

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

Петрозаводский государственный университет
Институт математики и информационных технологий

Отчет о прохождении учебной практики по получению профессиональных
умений и навыков профессиональной деятельности

Выполнил студент группы 22307
Максим Павлович Павлов

Направление подготовки:
09.03.04. – Программная инженерия

Место прохождения практики:
**Кафедра Информатики и
математического обеспечения**

Сроки прохождения практики:
05.02.24 – 26.05.24

Руководитель практики:
ст. преподаватель В. М. Димитров

Оценка _____

Дата _____

Подпись _____

Петрозаводск 2024 г.

Содержание

1. ЦЕЛЬ И ЗАДАЧИ ПРАКТИКИ.....	3
2. ОБЗОР ПРОГРАММНЫХ СРЕДСТВ, ИСПОЛЬЗУЕМЫХ В ХОДЕ ПРАКТИКИ	4
3. ОПИСАНИЕ РЕАЛИЗАЦИИ РЕШЕНИЯ.....	7
ЗАКЛЮЧЕНИЕ	16

1. Цель и задачи практики

Цель учебной практики – изучение и применение программных инструментов и технологий для разработки программного обеспечения.

Задачи производственной практики:

- Изучение документации по программному каркасу FastAPI;
- Изучение документации по программному каркасу React;
- Реализация веб приложения с использованием программных каркасов FastAPI и React.

Тема проекта: разработка веб приложения интернет магазина техники.

Описание предметной области: Требуется разработать веб приложения интернет магазина для продажи техники. Каждый товар оснащён одной фотографией и кратким описанием позиции, содержащей менее 250 символов. Каждый зарегистрированный пользователь должен иметь возможность оставлять отзывы на товары с небольшим текстом, содержащим до 250 символов, и оценку по пятибалльной шкале. Главная страница интернет магазина предлагает пользователям товары с самым высоким рейтингом по отзывам и наименьшей стоимости. Работа интернет магазина должна предусматривать три роли пользователей: покупатель, менеджер и администратор. Покупатель может смотреть товары, оставлять отзывы, добавлять товары в корзину, покупать товары, смотреть историю покупок и статус заказов. Менеджер может добавлять новые товары, редактировать и удалять текущие. Администратор же может ещё блокировать пользователей и создавать учетные записи менеджеров. После оплаты пользователем какого-либо заказа веб приложение должно отправлять письмо на почту всем администраторам системы с подробной информацией о заказе.

Репозиторий проекта на GitHub: <https://github.com/GhosT-FlexAgen/jojo-shop>

2. Обзор программных средств, используемых в ходе практики

Uvicorn — это современный, высокопроизводительный веб-сервер, основанный на библиотеке uvloop. Он обеспечивает масштабируемость и эффективность для веб-приложений. В данном проекте Uvicorn выступает в роли веб-сервера, ответственного за обслуживание входящих запросов и управление ответами.

FastAPI — это мощный веб-фреймворк, построенный на основе Starlette. Он предлагает разработчикам удобный и интуитивно понятный способ создания высокопроизводительных API. FastAPI автоматически генерирует интерактивную документацию и предоставляет инструменты для валидации запросов и отладки. В данном проекте FastAPI используется в качестве основного фреймворка для разработки API, обеспечивая простоту и скорость разработки.

CORSMiddleware — это промежуточное программное обеспечение (middleware), предоставляемое FastAPI, для обработки междоменных запросов (Cross-Origin Resource Sharing, CORS). Оно позволяет настроить заголовки, связанные с политикой междоменного доступа, и обеспечить безопасное взаимодействие между ресурсами с разных доменов. В данном проекте CORSMiddleware используется для управления заголовками и обеспечения корректного междоменного взаимодействия.

Pydantic — это библиотека для валидации, сериализации и десериализации данных в Python. Она предлагает удобный и типобезопасный способ обработки данных в приложении. В данном проекте Pydantic используется FastAPI для валидации входящих запросов, преобразования данных в объекты Python и автоматического создания схем данных для документации API.

SQLAlchemy — это мощная библиотека объектно-реляционного сопоставления (ORM — Object-Relational Mapping), позволяющая работать с реляционными базами данных в Python. Она предоставляет удобный и гибкий интерфейс для взаимодействия с базой данных, абстрагируя многие детали низкого уровня. В данном проекте SQLAlchemy используется для управления моделями данных, выполнения запросов к базе данных и обеспечения эффективного доступа к данным.

PostgreSQL - мощная реляционной системой управления базами данных (РСУБД). Он известен своей надежностью, гибкостью и расширенными функциями. В данном проекте PostgreSQL используется в качестве основной базы данных для хранения и управления данными. Он обеспечивает надежное и эффективное хранение данных, предоставляя богатый набор функций для обработки запросов, индексации и обеспечения целостности данных.

В данном проекте Uvicorn, FastAPI, CORSMiddleware, Pydantic и SQLAlchemy работают вместе, обеспечивая высокую производительность, безопасность и удобство разработки. Uvicorn обслуживает входящие запросы, FastAPI обрабатывает маршруты и валидацию, CORSMiddleware управляет междоменным взаимодействием, Pydantic обеспечивает валидацию и преобразование данных, а SQLAlchemy предоставляет доступ к базе данных и управление моделями. Интеграция этих инструментов позволяет создавать надежные и масштабируемые веб-API с эффективным управлением данными.

React — это популярная библиотека JavaScript для создания пользовательских интерфейсов. Он позволяет разработчикам создавать интерактивные и динамичные веб-приложения, используя компонентную архитектуру. React основан на концепции виртуального DOM, что обеспечивает эффективное обновление интерфейса. В данном проекте React используется в качестве основного инструмента для создания пользовательского интерфейса, управления состоянием и обработки пользовательских взаимодействий.

Redux — технология управления состоянием, используемая в данном проекте. Она обеспечивает централизованное и предсказуемое управление состоянием приложения. Redux основан на паттерне «однонаправленный поток данных», где действия инициируют изменения, а редюсеры обновляют состояние. В данном проекте Redux используется для управления состоянием, связанным с данными компонент (в особенности данных о товарах и их отзывах). Он обеспечивает предсказуемое обновление состояния, упрощает отладку и тестирование, а также способствует разделению логики управления состоянием и компонентов пользовательского интерфейса.

