «заказ». Фрагмент репозитория представлен на Листинг 3.2. Репозиторий представляет собой интерфейс, который определяет методы для взаимодействия с базой данных.

Листинг 3.2 – Репозиторий для работы с заказами

```
@Repository
public interface OrderRepo  extends JpaRepository<OrderModel,
Integer> {
    List<OrderModel> findByIdUser(Integer userId);
    List<OrderModel> findByIdCourier(Integer courierId);
    List<OrderModel> findByStatus(String status);
}
```

В качестве ответа сервер возвращает объект OrderModel. Структура ответа сервера представлена на Листинге 3.3

Листинг 3.3 – Структура ответа сервера

```
@Data
@Entity
@Table(name = "orders")
public class OrderModel {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(name = "status")
    private String status;
```

Продолжение листинг 3.3

```
    @Column(name = "end_point")
    private String endPoint;

    @JoinColumn(name = "id_user")
    private Integer idUser;

    @JoinColumn(name = "id_courier")
    private Integer idCourier;

    @JoinColumn(name = "id_stock")
    private Integer idStock;
}
```

3.2 Архитектура клиентской части приложения

3.2.1 Описание архитектуры

Для разработки клиентской части веб-приложений существует достаточно много различных архитектур, таких как: модульная, атомная, FSD-архитектура. Однако из-за достаточно малого размера приложения был выбран обычный компонентный подход. Пример работы такой архитектуры для представлен на Рисунке 3.4.

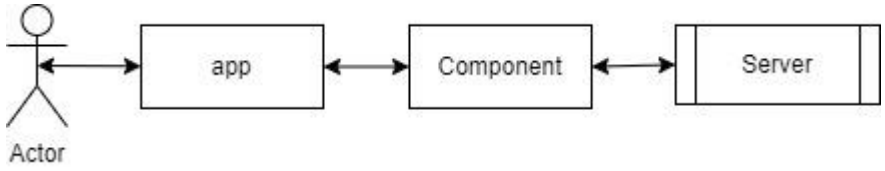

Рисунок 3.4 – Пример работы архитектуры

3.2.2 Пример реализации архитектуры

Предположим, что пользователь уже авторизован. Пользователь попадает на страницу со списком заказов. Фрагмент компонента страницы заказов представлен на Листинг 3.5.

Листинг 3.5 – Фрагмент компонента страницы заказов

```
const getOrders = async () => {
        await getPermissions();
        let request = `${BASE_URL}/api/orders`;
        if (idRole.value == 1) {
            request += '/user/' + id.value;
        } else if (idRole.value == 2) {
            request += '/courier/' + id.value;
        }
        let data = null;
        try {
            const response = await axios.get(request, {
                auth: {
                    username: username.value,
                    password: password.value
                }
            });
            data = response.data;
        } catch (error) {
            console.error('The request failed: ', error);
        }
        if (idRole.value == 1 || idRole.value == 2) {
            ordersData.value = data.filter(order =>
order.status !== 'Delivered');
        } else {
            ordersData.value = data;
        }
    }
```

При попадании на страницу отправляется запрос на сервер для получения данных о заказах, затем идет их отрисовка.

4 РАЗРАБОТКА БАЗЫ ДАННЫХ

4.1 Определение сущностей

Пользователи: Таблица «users» представляют собой учетные записи всех пользователей приложения. Сущность User.

Роли: Таблица «roles» представляет собой данные о ролях пользователей. Сущность Role

Склады: Таблица «stocks» представляет собой данные о складах. Сущность Stock.

Заказы: Таблица «orders» представляет собой данные о заказах. Сущность Order.

4.2 Создание сущностей на уровне СУБД

В качестве СУБД используется PostgreSQL. Структура базы данных представлена на рисунке 4.1.

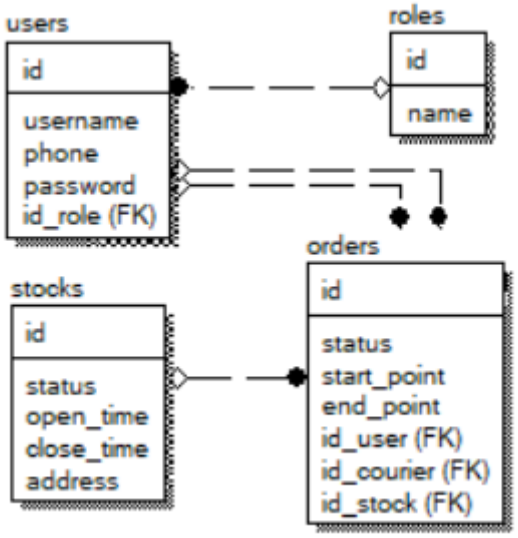

Рисунок 4.1 – Схема базы данных

5 РЕАЛИЗАЦИЯ СЛОЯ СЕРВЕРНОЙ ЛОГИКИ

5.1 Структура проекта

В директории проекта содержатся технические файлы и директории. Файл pom.xml отвечает за конфигурацию проекта. Директория src содержит исходные файлы проекта. Содержимое директории серверной части представлено на Рисунке 5.1.

