

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

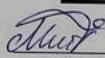
Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

**БИЗНЕС-ПРОЕКТ. ЭЛЕКТРОННЫЙ РЫНОК
ФЕРМЕРСКОЙ ПРОДУКЦИИ. КЛИЕНТСКАЯ
ЧАСТЬ**

Студент



(подпись, дата)

28.05.2025

Митрошин Александр Дмитриевич

(ФИО полностью)

Руководитель



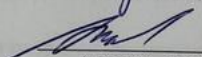
(подпись, дата)

28.05.2025

Карамышева Н.С.

(фамилия, инициалы)

Нормоконтролёр



(подпись, дата)

28.05.2025

Бычков А.С.

(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 28.05.2025г. № 16)

Заведующий кафедрой



(подпись)

М.А. Митрохин

(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

6 июня 2025г. № 10)

Секретарь ГЭК



(подпись)

Е.И. Гурин

(фамилия, инициалы)

Пенза 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 6 » мая 2025г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

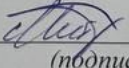
1. Студент Митрошин Александр Дмитриевич гр. 21ВВП1 факультета вычислительной техники
2. Тема работы Бизнес-проект. Электронной рынок фермерской продукции. Клиентская часть
Тема утверждена приказом ПГУ №601/о от "06" мая 2025г.
3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ ПГУ Карамышева Н.С.
4. Задание на работу (назначение разработки, исходные данные и т.п.)
Разработать клиентскую часть веб-сайта. Составить бизнес-модель и финансовый план.
Язык программирования JavaScript, среда разработки Visual studio code. Программа должна обладать интуитивно понятным интерфейсом и использовать следующие технологии:
1. Язык программирования JavaScript
5. Перечень подлежащих разработке вопросов
Требуется произвести анализ предметной области: изучить существующие аналоги, изучить способы проектирования и разработки веб-платформы. Выбрать язык программирования и среду разработки.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	4%	06.05.2025	
2	Анализ требований	8%	07.05.2025- 08.05.2025	
3	Выбор технологий, применяемых при проектировании и разработке	8%	09.05.2025- 10.05.2025	
4	Проектирование архитектуры	12%	11.05.2025- 13.05.2025	
5	Проектирование и Разработка ПО	44%	14.05.2025- 24.05.2025	
6	Тестирование и отладка	12%	25.05.2025- 27.05.2025	
7	Оформление выпускной работы	12%	28.05.2025- 30.05.2025	

Дата выдачи задания "06" мая 2025 г.

Руководитель бакалаврской работы  Карамышева Н.С.
(подпись, дата) 06.05.2025

Задание к исполнению принял студент  06.05.2025 Митрошин А.Д.
(подпись, дата)

Работу к защите допустить
Декан факультета  05.06.2025 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 98 листов, 24 рисунка, 1 таблицу, 6 источников, 3 приложения.

ВЕБ-ПЛАТФОРМА, ПОКУПАТЕЛИ, ПРОДАВЦЫ, ТЕСТИРОВАНИЕ, REACT.JS, MYSQL, VISUAL STUDIO CODE, REST API

Цель работы – разработать веб-платформу, объединяющую фермеров и покупателей для удобной и прозрачной торговли экологически чистыми продуктами.

Платформа должна позволять для покупателей осуществлять поиск ближайших фермеров через интерактивную карту, просматривать подробную информацию о товарах, оставлять отзывы и ставить рейтинги. Фермерам же нужно предоставить личный кабинет, где он сможет управлять ассортиментом, систему обратной связи.

В конечном итоге была разработана веб-платформа с использованием Python (Flask) для backend и React.js для frontend. Хранение информации о пользователях и товарах производилось в базе данных (MySQL).

					ПГУ1.090301.14.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Бизнес-проект. Электронный рынок фермерской продукции. Клиентская часть. Пояснительная записка	Лит.	Лист	Листов
Разраб.		Митрошин А.Д.		26.05.25				
Провер.		Карамышева Н.С.		26.05.25			2	98
Н. Контр		Бычков А.С.		26.05.25				
						ФВТ, каф. ВТ		
Утверд.		Митрохин М.А.		26.05.25		гр. 21ВВП1		

Содержание

Перечень сокращений и обозначений.....	5
Введение.....	6
1 Бизнес-проект. Электронный рынок фермерской продукции. Клиентская часть.....	7
1.1 Анализ предметной области	7
1.2 Анализ аналогов	9
1.2.1 Торговый рынок	9
1.2.2 АгроМП.....	9
1.2.3 Авито	10
1.2.4 Яндекс Маркет.....	11
1.2.5 Вывод.....	12
1.3 Постановка задачи.....	13
1.4 Требования к аппаратуре и программному обеспечению	14
2 Архитектура приложения.....	15
3 Выбор инструментов разработки	18
3.1 Выбор среды разработки	18
3.2 Выбор языка программирования.....	18
4 Бизнес-модель.....	21
4.1 Описание продукта	21
4.2 Потоки доходов	22
4.3 Каналы распространения.....	23
4.4 Взаимоотношения с клиентами	23
4.5 Ключевые ресурсы	24
4.6 Партнеры.....	24
4.7 Оценка затрат и доходов	25
4.8 Расчет срока окупаемости	26
5 Реализация приложения	27
5.1 Описание функционала	27

5.2 Описание работы приложения.....	30
5.3 Основные компоненты приложения	31
5.4 Описание интерфейса приложения	35
5.5 Тестирование приложения	40
5.5.1 Авторизация.....	41
5.5.2 Регистрация.....	42
5.5.3 Добавление новых товаров	43
5.5.4 Добавление нового адреса.....	44
6 Безопасность жизнедеятельности.....	45
6.1 Правила и техника безопасности при работе с ПК/ ноутбуком	45
6.2 Правила и техника безопасности при работе со смартфоном	46
Заключение	48
Список использованных источников	49
Приложение А	50
Приложение Б	56
Приложение В.....	88

Перечень сокращений и обозначений

В данной работе применяются следующие сокращения и обозначения

ОС – операционная система

ПК – персональный компьютер

ПО – программное обеспечение

API – (Application Programming Interface) – интерфейс программирования приложений, набор правил для взаимодействия программных компонентов

IDE – (Integrated Development Environment) – интегрированная среда разработки

REST – (Representational State Transfer) – архитектурный стиль взаимодействия между компонентами в распределённой среде, часто используется для создания

SEO – (Search Engine Optimization) – это комплекс мер по улучшению сайта для его ранжирования в поисковых системах

UI – (User Interface) – пользовательский интерфейс

UML – (Unified Modeling Language) – унифицированный язык моделирования

Введение

В современном мире цифровые технологии и маркетплейсы играют ключевую роль в модернизации традиционных бизнес-моделей, позволяя связать производителей и потребителей напрямую без лишних посредников. Особенно становится актуальным применение таких решений в сельскохозяйственном секторе, где спрос на свежую и натуральную фермерскую продукцию растет, а традиционные торговые площадки часто не справляются с вызовами современного рынка.