Tailwind CSS — это утилитарный фреймворк для CSS, который предоставляет готовые классы для быстрого создания стильных и

отзывчивых дизайнов. Он предлагает обширную библиотеку утилит, позволяя разработчикам создавать настраиваемые и адаптивные интерфейсы без написания большого количества кастомного CSS-кода. Tailwind CSS обеспечивает эффективность и последовательность в дизайне, позволяя разработчикам сосредоточиться на создании уникального внешнего вида и ощущений приложения. В данном проекте Tailwind CSS используется для стилизации компонентов React, обеспечивая привлекательный и отзывчивый дизайн.

В данном проекте React, Redux и Tailwind CSS работают вместе, дополняя друг друга. React обеспечивает структуру и интерактивность приложения, Redux управляет состоянием и обновлением компонент, их логикой, а Tailwind CSS упрощает процесс стилизации, позволяя разработчикам быстро создавать привлекательные и отзывчивые интерфейсы. Интеграция этих инструментов позволяет разработчикам создавать мощные, масштабируемые и стильные веб-приложения с улучшенным опытом пользователя.

3. Описание реализации решения

Веб приложение интернет-магазина реализовано с использованием клиент-серверной архитектуры, где фронтенд часть написана на React, а бэкенд реализован на FastAPI.

В фронтенд части приложения используется React Router для маршрутизации между различными экранами, такими как домашняя страница, страница корзины, страница продукта и т. д. Также используется Redux для управления состоянием, включая информацию о товарах, корзине и заказах. Взаимодействие с бэкендом организовано через API веб приложения и запросов на получение данных о товарах, добавление товаров в корзину, оформление заказов и т. д.

Бэкенд, реализованная на FastAPI, обеспечивает обработку запросов от фронтенд-части и взаимодействие с базой данных. Он определяет маршруты API для получения данных и управления ими через GET, POST, PUT и DELETE запросы. Для взаимодействия с локальной реляционной базой данных PostgreSQL где хранятся все данные приложения: товары, пользователи, заказы и т. д., используется библиотека SQLAlchemy.

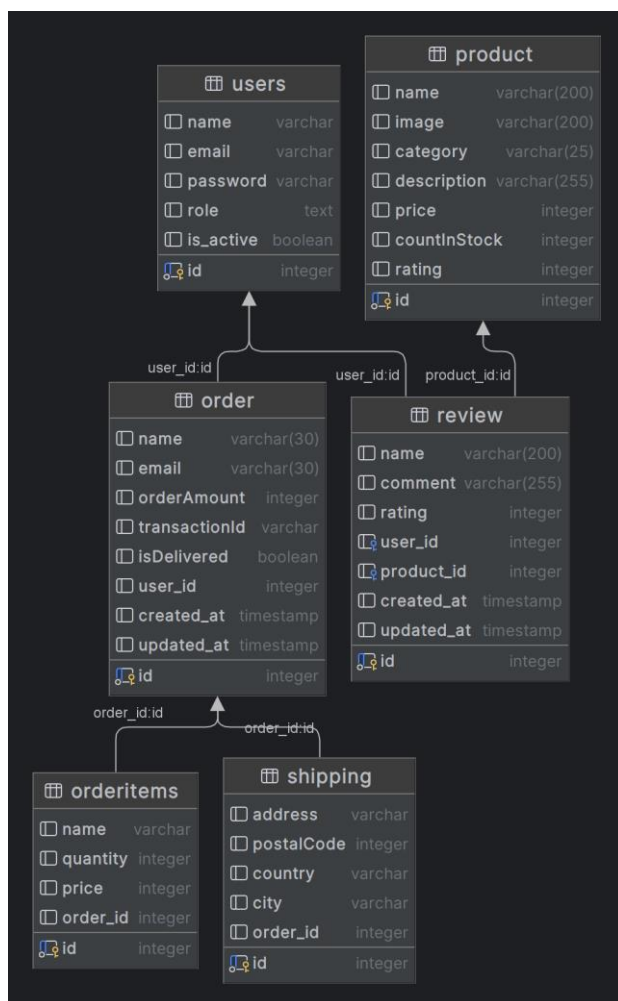


Рисунок 3.1. Реляционная модель структуры базы данных

Основные экраны веб приложения:

- **Экран регистрации (RegisterScreen.jsx):** Экран регистрации позволяет новым пользователям зарегистрироваться в приложении, создавая учетную запись с именем, электронной почтой и паролем.