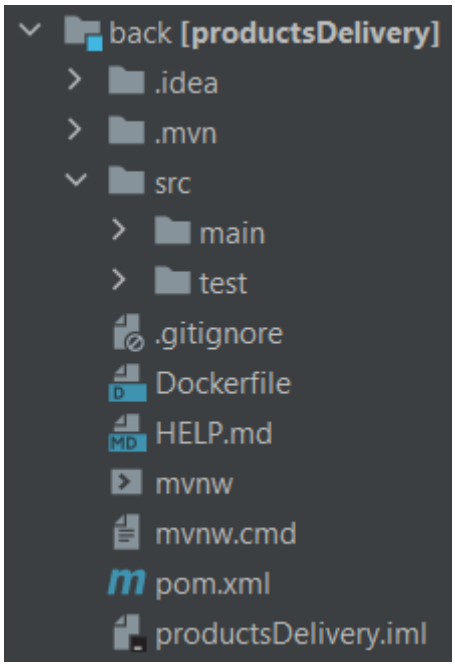

Рисунок 5.1 – Структура проекта

На Рисунке 5.2 показана структура главной директории разработки – main. В директории java располагается исходные файлы кода на языке Java. В директории resources хранятся ресурсы, которые используются в приложении. Это могут быть файлы конфигурации, изображения, шаблоны и другие.

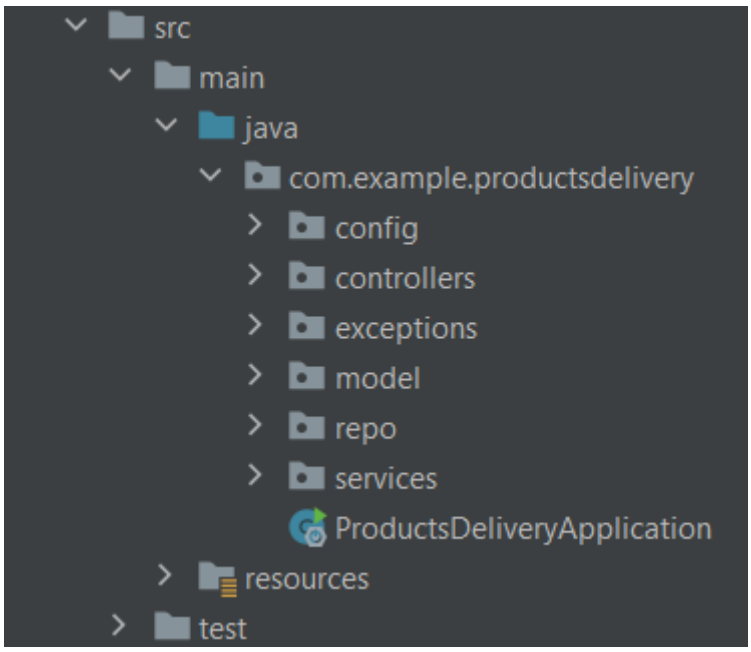

Рисунок 5.2 – Структура директории main

5.2 Конфигурации проекта

В работе используются Spring Security для авторизации. Код для конфигурации securityFilterChain представлен на Листинге 5.2.

Листинг 5.2 – Конфигурация securityFilterChain

```
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity
http) throws Exception {
        return http.csrf(AbstractHttpConfigurer::disable)
                .cors(AbstractHttpConfigurer::disable)
                .authorizeHttpRequests(auth ->
                        auth.requestMatchers("api/**",
"login").permitAll())

.formLogin(AbstractAuthenticationFilterConfigurer::permitAll).ht
tpBasic(Customizer.withDefaults())
                .build();
    }
```

5.3 Тестирование

Данный подраздел включает в себя часть доступных эндпоинтов, остальные эндпоинты также протестированы.

Запрос для регистрации пользователя представлен на Рисунке 5.3.

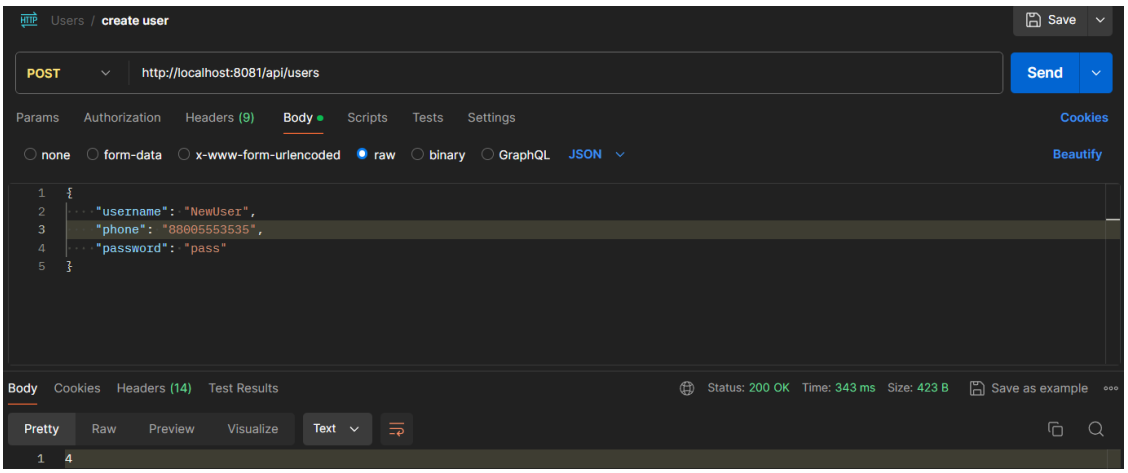

Рисунок 5.3 – Регистрация пользователя

Запрос для авторизации пользователя представлен на Рисунке 5.4.