Проект «Электронный рынок фермерской продукции» направлен на создание инновационной цифровой платформы, которая объединяет фермеров и покупателей в одной экосистеме. Главная цель проекта заключается в предоставлении удобного онлайн-инструмента для просмотра ассортимента товаров, ознакомления с детальными характеристиками (фото, описание, цена). На этой платформе покупатели смогут легко находить нужную продукцию, а фермеры – расширять базу своих клиентов за пределами традиционных рынков.

Рост использования цифровых сервисов и стремление к поддержке локальных производителей являются основными мотивирующими факторами для разработки этого проекта. Внедрение технологии динамической подгрузки данных через API позволит обеспечить актуальность и оперативное обновление информации о товарах. А продуманная архитектура системы обеспечит ее масштабируемость. Интуитивно понятный интерфейс платформы создаст комфортные условия для выбора товара, делая этот процесс прозрачным и доступным.

Таким образом, цель данной дипломной работы – разработка функциональной и комфортной веб-платформы, которая станет эффективным инструментом для развития электронной коммерции в сельскохозяйственном секторе и поспособствует цифровизации с/х, улучшая взаимодействие между фермерами и покупателями.

					<i>ПГУ1.090301.14.001 ПЗ</i>	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

1 Бизнес-проект. Электронный рынок фермерской продукции. Клиентская часть

1.1 Анализ предметной области

«Электронный рынок фермерской продукции» - электронная платформа, которая обеспечивает прямое взаимодействие между потребителями и фермерами, упрощает процесс заказа и доставки свежих фермерских продуктов, поддерживает местных производителей и улучшает качество продуктов на рынке.

Развитие электронной торговли фермерскими продуктами в России набирает обороты, и идея создания платформы для прямого взаимодействия между производителями и потребителями отлично вписывается в этот тренд. Согласно данным, рынок онлайн-продаж продуктов питания в России стремительно растет и уже превысил 1 трлн рублей к концу 2024 года. При этом специализированные фермерские маркетплейсы становятся все более востребованными, а объем продаж фермерской продукции может достигнуть 80 млрд рублей в 2025 году.

Вместе с развитием онлайн продажи продуктов растет и спрос на фермерскую продукцию. Например, офлайн-продажи на рынках в июле 2023 года выросли на 28% по сравнению с аналогичным периодом 2022 года, а онлайн-заказы фермерских продуктов увеличились на 33%. Кроме того, объем онлайн-продаж фермерской продукции достиг 50 млрд рублей к 2024 году [1]. В ближайшие годы спрос будет смещаться в сторону свежего мяса, птицы, яиц и молока, поскольку потребители все больше ценят натуральные и качественные продукты.

Электронный рынок является стартапом, основная цель которого – решение проблемы доступности качественных продуктов. Это позволит покупателям приобретать свежие фермерские товары напрямую у

					ПГУ1.090301.14.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

производителей, минуя посредников. К тому же мы сможем поддерживать местных фермеров, так как платформа обеспечит им новые каналы сбыта и даст возможность расширения бизнеса. Ну и конечно нельзя забывать про прозрачность и доверие, которые так ценятся в современном мире. Пользователи смогут видеть рейтинги фермеров, отзывы и подтвержденное качество продукции.

С помощью нашей платформы пользователи смогут фильтровать запросы по регионам, категориям и производителям, тем самым облегчая поиск. Так же они смогут воспользоваться картой, на которой смогут найти ближайших производителей и быстрее купить нужный товар.

Рейтинговая система обеспечит прозрачность и доверие между покупателями и продавцами путем реализации отзывов от покупателей, рейтингов фермеров и карточек качества продукции производителей. Пользователи смогут оставлять комментарии после покупки, оценивая вкус и качество продукции. На карточках фермеры смогут добавлять свои сертификаты и лицензии, подтверждающие качество их продукции. А на основе количества успешных заказов, позитивных отзывов и соответствий стандартам качества у продавцов будет формироваться рейтинг.

Платформа должна поддерживать фермеров в продвижении их продукции путем рекламных кампаний (предоставление создания акций и специальных предложений), персонализированных рекомендаций покупателям, то есть им будут предлагаться продукты, основанные на их прошлых заказах и внедрения программы лояльности (система скидок и бонусов) для постоянных клиентов.

Таким образом, электронный рынок объединит в себе функции маркетплейса, рекламной площадки и инструмента продвижения для фермеров и сельских хозяйств. Оно поможет покупателям облегчить поиск свежих фермерских продуктов, а фермерам поможет продвигать и развивать своё дело.

1.2 Анализ аналогов

1.2.1 Торговый рынок

Торговый рынок – это классическая форма организации продажи товаров, где фермеры, продавцы и покупатели взаимодействуют напрямую. Он обеспечивает доступ к свежим продуктам и личное общение между продавцом и клиентом. В ходе анализа были выявлены следующие преимущества: живое общение, гибкость цен, возможность сразу оценить свежесть и внешний вид продукта перед покупкой.

К недостаткам можно отнести: ограниченная доступность, сезонность некоторых продуктов, ограниченный выбор. На рисунке 1 представлен типичный торговый рынок, который можно встретить в любом городе.



Рисунок 1 – Торговый рынок

1.2.2 АгроМП

АгроМП – онлайн площадка для продажи и покупки агропродукции. Сервис для фермеров, владельцев магазинов любого размера, центров питания и производителей продуктов питания. К плюсам можно отнести: помимо

продуктов продается с/х оборудование и техника, а также семена, возможно создание запроса на товары, которых нет, есть дополнительные статьи по теме с/х и посевной календарь.

К явным минусам относится: нет доставки товаров, платные тарифы для фермеров, нет карты с метками фермеров поблизости, отсутствует просмотр истории заказов, отсутствует просмотр истории заказов, больше ориентировано на оптовые заказы, нет возможности приобрести скоропортящиеся товары. На рисунке 2 представлена главная страница портала.