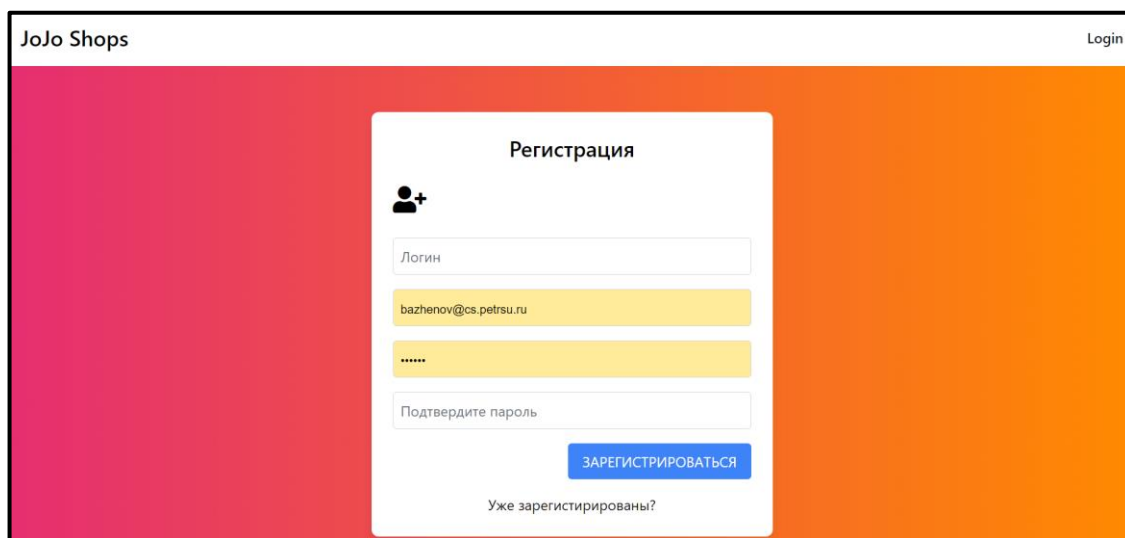


Рисунок 3.4. Страница регистрации пользователя

- **Экран входа в систему (LoginScreen.jsx):** Экран входа в систему позволяет пользователям входить в систему, используя свою электронную почту и пароль. Он также предоставляет возможность восстановления пароля.

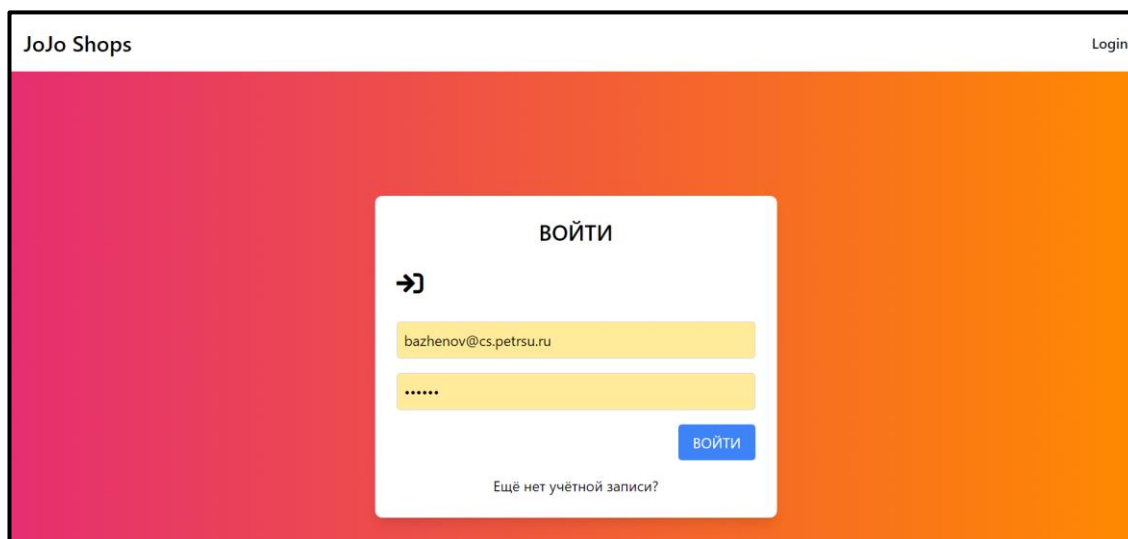


Рисунок 3.3. Страница авторизации пользователя

- **Домашняя страница (HomeScreen.jsx):** Домашняя страница является основным экраном приложения, где пользователи могут просматривать товары и добавлять товары в корзину. Она также содержит навигацию по другим экранам, таким как корзина, оформление заказа и профиль пользователя.

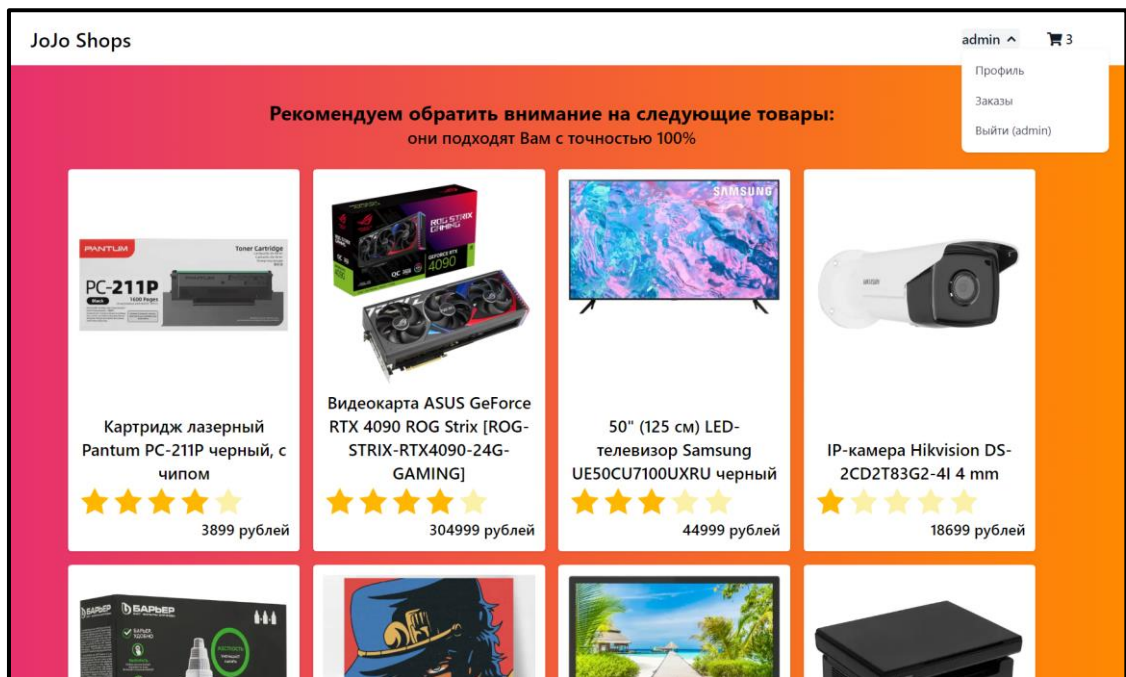


Рисунок 3.4. Основная страница веб приложения со всеми товарами интернет магазина