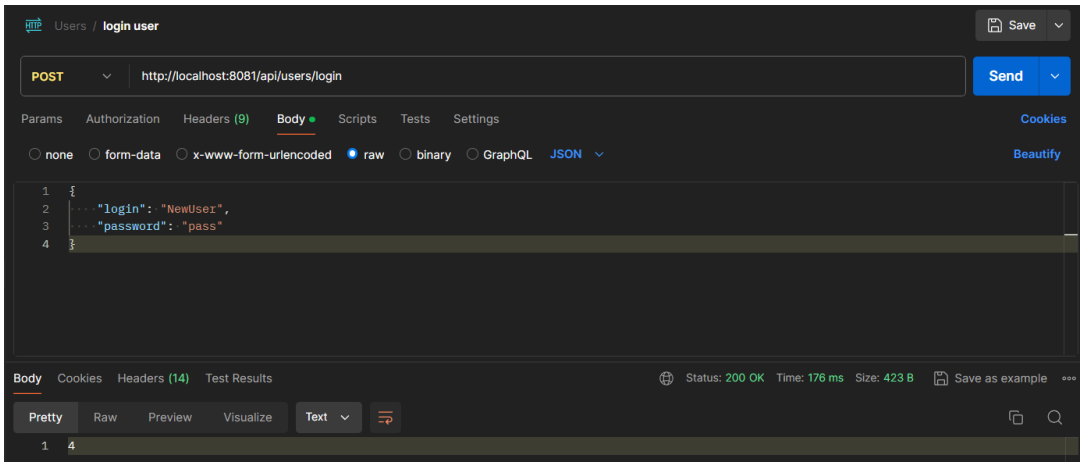

Рисунок 5.4 – Авторизация пользователя

Запрос на получение списка заказов представлен на Рисунках 5.5-5.7.