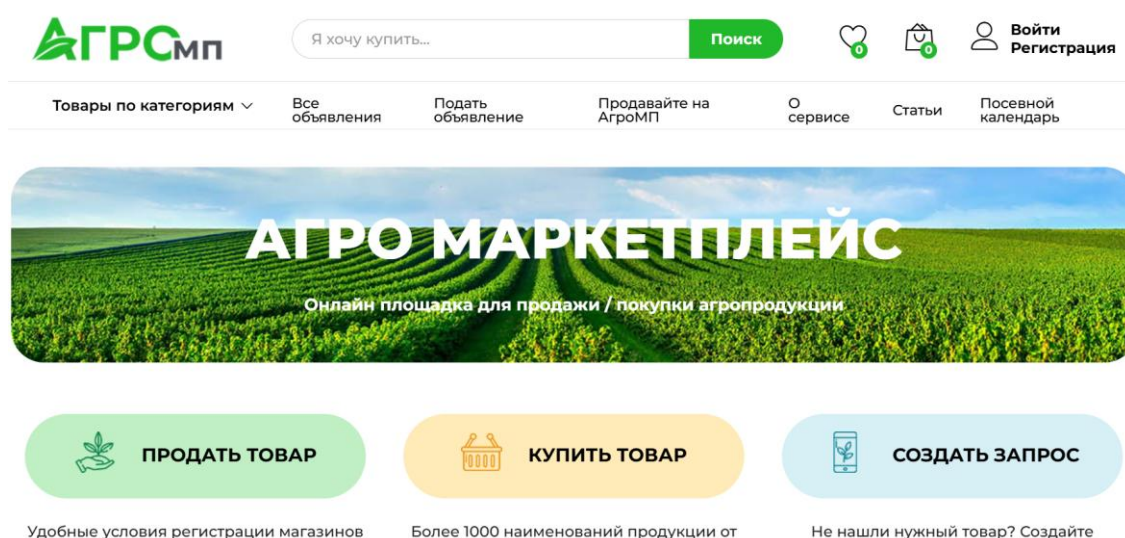


Рисунок 2 – Главная страница АгроМП

1.2.3 Авито

Авито — российский интернет-сервис для размещения объявлений о товарах, недвижимости, вакансиях и резюме на рынке труда, а также услугах, занимающий первое место в мире среди сайтов объявлений. Объявления могут размещать как частные лица, так и компании. К достоинствам можно отнести: возможность поиска товаров в своём городе, наличие чата для связи, рейтинг продавцов, наличие доставки.

При анализе платформы были выявлены следующие минусы: нет карты фермеров. На рисунке 3 представлена главная страница портала.

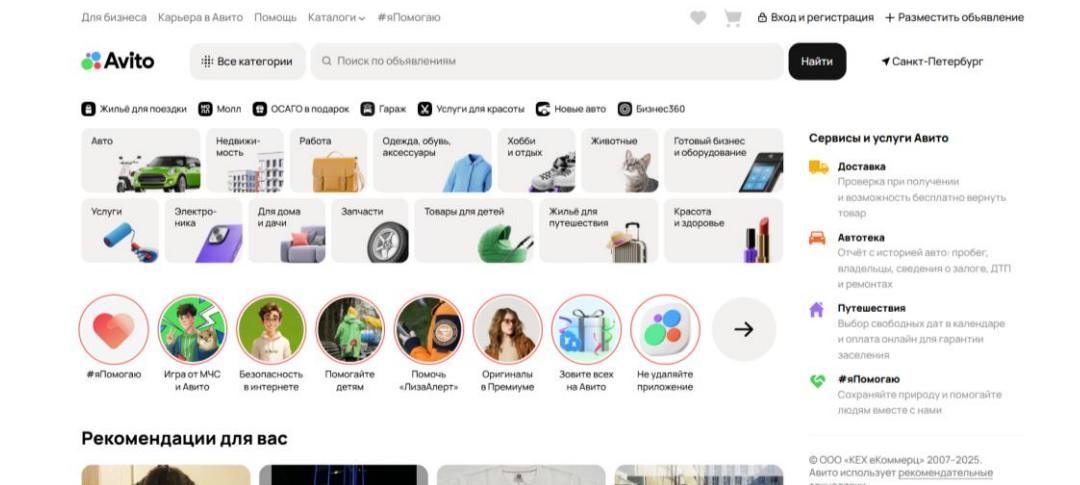


Рисунок 3 – Главная страница Авито

1.2.4 Яндекс Маркет

Яндекс Маркет – сервис компании Яндекс в сфере электронной коммерции. Состоит из маркетплейса Яндекс Маркет с сетью постаматов, складов и пунктов выдачи заказов (ПВЗ), и сервиса «Маркет для бизнеса». К очевидным плюсам относятся наличие рейтинга продавцов и доставки.

К недостаткам можно отнести то, что отсутствует карта фермеров, нет возможности приобрести многие товары. На рисунке 4 представлена главная страница портала.

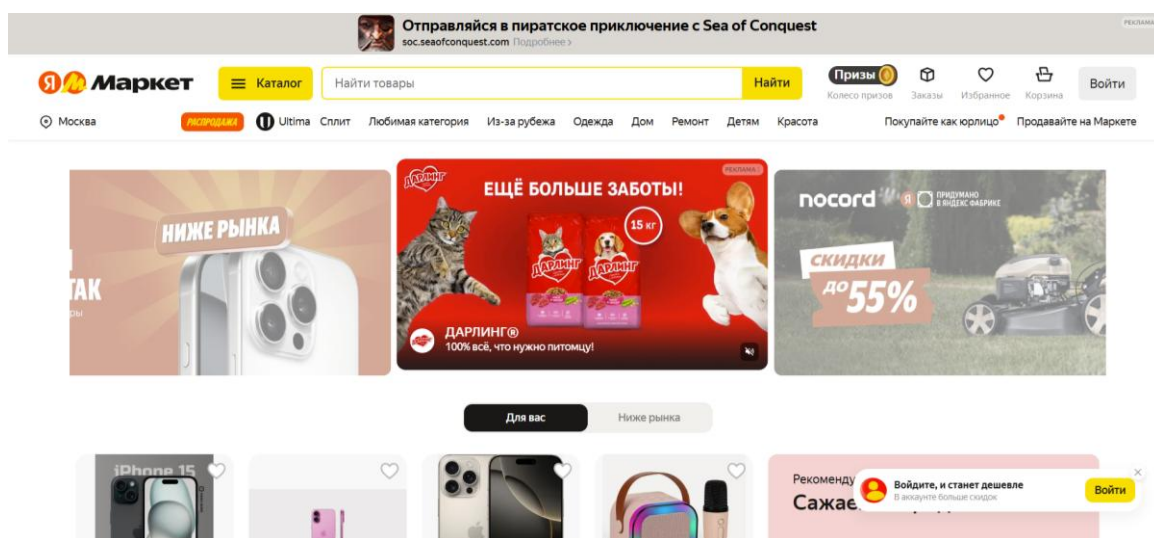


Рисунок 4 – Главная страница яндекс маркет

1.2.5 Вывод

В процессе исследования существующих аналогов было обнаружено, что ни один из них не обладает полным набором необходимых функций. В таблице 1 приведены основные критерии анализа аналогов.