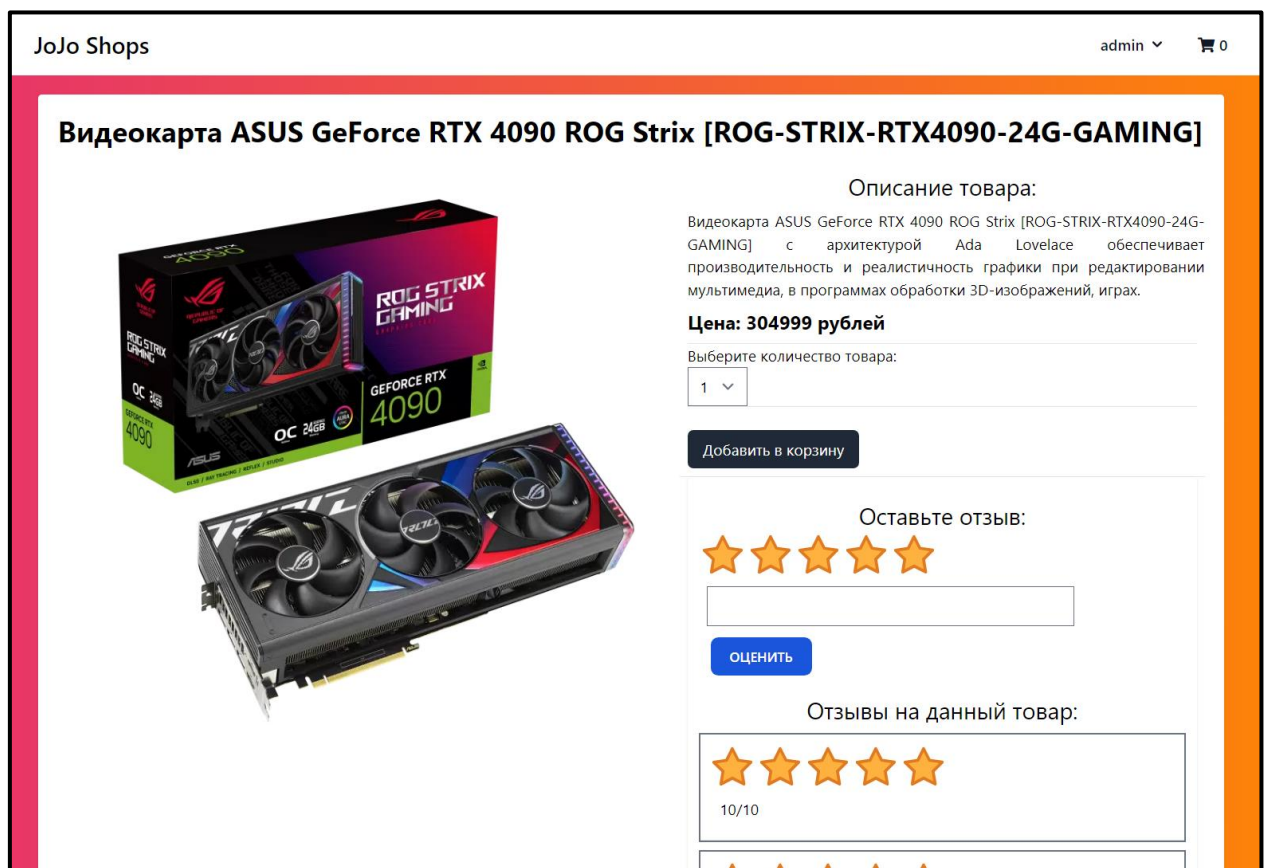


Рисунок 3.5. Страница информации о товаре

- **Корзина** (CartScreen.jsx): Экран корзины позволяет пользователям просматривать товары, добавленные в корзину, изменять их количество и удалять товары из корзины. Он также отображает общую сумму заказа и позволяет пользователям переходить к оформлению заказа.

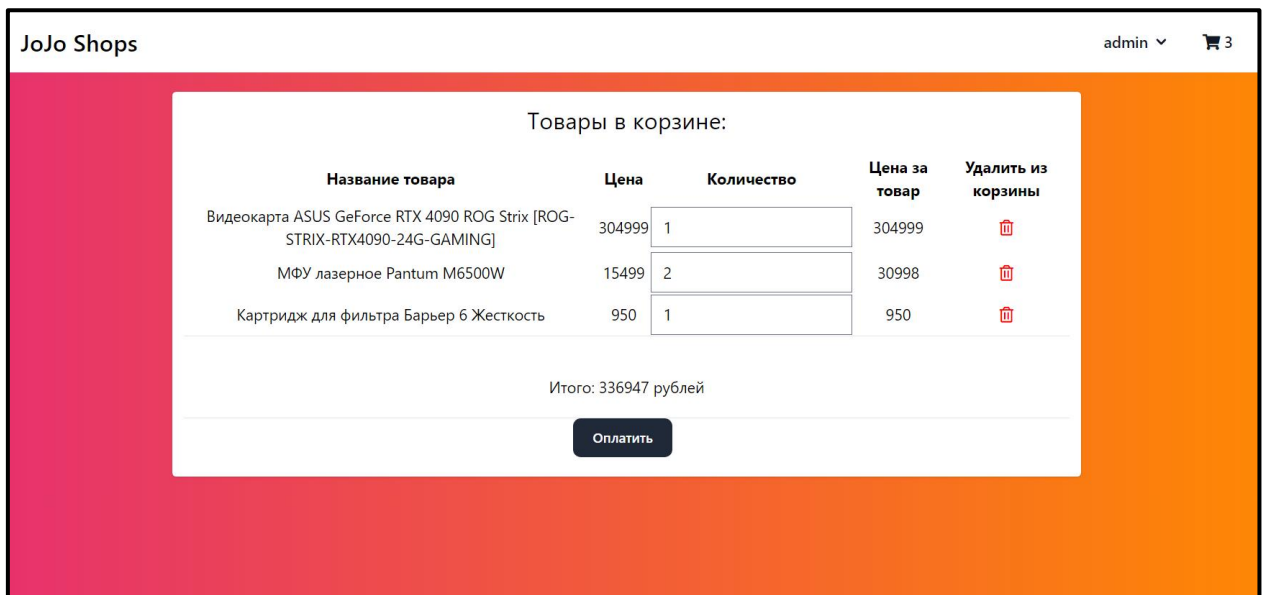


Рисунок 3.6. Страница выбранных пользователем товаров (корзина)

- **Оформление заказа (OrderScreen.jsx):** Экран оформления заказа позволяет пользователям проверить выбранные товары для оплаты и подтвердить заказ. Для оплаты товаров используется встроенная компонента `react-stripe-checkout`, проверяющая валидность почты пользователя и его карты.