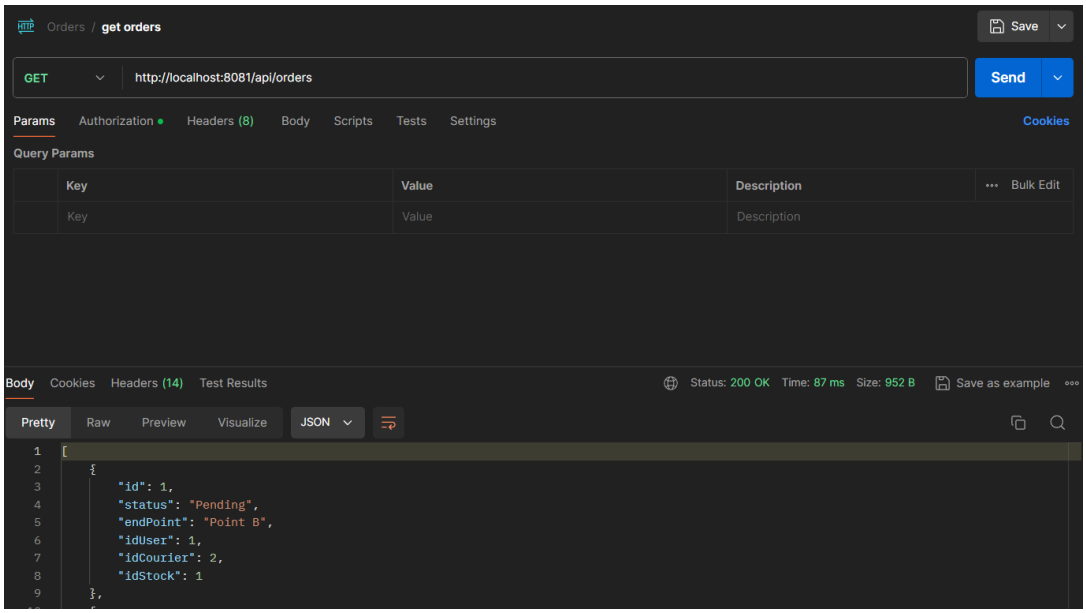

Рисунок 5.5 – Получение всех заказов

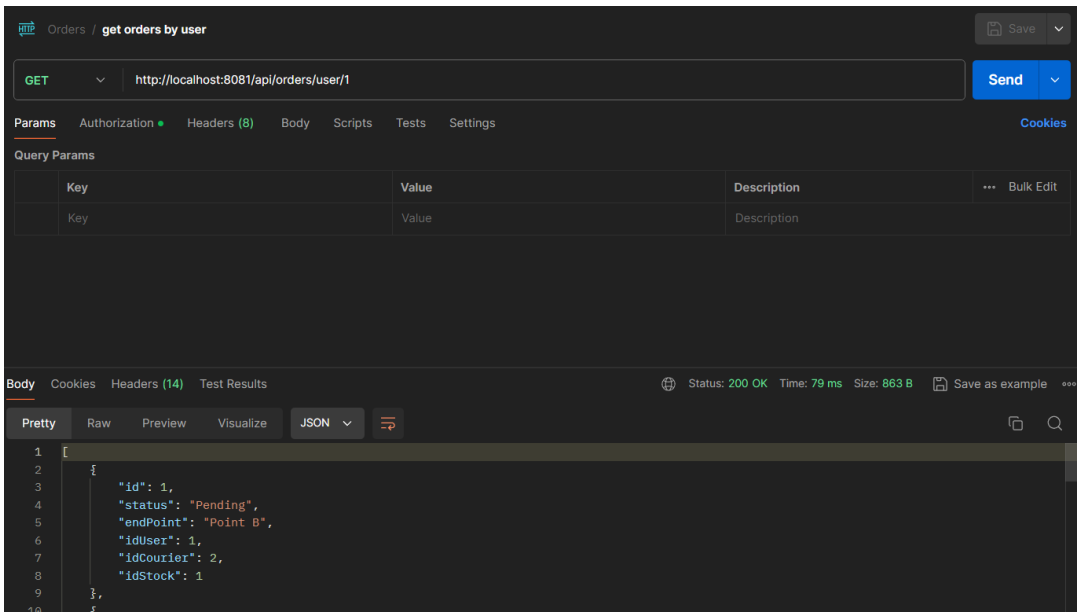

Рисунок 5.6 – Получение заказов пользователя

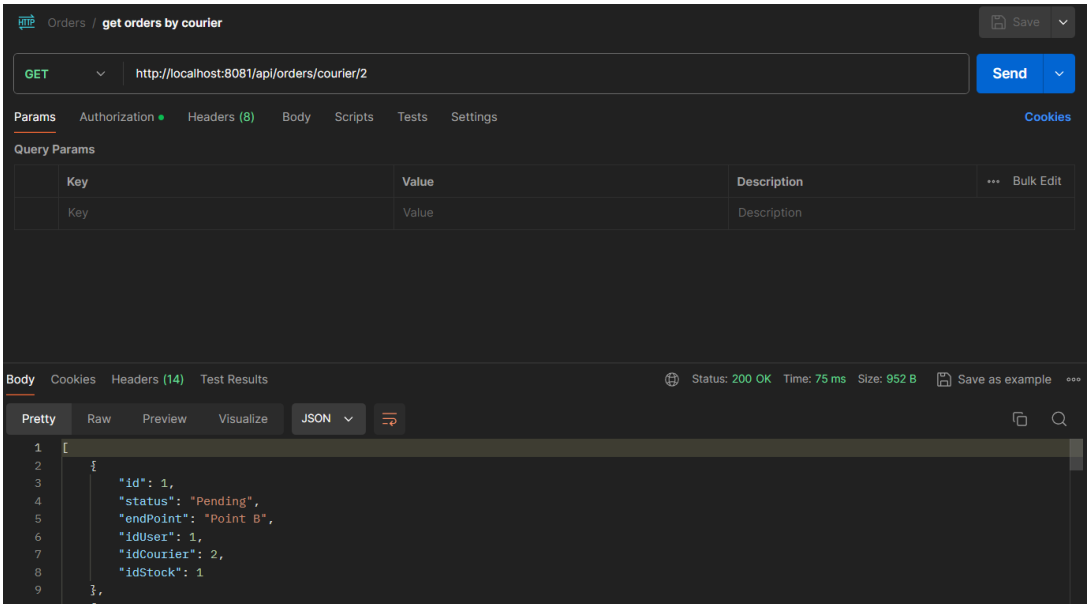

Рисунок 5.7 – Получение активных доставок курьера

Запросы для взятия и завершения курьером заказа представлены на Рисунках 5.8-5.9.