Таблица 1 – Достоинства и недостатки аналогов

	Оффлайн рынок	АгроМП	Авито	Яндекс Маркет
Возможность покупки без доставки	+	-	+	-
Карта фермеров	-	-	-	-
Система отзывов и рейтингов	-	-	+	+
Возможность заказа скоропортящегося товара	+	-	+	-
Ориентация на локальный рынок	+	-	+	-
Ориентация на фермерские товары	+	+	-	-

Большинство решений являются универсальными, охватывая широкий ассортимент товаров и услуг, что снижает удобство поиска именно фермерской продукции и ослабляет доверие со стороны покупателей. Отсутствие специализированных функций (карты фермеров, отзывов о продукции)

препятствует формированию устойчивых каналов взаимодействия между покупателями и производителями.

Таким образом, актуальной задачей является создание специализированной платформы, сфокусированной исключительно на нише фермерских продуктов, обеспечивающей удобное взаимодействие, высокую прозрачность и доверие со стороны пользователей.

1.3 Постановка задачи

Для полноценной работы с платформой пользователям необходимо зарегистрироваться или войти в существующий аккаунт. Это обеспечивает доступ ко всему функционалу.

Платформа предоставляет возможность просматривать каталог фермерских продуктов. Для удобного поиска товаров пользователи могут применять фильтрацию, в том числе по категории товара, а также использовать строку поиска для нахождения товара или производителя по названию. Дополнительно реализована функция просмотра на карте ближайших производителей.

Регистрация и вход в систему доступны для двух типов аккаунтов: покупателя и производителя. После авторизации покупатели могут переходить в карточки товара и продавца, оставлять отзыв на конкретных продавцов. Производители могут загружать лого своего хозяйства и редактировать описание, и адрес(а) мест нахождения товара.

Производители имеют возможность добавлять свои товары на платформу. Для этого им необходимо ввести название товара, его описание, указать цену и загрузить фотографию товара. Ниже на рисунке 5 представлена диаграмма использования.

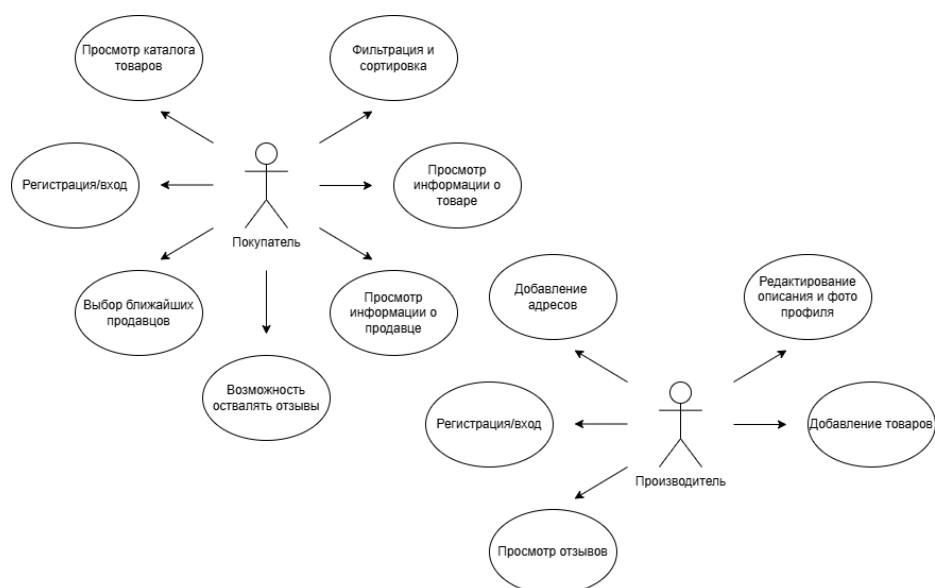


Рисунок 5 – UML-диаграмма вариантов использования

1.4 Требования к аппаратуре и программному обеспечению

Для полноценной работы клиентской части необходимы следующие минимальные требования к устройству:

- любое устройство, поддерживающее HTTP(S)-протокол — ПК, ноутбук, планшет, смартфон;
- современный браузер с поддержкой JavaScript, CSS3 и HTML5 (например, Google Chrome, Mozilla Firefox, Microsoft Edge);
- стабильное подключение к интернету;
- объём оперативной памяти от 2 ГБ (рекомендуется 4 ГБ и выше).

2 Архитектура приложения

Для разработки веб-приложения «Электронный рынок фермерской продукции» была выбрана клиент-серверная архитектура с чётким разделением на frontend и backend, взаимодействующих между собой через REST API. Такой подход обеспечивает масштабируемость, модульность, удобство сопровождения и даёт возможность разрабатывать клиентскую и серверную части независимо друг от друга.

На серверной стороне используется архитектурный паттерн Model-View-Controller (MVC), который структурирует код и способствует упрощённому управлению бизнес-логикой, пользовательским интерфейсом и взаимодействием с данными.

Клиентская часть реализована с использованием библиотеки React [2]. Она отвечает за отображение данных, взаимодействие с пользователем и отправку запросов к серверу. Интерфейс приложения построен на компонентной архитектуре. Основные разделы:

- каталог товаров;
- карточки продавцов;
- карточки товаров;
- личный кабинет (профили пользователей).

React-компоненты рендерят данные, полученные через REST API, и динамически обновляются при изменении состояния, что обеспечивает плавный пользовательский опыт [3].

Серверная часть приложения реализована с использованием Python и микрофреймворка Flask. Flask отвечает за маршрутизацию HTTP-запросов, выполнение бизнес-логики и взаимодействие с базой данных. API формирует и возвращает ответы в формате JSON, обеспечивая бесппроблемную интеграцию с frontend.

Контроллеры Flask принимают запросы от клиента, обрабатывают их, выполняют соответствующую логику, взаимодействуют с моделями и возвращают данные обратно. Они организуют маршруты API и обрабатывают доступ к данным в соответствии с правами пользователя.

Модели реализованы с использованием SQLAlchemy, выступающей в роли ORM. Это позволяет описывать структуру базы данных через Python-классы, избавляя от необходимости вручную писать SQL-запросы.

Ключевые сущности:

- покупатель (Buyer);
- продавец (Seller);
- адрес продавца (SellerAddress);
- товары (Product);
- отзывы (Review).

Особенностью архитектуры является система отзывов. Это повышает уровень доверия между участниками рынка.

Также планируется внедрение чата между продавцами и покупателями, что потребует реализации системы сокетов или отдельного API-узла для обработки сообщений в реальном времени.

Роль представления на клиентской стороне выполняют React-компоненты. Получая данные от API, они формируют интерфейс, отображая, например, карточки товаров, профили продавцов, отзывы и другую информацию. Таким образом, визуальная часть полностью изолирована от бизнес-логики и может развиваться независимо.