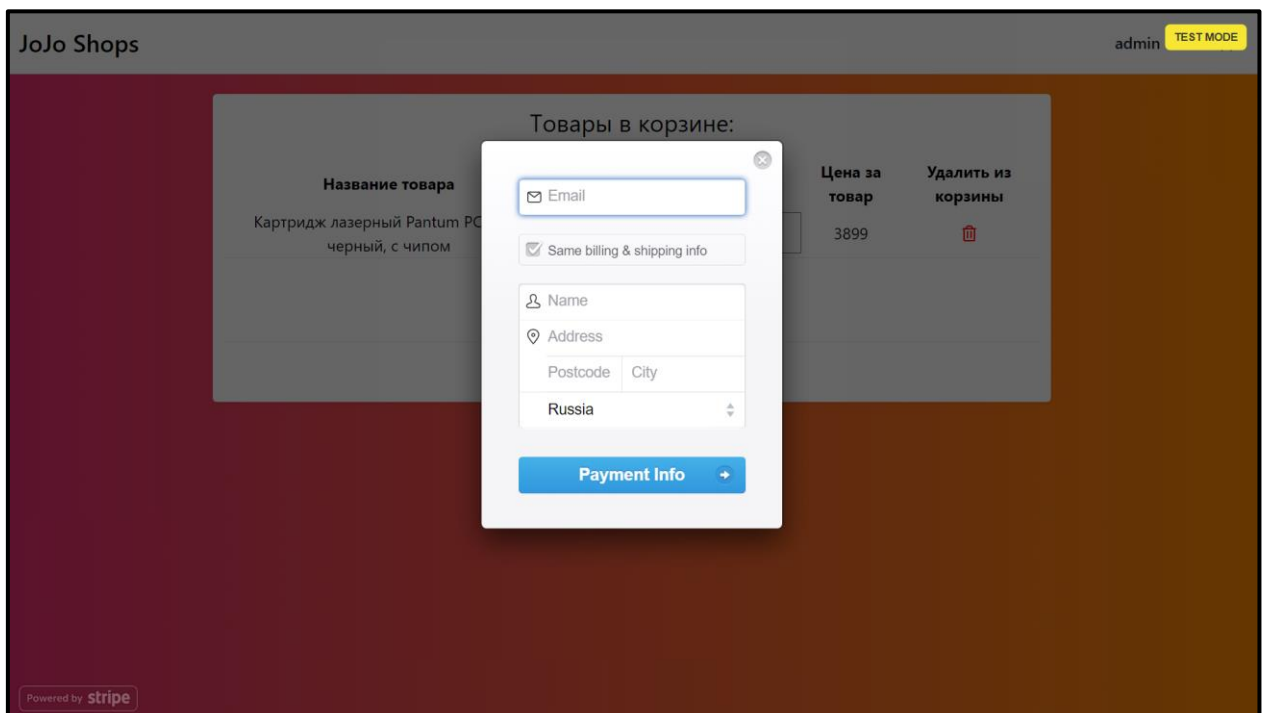


Рисунок 3.7. Страница оплаты выбранных товаров в корзине

- **Информация о заказе (OrderInfo.jsx):** Экран информации о заказе отображает подробную информацию о конкретном заказе, включая список товаров в заказе, общую сумму, информацию о доставке и статус заказа. Пользователи могут отслеживать статус доставки и получать подробную информацию о каждом товаре в заказе.