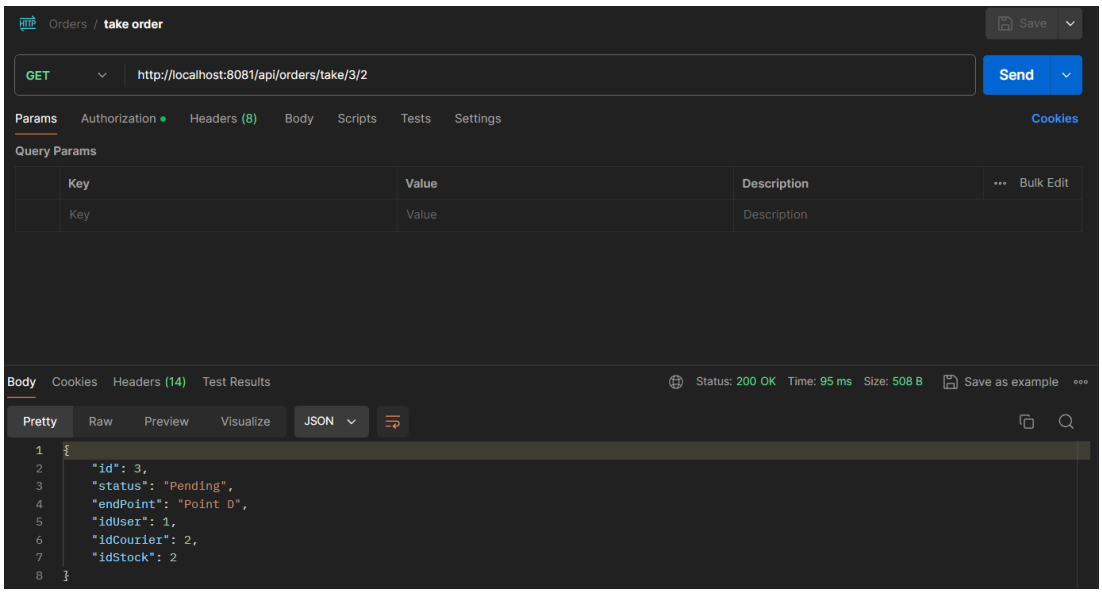

Рисунок 5.8 – Взятие заказа курьером

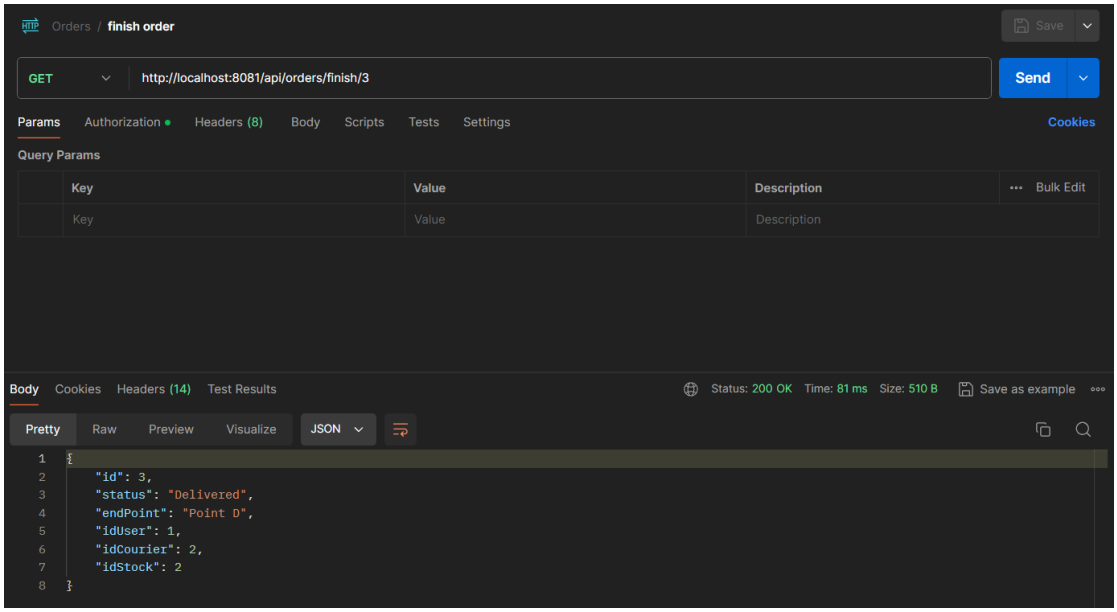

Рисунок 5.9 – Завершение заказа курьером

Запрос для создания заказа представлен на Рисунке 5.10.

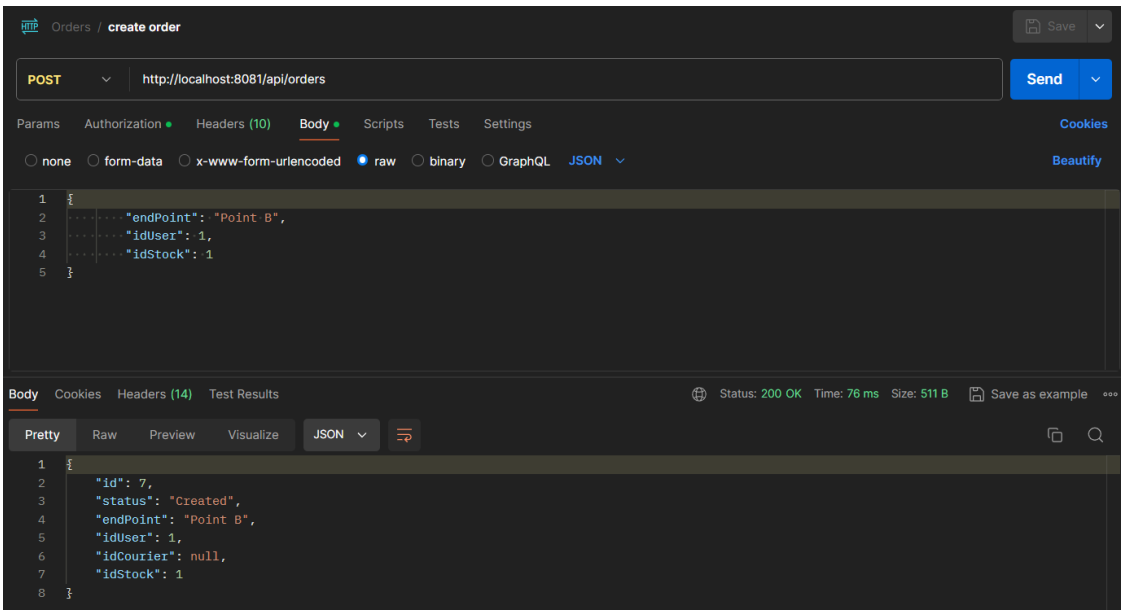

Рисунок 5.10 – Создание заказа

Запрос для изменения роли пользователя представлен на Рисунке 5.11.