Аутентификация реализована на основе токенов доступа. После успешной авторизации пользователь получает токен, который прикрепляется к последующим запросам. Это позволяет контролировать доступ к определённым разделам API, реализовать проверку прав (например, запрет на редактирование чужих данных), а также защищать чувствительную информацию.

На серверной стороне реализована проверка прав и логика ограничения доступа, обеспечивающая безопасность операций. Например, продавец может редактировать только свои товары.

Всё взаимодействие между клиентом и сервером осуществляется через REST API, в котором реализованы CRUD-операции для всех ключевых сущностей. Формат обмена — JSON. API легко расширяем: добавление новых маршрутов и логики не требует пересмотра клиентской части, если структура данных остаётся прежней [4].

В качестве СУБД используется MySQL. Работа с базой осуществляется через SQLAlchemy, что упрощает миграции, валидацию и взаимодействие с таблицами. Важной особенностью является чёткая нормализация данных и наличие внешних ключей, обеспечивающих согласованность информации. Модели связаны друг с другом отношениями "один ко многим" и "многие ко многим", где это необходимо.

					ПГУ1.090301.14.001 ПЗ	Лист
						17
Изм.	Лист	№ докум.	Подпись	Дата		

3 Выбор инструментов разработки

3.1 Выбор среды разработки

Рассмотрим самую популярную и поддерживаемую IDE для разработки WEB приложений – Visual Studio Code – это легкая и при этом мощная IDE, созданная Microsoft, которая отлично подходит для разработки интерфейсов благодаря следующим особенностям:

- удобство редактирования кода. Поддерживается подсветка синтаксиса, автодополнение кода, интегрированная поддержка TypeScript;
- поддержка работы с файлами проекта. Можно легко отследить структуру файлов проекта через внутренний файловый менеджер;
- интеграция с системой контроля версий;
- всевозможные плагины и расширения;
- интеграция с терминалом;
- интуитивно понятный интерфейс.

Для создания веб-приложений можно использовать среды разработки вроде IntelliJ IDEA, Eclipse или Visual Studio. Однако их возможности ограничены при работе над пользовательским интерфейсом, поэтому выбор пал на Visual Studio Code.

3.2 Выбор языка программирования

Для создания современного, мощного и масштабного интерфейса важно выбрать подходящую технологию и инструменты, которые позволят обеспечить стабильную работу приложения. Для web разработки был выбран язык JavaScript. Рассмотрим три популярных технологии для разработки frontend:

- React;
- Vue.js;

– Angular.

React – библиотека для создания пользовательского интерфейса. Обладает следующими характеристиками:

– компонентный подход позволяет разделить интерфейс на разные, независимые друг от друга модули. Это упрощает разработку, тестирование и повторное использование кода;

– виртуальный DOM обеспечивает высокую производительность при каком-либо изменении состояния компонентов;

– обширная экосистема – богатое количество библиотек и инструментов для управления состоянием (например, Redux, MobX) и роутинг (React Router);

– большое сообщество разработчиков, множество обучающих материалов и сильная поддержка;

– однако, React дает только базовые возможности для UI, поэтому для реализации сложных проектов требуется интеграция других библиотек.

Vue.js – это современный фреймворк для разработки интерфейсов, который направлен на простоту и гибкость. Главные плюсы:

– легкость освоения – понятный синтаксис и достаточно дружелюбная документация;

– быстрая изменчивость данных – изменение данных автоматически обновляют интерфейс;

– обширный функционал встроенных инструментов для роутинга и управления состоянием;

– гибкость – дает возможность интегрировать Vue.js в уже существующий проект.

Vue.js подойдет только для небольших и средних проектов. Для больших приложений его функционала может быть недостаточно без дополнительных инструментов.

Angular – мощный фреймворк для разработки интерфейсов, который поддерживается Google. Основные особенности:

- объемный фреймворк с встроенными инструментами для работы с формами, роутингом, HTTP – запросами и управлением состоянием;
- двусторонняя привязка данных, которая упрощает синхронизацию между моделью и представлением;
- высокая производительность, которая позволяет работать со сложными приложениями с большим количеством данных;
- Angular основан на языке TypeScript, а это способствует поддержке и структурированности кода;
- Angular может быть излишним для небольших проектов и требует больше времени на обучение.

Проанализировав все варианты для проекта, было принято решение использовать технологию React, потому что с помощью его можно создать компактное, масштабируемое и производительное приложение, которое будет соответствовать всем потребностям проекта.

					ПГУ1.090301.14.001 ПЗ	Лист
						20
Изм.	Лист	№ докум.	Подпись	Дата		

4 Бизнес-модель

4.1 Описание продукта

Электронный рынок фермерской продукции – это веб-платформа, которая объединяет фермеров и покупателей, обеспечивает удобный поиск.

Целевая аудитория приложения – покупателя, предпочитающие свежие фермерские продукты. К ним можно отнести жителей города, которые придерживаются здорового питания, рестораны и кафе, ищущие локальных поставщиков, люди, желающие покупать экологически чистые продукты напрямую у производителей. Так же целевой аудиторий выступают фермеры и мелкие производства, которые хотят расширить сбыт:

- частные фермерские хозяйства;
- владельцы садов и огородов;
- производители органической продукции.

К основным функциям платформы можно отнести поиск фермеров и товаров при помощи интерактивной карты с геолокацией продавцов и фильтру по категории и названию товара или производителя, прозрачность сделок, которая будет осуществляться с помощью подробных карточек товара с описанием, персонализированный опыт в виде личного кабинета продавца, где он может управлять ассортиментом.

В данной платформе реализованы пользователи двух ролей – покупатели и продавцы. Для первых преимущество будет заключаться в доступе к свежим фермерским продуктам без посредников, в возможности проверить происхождение товаров через отзывы и рейтинги, в экономии времени за счет удобного поиска ближайшего производителя на карте. Для продавцов же очевидными преимуществами будут расширение клиентской базы без затрат на маркетинг и управлением продажами через удобный личный кабинет.

В процессе реализации был создан адаптивный интерфейс (frontend), созданный с помощью языка JavaScript и библиотеки React, а также серверная часть (backend), который был написан на языке Python (Flask). Общение между клиентской и серверной частью осуществлялось при помощи REST API. Хранение практически всей информации происходило в базе данных (MySQL). Так же во время разработки было обращение к дополнительным сервисам:

- react-leaflet (карта фермеров);
- cloudinary (облачное хранилище для упрощения передачи изображения).

Планом развития нашего проекта является создание чата, где покупатель и производитель смогли бы договориться о деталях заказа, внедрение системы подписок на фермерские продукты (регулярные заказы) будет очень актуально для кафе и ресторанов, интеграция с платежными системами, разработка мобильного приложения для iOS/Android.