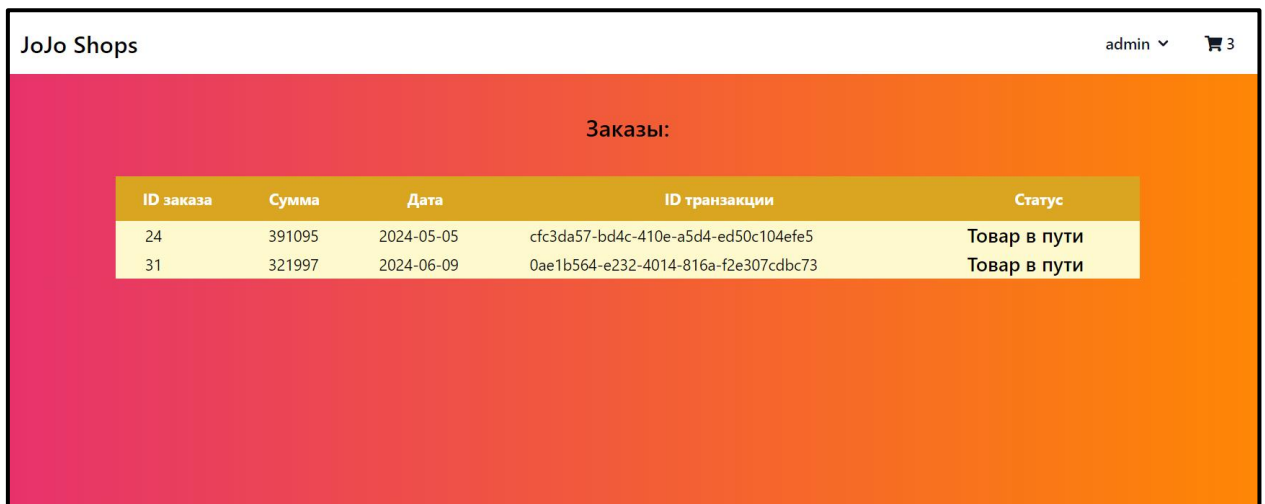


Рисунок 3.8. Страница заказов пользователя

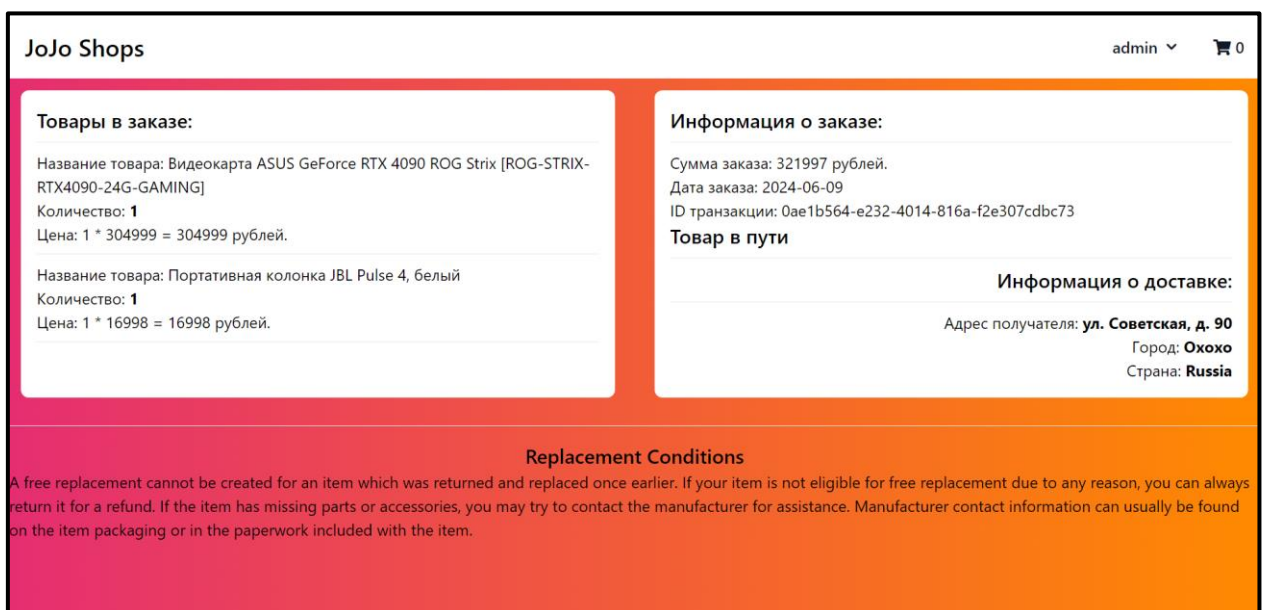


Рисунок 3.9. Страница информации о заказе (отображается при нажатии на строку с заказом)

- **Профиль пользователя (ProfileScreen.jsx):** Экран профиля пользователя позволяет пользователям просматривать и редактировать свою личную информацию, такую как имя, электронную почту, пароль и роль.

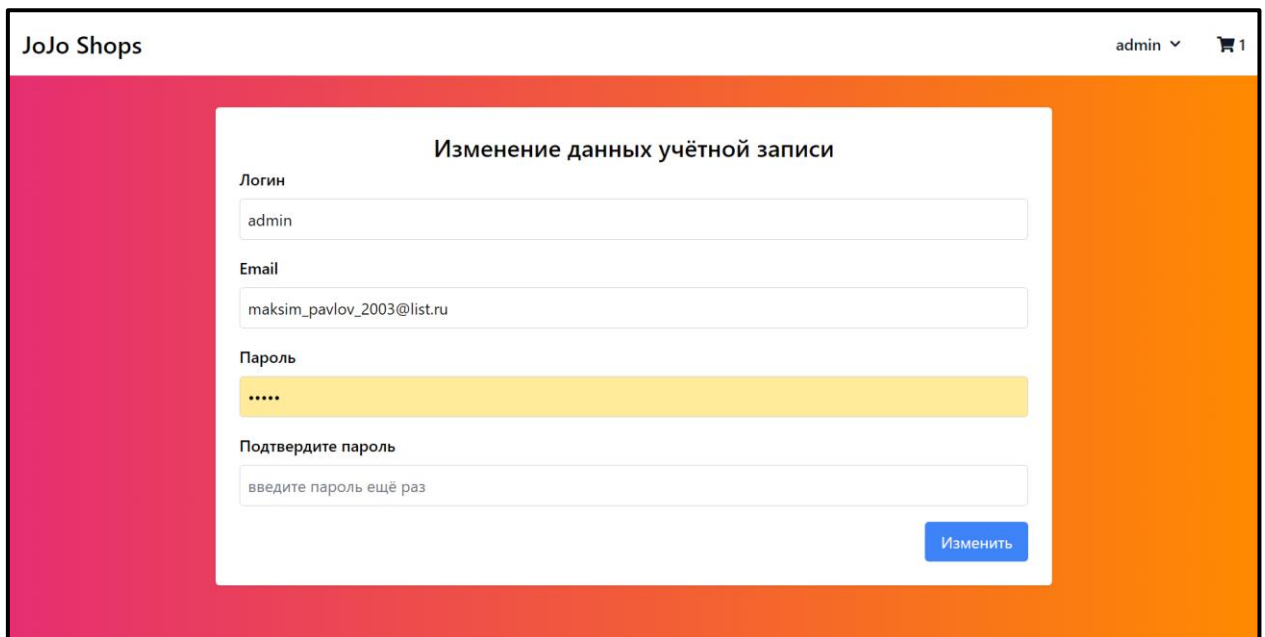


Рисунок 3.10. Страница профиля пользователя

- **Панель администратора (AdminScreen.jsx):** Панель администратора доступна только для пользователей с ролью «администратор» или «менеджер». Она предоставляет доступ к управлению пользователями, товарами, заказами и позволяет добавлять новые товары. Панель администратора включает в себя список пользователей, список товаров, список заказов и другие функции управления.