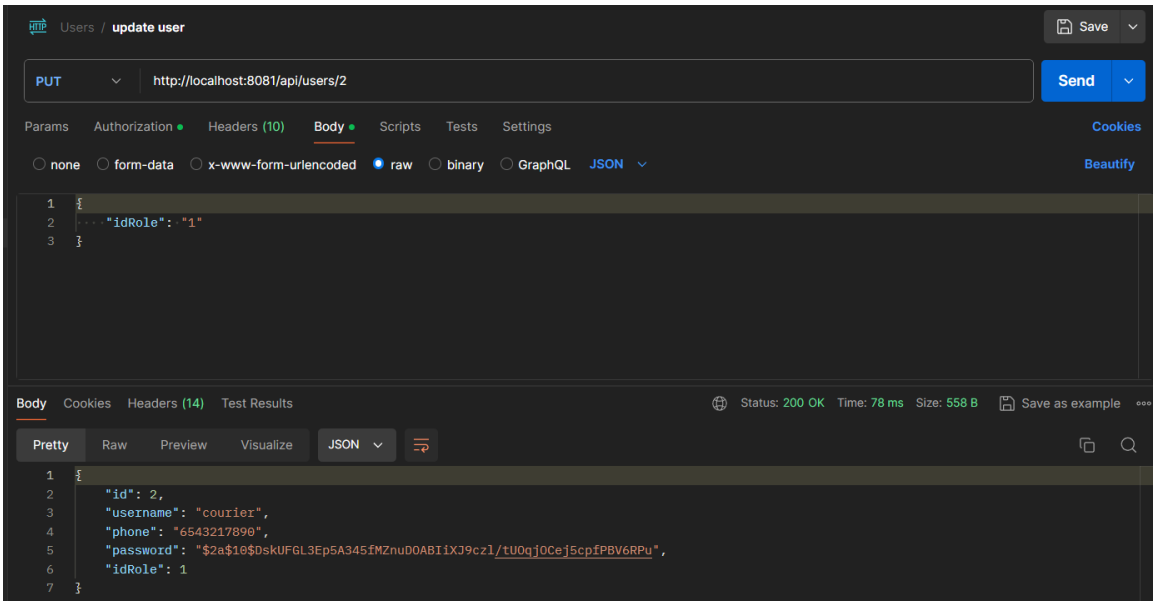

Рисунок 5.11 – Изменение роли пользователя

6 РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ ПРИЛОЖЕНИЯ

6.1 Работа с API сервера

Для отправки запросов на сервер используется axios. Пример запроса представлен на листинге 6.1

Листинг 6.1 – Пример отправки запроса

```
const getOrders = async () => {
        await getPermissions();
        let request = `${BASE_URL}/api/orders`;
        if (idRole.value == 1) {
            request += '/user/' + id.value;
        } else if (idRole.value == 2) {
            request += '/courier/' + id.value;
        }
        let data = null;
        try {
            const response = await axios.get(request, {
                auth: {
                    username: username.value,
                    password: password.value
                }
            });
            data = response.data;
        } catch (error) {
            console.error('The request failed: ', error);
        }
}
```

6.2 Вид конечного приложения

На рисунках 6.1 – 6.14 изображены страницы получившегося ресурса.