Платформа устраняет ключевые проблемы рынка: недоверие покупателей, сложность поиска локальных продуктов и отсутствие удобных инструментов для фермеров.

Таким образом, «Электронный рынок фермерской продукции» — это не просто маркетплейс, а экосистема для честной и удобной торговли фермерскими товарами.

4.2 Потоки доходов

Платформа позволит получать доход за счет следующих функций:

- контекстная реклама. За счет показа сначала товаров от тех производителей, которые оплатили рекламу, эти фермеры смогут получать больше заказов, а значит и прибыли. А платформа сможет получать доход от партнеров за рекламу;
- система подписок. Благодаря платным подпискам покупатели смогут оформлять регулярные заказы свежих и натуральных продуктов, особый

интерес это вызовет у ресторанов и кафе. Так же подписку смогут оформлять и обычные покупатели, например, для доступа к редким товарам производителя;

– доставка. Взимание платы за доставку товаров от производителя к покупателю.

4.3 Каналы распространения

Для эффективного развития приложения недостаточно просто проводить его доработку и поддержку. Платформа требует активной рекламной кампании с целью привлечения новой аудитории.

Продвижение будет реализовываться посредством онлайн-маркетинга, включая инструменты SEO, контекстную рекламу и активное использование социальных сетей.

Дополнительно, привлечение новых пользователей будет способствовать реализации партнерских программ. Например, организация совместных акций с Министерством сельского хозяйства не только повысит уровень доверия фермеров к министерству, но и заинтересует новых производителей к участию в платформе.

4.4 Взаимоотношения с клиентами

Успешное развитие приложения подразумевает не только привлечение новых клиентов, но и поддержание крепких отношений с уже существующими пользователями. Для этого планируется создание активного сообщества, организация эффективной онлайн-поддержки и регулярное обновление контента. Формирование сообщества позволит участникам обмениваться опытом и оставаться в курсе акционных предложений, инициатив партнеров и последних нововведений платформы.

4.5 Ключевые ресурсы

Для разработки и поддержания работоспособности приложения нужны такие ресурсы как разработчики, тестировщики, контент, специалист по маркетингу.

На разработчиках лежит задача разработки архитектуры приложения, написания кода. Так как платформа должна работать качественно из-за большого масштаба, то разработчики не смогут уследить за всеми неточностям, поэтому с этими задачами будет справляться специалист по тестированию, который сможет своевременно протестировать платформу и дать точный отзыв разработчикам. Чтобы на нашу платформу приходило больше продавцов и покупателей, нужно развивать интерес в двух направлениях соответственно: продвигать и рассказывать о нашем проекте в различных фермерских объединениях и клубах для поиска фермеров и создание актуальной, интересной рекламы для привлечения покупателей. Для выполнения этой задачи нужен специалист по маркетингу и рекламе.

4.6 Партнеры

Для успешного развития платформы ключевыми партнерами могут стать:

- сельскохозяйственные кооперативы и объединения помогут привлекать на платформу новых производителей, могут учувствовать в сертификации продукции и обеспечат контроль качества товаров;
- эконо-магазины и кафе здорового питания смогут размещать рекламу и специальные предложения. Также можно создать партнерские программы лояльности и организацию совместных промоакций и дегустаций;
- логистические компании обеспечат доставку товаров покупателям. Могут предложить специальные тарифы для пользователей платформы;
- платежные системы и банки обеспечат безопасные транзакции;
- государственные и общественные организации.

					ПГУ1.090301.14.001 ПЗ	Лист
						24
Изм.	Лист	№ докум.	Подпись	Дата		

Каждый из партнеров получит выгоду от сотрудничества: фермеры – новые возможности сбыта, покупатели – гарантии качества и удобство покупок, бизнес-партнеры – доступ к целевой аудитории, государство – развитие локальных производителей.

4.7 Оценка затрат и доходов

Разработка и запуск:

- дизайн UX/UI и прототипирование: 100 000 руб.;
- разработка frontend: 130 000 руб.;
- разработка backend: 150 000 руб.;
- настройка и развертывание серверной инфраструктуры: 70 000 руб.;
- тестирование и отладка: 60 000 руб.

Итого первоначальные затраты: 510 000 руб.

Ежемесячные расходы:

- хостинг и обслуживание серверов: 30 000 руб./мес.;
- поддержка и обновление платформы: 60 000 руб./мес.;
- модерация контента и служба поддержки: 35 000 руб./мес.;
- маркетинг и продвижение: 75 000 руб./мес.

Итого ежемесячные затраты: 200 000 руб.

Прогнозируемые доходы (при 5 000 активных пользователях).

Комиссия с доставки.

Средний стоимость доставки ~ 200 руб.

Количество заказов в месяц: 5 000.

Комиссия платформы (30 руб. за каждую доставку): 150 000 руб./мес.

Подписка на фермерскую продукцию.

10% покупателей (750 пользователей) * 200 руб. = 150 000 руб./мес.

Контекстная реклама и партнерские программы.

Контекстная реклама: 10 000 руб./мес.

Партнерские программы с эко-магазинами: 10 000 руб./мес.

Итого ежемесячный доход: 320 000 руб.

4.8 Расчет срока окупаемости

Для определения периода окупаемости проекта учитываются как первоначальные инвестиции, так и ежемесячные затраты и доходы. Исходя из того, что ежемесячный доход составляет 320 000 руб., а расходы – 200 000 руб., чистая прибыль за месяц равна 120 000 руб.

Первоначальный расчет показывает, что при условии возврата инвестиций в 510 000 руб., срок окупаемости составит $510\,000 \text{ руб.} / 120\,000 \text{ руб.} \approx 5$ месяцев. Однако, допущения по окупаемости основываются на выполнении ряда условий: активная пользовательская база должна включать 5 000 активных покупателей, конверсия в премиум-подписку должна достигать 10% от общей аудитории, а средний доход от рекламы – 10 000 руб. на одного пользователя в месяц.

Учитывая эти параметры, окончательный срок окупаемости рассчитывается в размере примерно 5 месяцев.

5 Реализация приложения

5.1 Описание функционала

На платформе можно легко создать новый аккаунт или войти в уже существующий, и это открывает доступ к большему числу функций. Любой не авторизованный пользователь может просто посмотреть каталог, перечень категорий или воспользоваться интерактивной картой, но зарегистрировавшись, получает гораздо больше возможностей. При регистрации пользователю предлагается выбрать роль покупателя или производителя. Если выбирается роль покупателя, то для доступны следующие опции:

- просмотр каталога товаров от фермеров;
- отфильтровать каталог по той или иной категории;
- поиск ближайшего нужного производителя на интерактивной карте;
- просмотр подробной информации о каждом товаре отдельно в карточке товара;
- просмотр подробной информации о каждом производителе отдельно в карточке производителя;
- оставить отзыв и поставить рейтинг о производителе, а карточке производителя;
- удалить свой аккаунт.