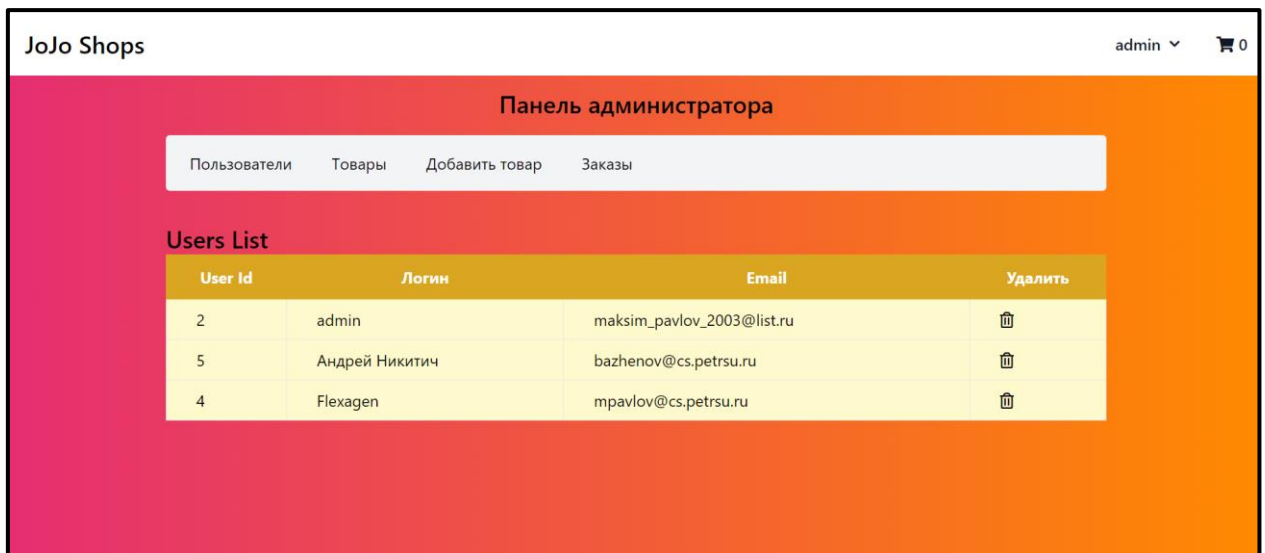


Рисунок 3.11. Страница управления учетными записями пользователей панели администратора

- **Экран списка товаров (ProductList.jsx):** Экран списка товаров доступен для пользователей с ролью «администратор» или «менеджер» и позволяет просматривать список всех товаров, добавленных в магазин. Он включает в себя информацию о каждом товаре, такую как название, цену, количество в наличии и оценку.

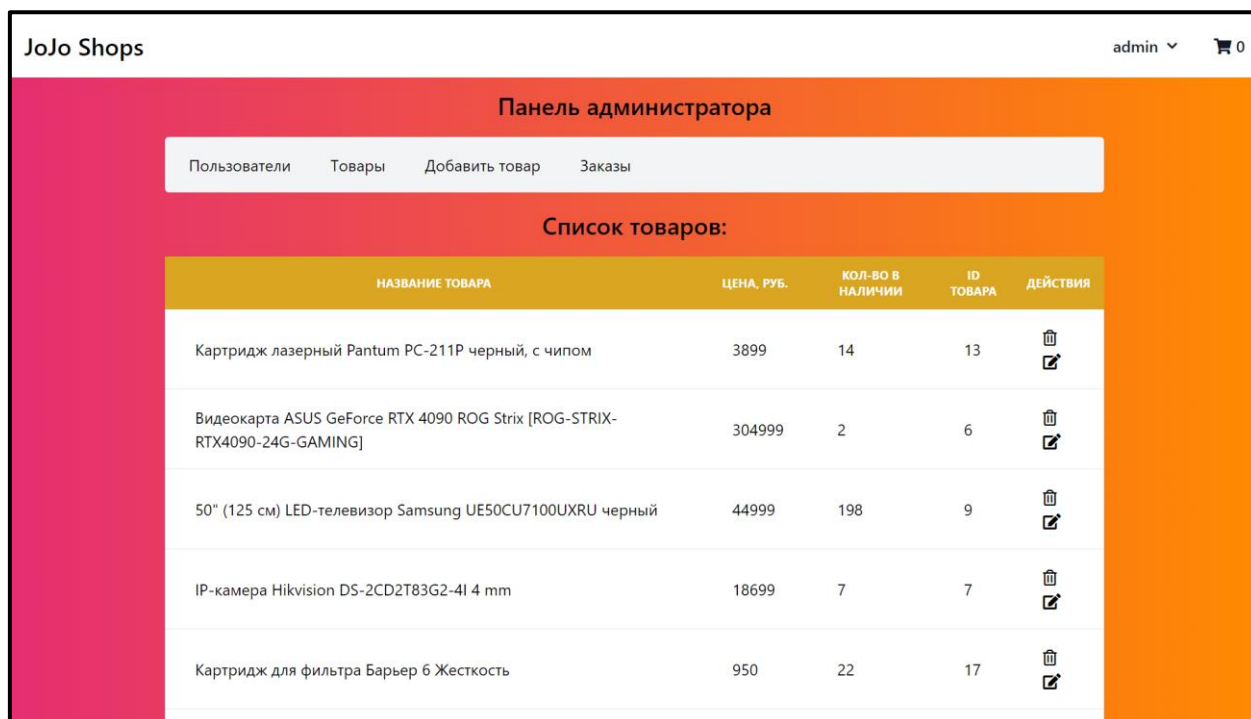


Рисунок 3.12. Страница управления товарами интернет магазина панели администратора

- **Экран редактирования товара (EditProduct.jsx):** Экран редактирования товара доступен для пользователей с ролью «администратор» или «менеджер». и позволяет редактировать информацию о существующих товарах, включая название, описание, цену, количество в наличии и категорию.