Рисунок 6.1 – Главная страница сайта

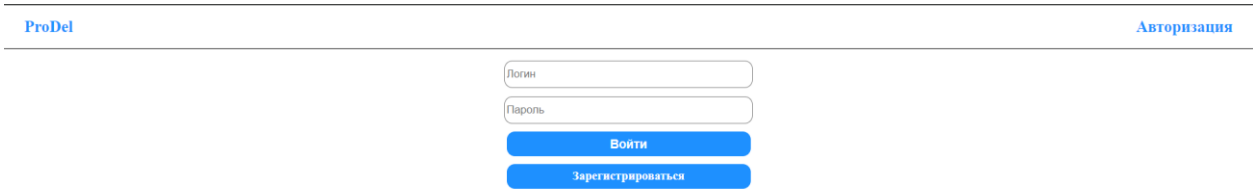

Рисунок 6.2 – Страница авторизации

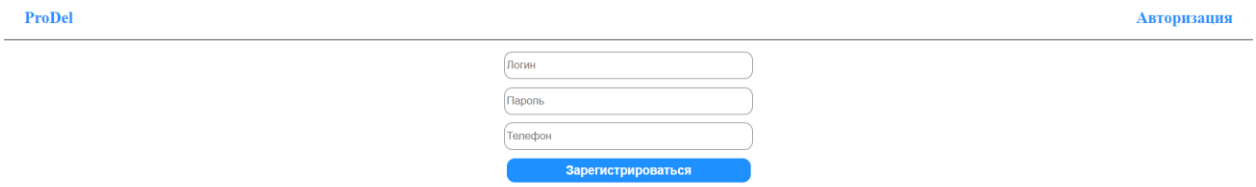

Рисунок 6.3 – Страница регистрации

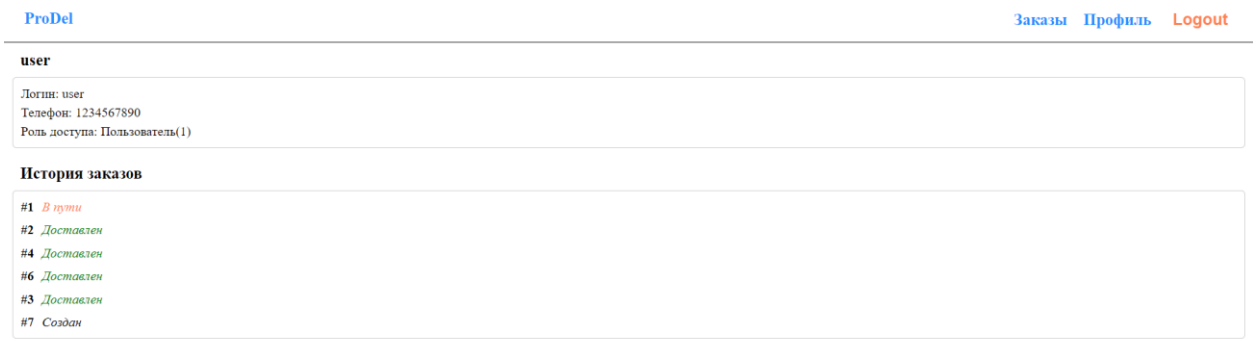

Рисунок 6.4 – Страница профиля

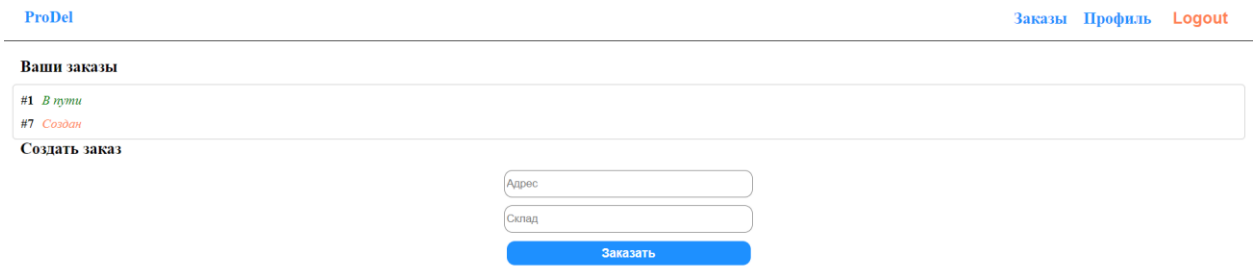

Рисунок 6.5 – Страница заказов пользователя

Рисунок 6.6 – Страница доставок курьера

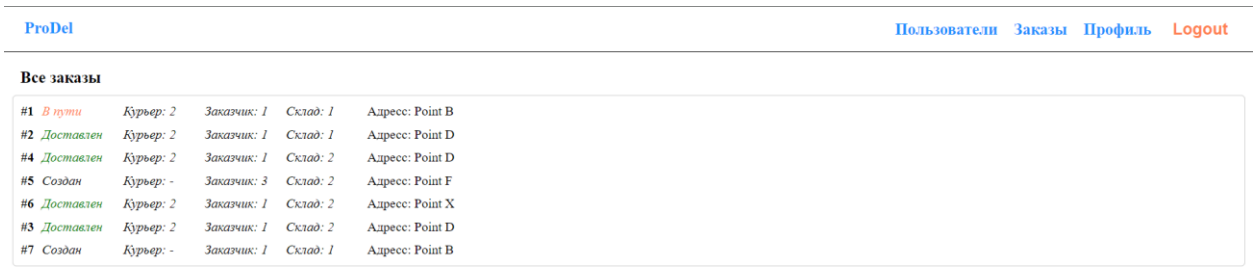

Рисунок 6.7 – Страница заказов администратора

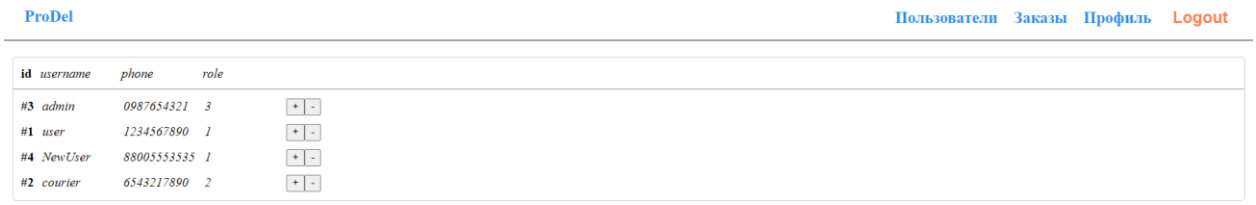

Рисунок 6.8 – Страница управления пользователями

7 КОНТЕЙНИРИЗАЦИЯ И ДЕПЛОЙ

Контейнеризация и развертывание приложения было выполнено с помощью Docker и docker-compose на выделенном сервере.

Содержимое Dockerfile серверной части представлено на Листинге 7.1.

Листинг 7.1 – Dockerfile серверной части

```
FROM openjdk:17-jdk-alpine AS build
ENV HOME=/usr/app
RUN mkdir -p $HOME
WORKDIR $HOME
ADD . $HOME
RUN --mount=type=cache,target=/root/.m2 ./mvnw -f
$HOME/pom.xml clean package -Dskiptests

FROM openjdk:17-jdk-alpine
ARG JAR_FILE=/usr/app/target/*.jar
COPY --from=build $JAR_FILE /app/runner.jar
EXPOSE 8080
ENTRYPOINT java -jar /app/runner.jar
```

Содержимое Dockerfile клиентской части представлено на Листинге 7.2.

Листинг 7.2 – Dockerfile клиентской части

```
FROM node:latest AS builder

WORKDIR /app

ENV PATH /app/node_modules/.bin:$PATH

COPY ./package.json .
COPY ./vite.config.js .
RUN npm install

COPY . .

CMD ["npm", "run", "dev"]
```

Содержимое docker-compose представлено на Листинге 7.3.

Листинг 7.3 – docker-compose

```
services:
  db:
    container_name: db
    networks:
      - delivery
```

Продолжение Листинга 7.3

```
image: postgres:13.3
    environment:
      POSTGRES_DB: "delivery_db"
      POSTGRES_USER: "admin"
      POSTGRES_PASSWORD: "1234"
    volumes:
      - ./db/init.sql:/docker-entrypoint-initdb.d/init.sql
    ports:
      - "5435:5432"

  back:
    container_name: back
    depends_on:
      - db
    networks:
      - delivery
    build:
      context: ./back
      dockerfile: Dockerfile
    ports:
      - "8081:8080"
    volumes:
      - ./service/web/src:/var/www/html

  front:
    container_name: front
    depends_on:
      - back
    build:
      context: ./front
      dockerfile: Dockerfile
    ports:
      - "3000:5173"
    volumes:
      - ./front:/app
      - /app/node_modules
    stdin_open: true
    tty: true
    networks:
      - delivery

networks:
  delivery:
```

На Рисунке 7.1 можно увидеть дашборд выделенного сервера.