После регистрации у производителя есть следующие возможности:

- редактировать лого производства;
- редактирование описания;
- посмотреть отзывы о своем производстве и увидеть свой рейтинг у покупателей;
- управлять адресами своего производства (создавать новые, удалять);
- создавать/удалять карточки своих товаров;
- удалять свой аккаунт.

Платформа позволяет покупателям и фермерам находить друг друга и без посредников покупать свежие фермерские товары. Так же больше не нужно никому ехать на рынок или базар в поиск качественных товаров, фермерам теперь достаточно добавить новый товар, загрузить адреса откуда можно забрать, а покупателям скролить каталог и писать продавцам напрямую. Это значительно упрощает жизнь всем.

При открытии платформы пользователь видит главную страницу для не авторизованных пользователей. На этой странице возможно просматривать каталог товаров, пользоваться интерактивной картой, пользоваться фильтром и поисковой строкой. Также можно авторизоваться, нажав на кнопку «Войти». На рисунке 6 представлена страница для не авторизованных пользователей.

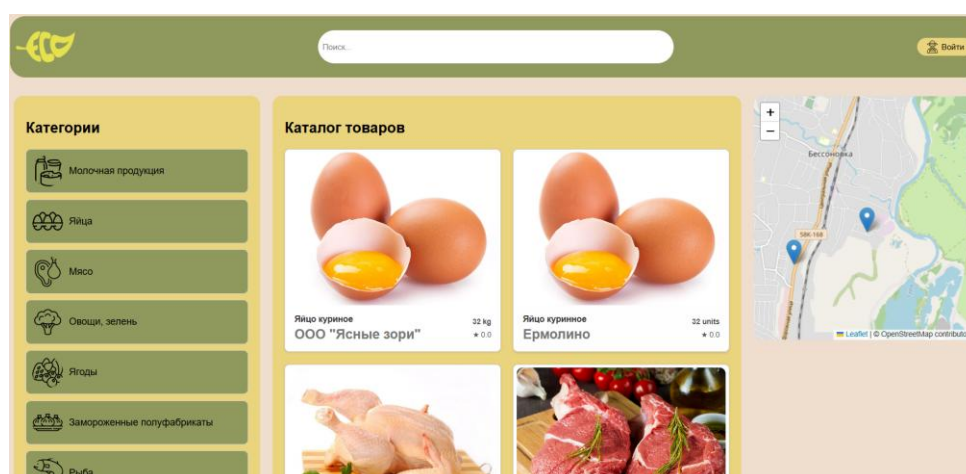


Рисунок 6 – Страница для не авторизованных пользователей

При нажатии на кнопку «Войти» пользователю предоставляется ввести email и пароль. В случае неверных введенных данных выйдет окно с ошибкой. Если у пользователя нет аккаунта, то предлагает зарегистрироваться, далее пользователь может выбрать зарегистрироваться ему как покупатель или производитель, от этого будет зависеть какие поля ему нужно будет заполнить. После успешной авторизации покупателя его перекидывает на страницу каталога, про который было упомянуто выше, только привязанного к конкретному пользователю, а после успешной авторизации производителя его

перекидывает на страницу аккаунта, в целом это единственная страница, которая ему будет доступна. На рисунке 7 и 8 представлены каталог и страница аккаунта продавца соответственно.

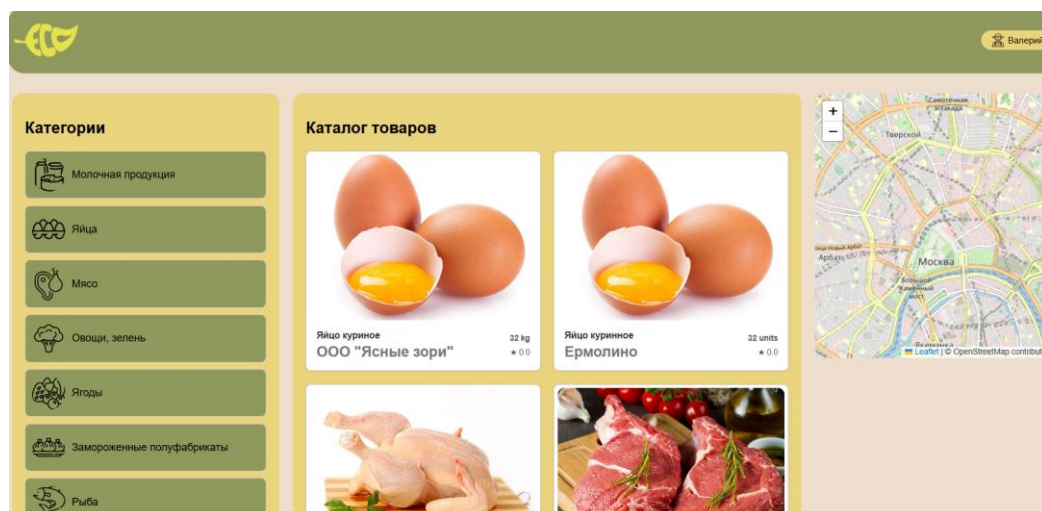


Рисунок 7 – Каталог

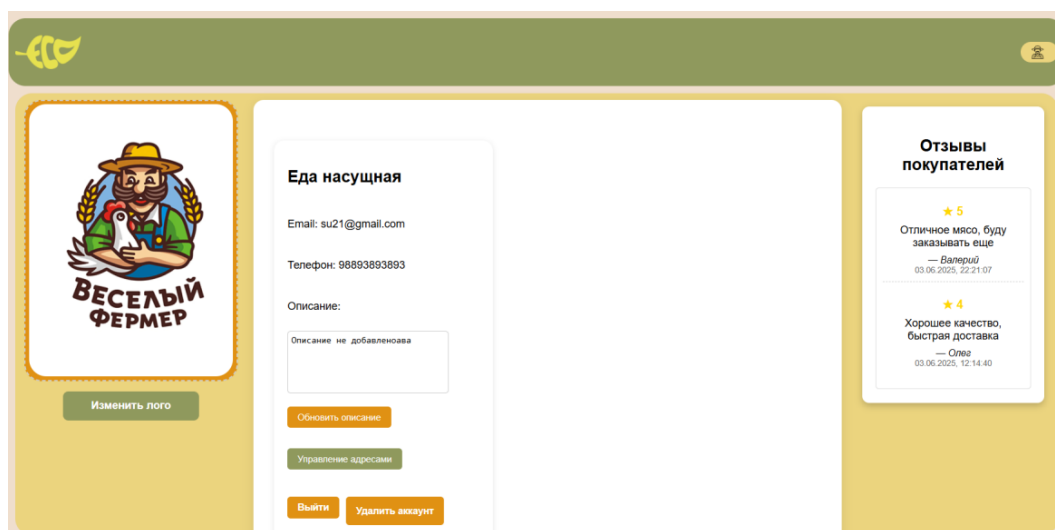


Рисунок 8 – Страница аккаунта продавца

После успешного входа покупатель может: просматривать каталог товаров, пользоваться интерактивной картой, пользоваться фильтром и поисковой строкой, открывать карточку товара или карточку производителя, оставлять отзыв производителю и поставить ему рейтинг, удалить свой аккаунт.