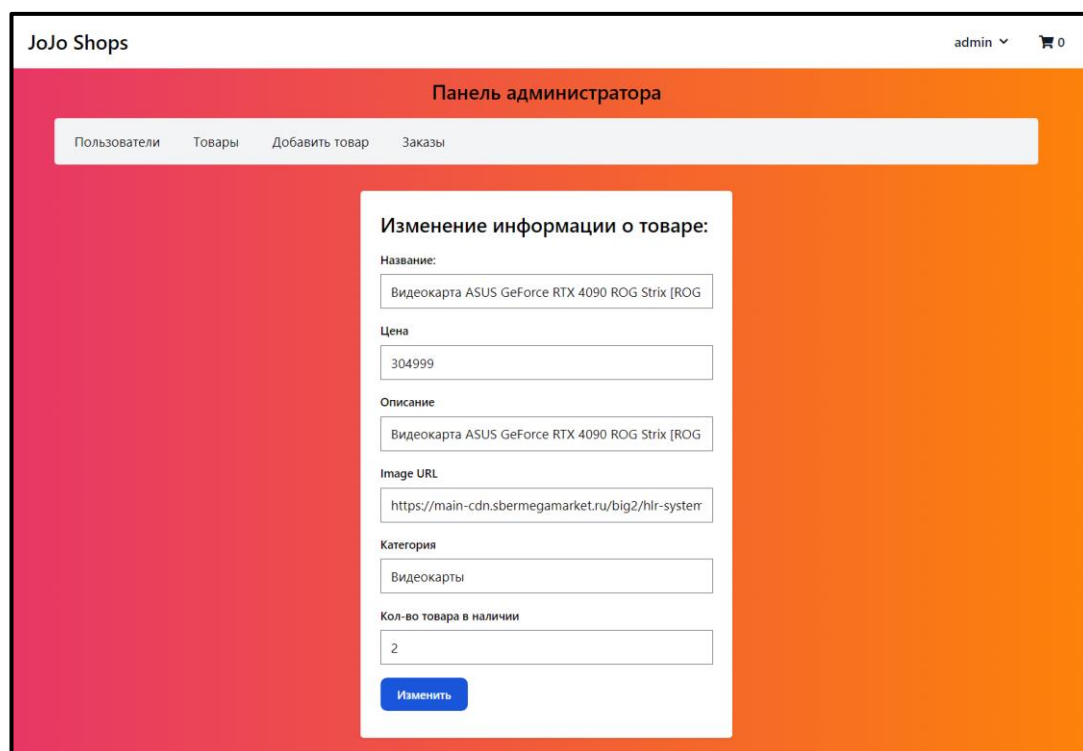


Рисунок 3.13. Страница редактирования информации о товаре интернет магазина в панели администратора

- **Экран добавления товара (AddProduct.jsx):** Экран добавления товара доступен только для пользователей с ролью «администратор» или «менеджер». Он позволяет добавлять новые товары в магазин, включая информацию о товаре, такую как название, описание, цену, количество в наличии и категорию.

Рисунок 3.14. Страница создания товара

- **Экран списка заказов (Orderslist.jsx):** Экран списка заказов доступен для пользователей с ролью «администратор» или «менеджер» и позволяет просматривать список всех заказов, совершенных в магазине. Он включает в себя информацию о каждом заказе, такую как ID заказа, сумму, дату и статус.

Id	Email	User Id	Цена	Дата	Transaction Id
24	maksim_pavlov_2003@list.ru	2	391095	2024-05-05T11:35:48.804294	cfc3da57-bd4c-410e-a5d4-ed50c104efe5
25	mpavlov@cs.petrus.ru	4	62296	2024-05-05T12:22:55.009728	9ec0f4f8-daf3-42db-bb11-5047c346ae98
27	bazhenov@cs.petrus.ru	5	18699	2024-05-06T12:35:59.216620	173f15bd-93da-441a-a89d-68e7767555bd
31	maksim_pavlov_2003@list.ru	2	321997	2024-06-09T07:19:52.075920	0ae1b564-e232-4014-816a-f2e307cdabc73

Рисунок 3.15. Страница просмотра всех заказов интернет магазина

В приложении также реализована система ролей пользователей, которая определяет доступ к определенным экранам и функциям. Пользователи с ролью «администратор» имеют полный доступ к панели администратора и могут управлять пользователями, товарами, заказами и добавлять новые товары. Пользователи с ролью «менеджер» имеют ограниченный доступ к панели администратора и не могут управлять пользователями. Обычные пользователи могут просматривать товары, добавлять их в корзину, оформлять заказы и просматривать историю своих заказов.

Заключение

В ходе практики были успешно решены все поставленные задачи. В результате прохождения практики были достигнуты следующие результаты:

- изучена документация по программным каскадам FastAPI и React;
- развернута база данных проекта средствами СУБД PostgreSQL;
- автоматическая инициализация и работа с базой данных с помощью SQLAlchemy;
- реализована, настроена и защищена серверная часть веб приложения средствами программного каскада FastAPI;
- реализована клиентская часть веб приложения с помощью программного каскада React;

Одной из особенностей разработанного веб приложения является использование безопасной авторизации и верификации пользователей. Используются JWT токены, пароли хранятся в базе данных в зашифрованном виде и используются только на серверной части приложения (никуда не передаются). Также одной из особенностей является алгоритм рекомендации опросов пользователям. На серверной части веб приложения реализован алгоритм рекомендации самых релевантных товаров – товаров с самым большим рейтингом и наименьшей ценой. Помимо этого, веб приложение полностью направлено на удобство пользователей при работе с магазином: каждый пользователь может покупать товары, оставлять на них отзывы, добавлять выбранные товары в корзину, а также покупать выбранные товары в корзине.

Таблица 1. Метрики кода проекта

Язык программирования	Количество строк кода	Количество строк с комментариями	Общее количество строк
Python	738 (75 %)	12 (1 %)	985
CSS	36 (47 %)	32 (42 %)	76
JavaScript	2413 (93 %)	0 (0 %)	2596

Репозиторий проекта на GitHub: <https://github.com/GhosT-FlexAgen/jojo-shop>