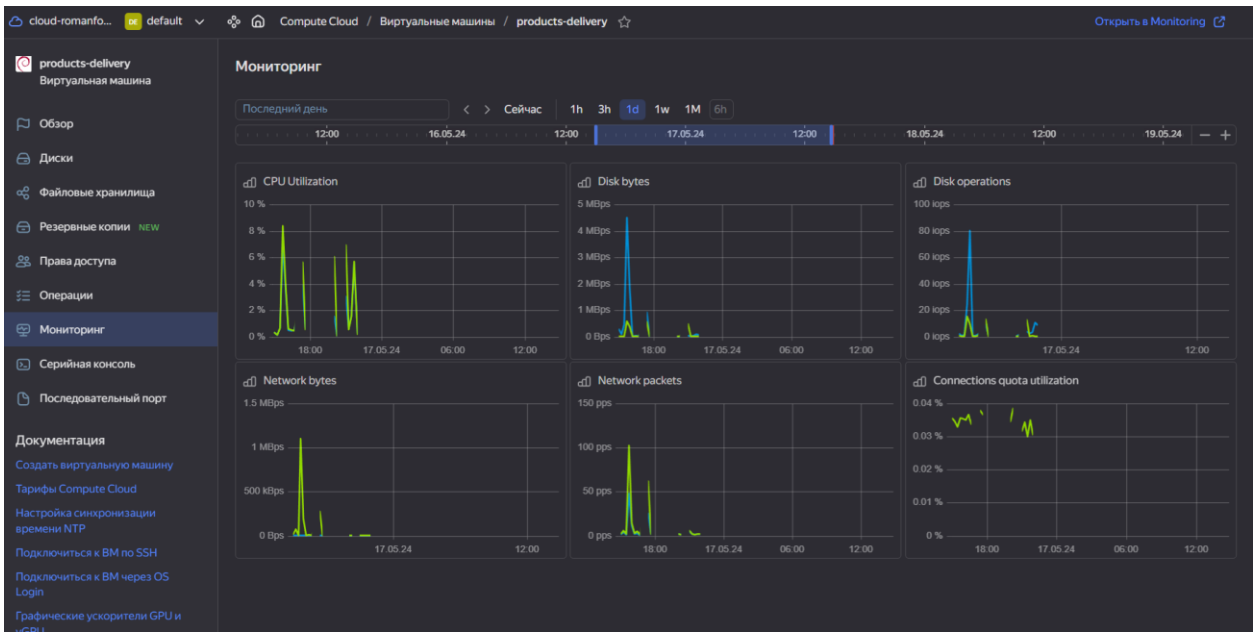

Рисунок 7.1 – Дашборд выделенного сервера

8 ЗАКЛЮЧЕНИЕ

В результате выполнения данной курсовой работы была успешно достигнута поставленная цель – разработка клиент-серверного фуллстек-приложения для доставки продуктов. Проект представлен в репозитории https://github.com/RomanfomicU/ProductsDelivery.

Для достижения этой цели были выполнены следующие задачи: систематизация и углубление теоретических знаний и практических навыков, полученных в процессе изучения соответствующих дисциплин; самостоятельное исследование технической и научной литературы, касающейся архитектуры и разработки программного обеспечения; закрепление навыков использования методов и подходов к разработке программного обеспечения.

В ходе курсовой работы были определены основные сущности системы, спроектированы необходимые таблицы базы данных, разработана и протестирована бизнес-логика приложения. Настроена система регулярных задач и создано решение для управления глобальной конфигурацией проекта без необходимости изменения кода. Полученные навыки позволили эффективно разработать и оформить веб-приложение для платформы доставки продуктов.

8 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Сбермаркет [Электронный ресурс]. – Режим доступа: https://sbermarket.ru (дата обращения 28.04.2024).
2. Яндекс Еда [Электронный ресурс]. – Режим доступа: https://eda.yandex.ru (дата обращения 28.04.2024).
3. Самокат [Электронный ресурс]. – Режим доступа: https://samokat.ru (дата обращения 28.04.2024).
4. Уоллс К. Spring в действии – М.: ДМК Пресс, 2013. – 752 с.
5. Оптимизация запросов PostgreSQL – М: ДМК Пресс, 2022. – 25 с.
6. Vue.js [Электронный ресурс]. – Режим доступа: https://v3.ru.vuejs.org/ (дата обращения 28.04.2024).
7. Сайт компании JavaRush. Программирование на языке Java [Электронный ресурс] – URL: https://javarush.com/ (Дата последнего обращения: 28.04.2024).
8. Раджпут Д. Spring. Все паттерны проектирования. – М: СПб.: Питер, 2019 – 320 с.
9. Бауэр К., Кинг Г. Java Persistence API и Hibernate – М: ДМК Пресс, 2019 – 42 с.
10. Сайт технологии Spring. Документация по использованию Spring Data Jpa [Электронный ресурс] – URL: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/ (Дата последнего обращения: 28.04.2024).
11. Сайт технологии Spring. Начало работы со Spring Security [Электронный ресурс] – URL: https://docs.spring.io/spring-security/reference/index.html (Дата последнего обращения: 28.04.2024).
12. Хоффман Э. Безопасность веб-приложений – СПб.: Питер, 2021. – 354 с.
13. Хабр. Lombok: Полное руководство [Электронный ресурс] – URL: https://habr.com/ru/companies/piter/articles/676394/ (Дата последнего обращения: 7.12.2023).

14. Хабр. Postman: Основы тестирования API и первые шаги с инструментом [Электронный ресурс] – URL: https://habr.com/ru/companies/vk/articles/750096/ (Дата последнего обращения: 7.12.2023).

15. Реализация паттерна проектирования MVC с использованием фреймворка spring MVC / А. И. Тымкив, А. В. Федоренко, Ю. Г. Худасова, О. Г. Худасова // Системная трансформация - основа устойчивого инновационного развития: сборник статей Международной научно-практической конференции, Новосибирск, 20 декабря 2021 года. — Уфа: Общество с ограниченной ответственностью "Аэтерна", 2021. — С. 104-108.— (дата обращения: 30.11.2023)