А производитель: редактировать лого производства, редактировать описание, посмотреть отзывы о своем производстве и увидеть свой рейтинг у покупателей, управлять адресами своего производства (создавать новые, удалять), создавать/удалять карточки своих товаров, удалять свой аккаунт.

Пользователи также могут выйти из аккаунта в любой момент времени.

5.2 Описание работы приложения

Приложение работает через клиентскую часть, с которой взаимодействует пользователь. Чтобы начать, достаточно открыть приложение в браузере и выполнить авторизацию или регистрацию. Затем клиент отправляет запрос на сервер через REST API, где введенные данные проверяются, и в случае успеха сервер возвращает подтверждение. После этого пользователь видит страницу, соответствующую его роли.

Все шаги, как взаимодействия пользователя с клиентом, так и общения клиента с сервером, отражены на диаграмме последовательности (см. рисунок 9).

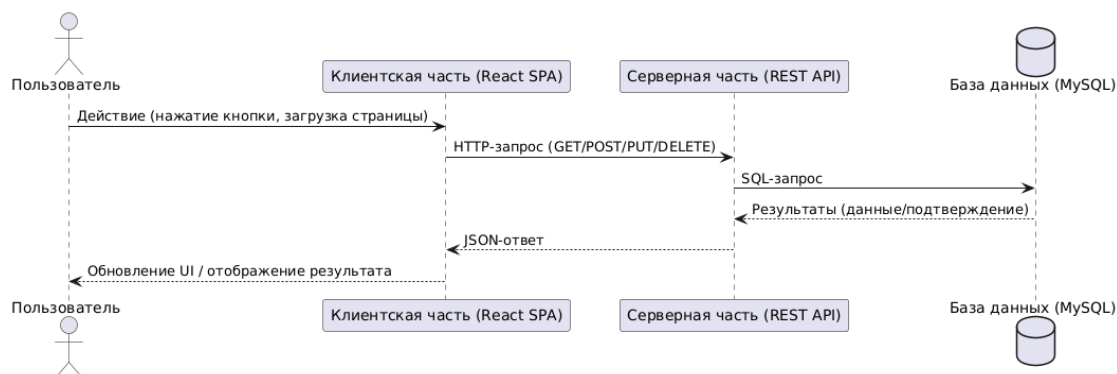


Рисунок 9 – UML-диаграмма последовательности

А расположение каждого из компонентов данной системы представлено на рисунке 10.

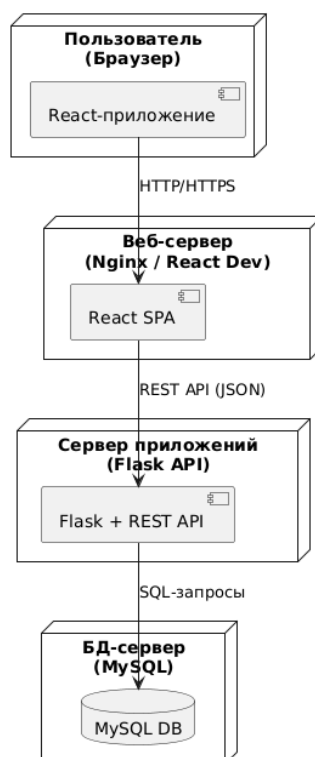


Рисунок 10 – UML диаграмма развертывания

5.3 Основные компоненты приложения

Платформа включает в себя следующие основные компоненты: LoginModal, RegisterModal, ProductModal, AddressModal, AccountSeller, Catalog, Main.

Компонент LoginModal отвечает за вход пользователей в систему. Он предоставляет два поля для ввода — email и пароль. После отправки формы компонент пытается авторизоваться сначала как покупатель (через API-метод loginBuyer), а в случае ошибки — как продавец (через loginSellers). При успешной авторизации полученный токен сохраняется в localStorage и пользователь перенаправляется на нужную страницу: для покупателя — каталог товаров, а для продавца — личный кабинет.

Ключевые моменты реализации:

- используются хуки useState для управления состоянием введенных данных (email и password);

- обработчик `handleSubmit` делает два последовательных вызова: сперва пытается авторизоваться как покупатель, а затем как продавец, если первый запрос возвращает ошибку;

- сохранение токена в `localStorage` происходит при успехе;

- после успешного входа происходит перенаправление через `useNavigate`.

Компонент `RegisterModal` предназначен для регистрации новых пользователей. Он поддерживает две формы – одну для покупателей, другую для продавцов (производителей). Пользователь может переключаться между регистрацией как `buyer` или `seller` с помощью кнопок. В зависимости от выбранного типа при отправке данных происходит вызов соответствующего API-эндпоинта (`registerBuyer` или `registerSellers`). После успешной регистрации окно закрывается, и происходит перенаправление на нужную страницу.

- Ключевые моменты реализации:

- состояние `regType` определяет тип регистрации (`buyer` или `seller`);

- форма заполняется в зависимости от типа: для продавца добавляется поле для наименования производства;

- при отправке формы вызывается соответствующий обработчик: `handleSubmitBuyer` или `handleSubmitSeller`;

- после успешного ответа API происходит сохранение пользователя (и возможное логин) и перенаправление.

Компонент `ProductModal` используется для добавления нового товара продавцом. Он реализует следующие функции:

- позволяет загрузить изображение товара через `Cloudinary`; при выборе файла происходит загрузка на `Cloudinary`, и полученный URL сохраняется;

- предоставляет форму с полями: категория, название, описание, цена, количество, единица измерения;

- при отправке формы данные товара отправляются на сервер, используя API-метод `createProduct`, после чего модальное окно закрывается.

Ключевые моменты реализации:

- для загрузки изображения используется библиотека axios, а само изображение отправляется с помощью FormData;
- после успешной загрузки изображения и получения URL, он используется для формирования объекта товара;
- перед отправкой проверяются обязательные поля.

Компонент AddressModal используется для управления адресами продавца. Он позволяет:

- отобразить список ранее сохранённых адресов с возможностью удаления;
- добавить новый адрес с помощью интерактивной карты (реализованной с использованием react-leaflet);
- при перетаскивании маркера с помощью компонента DraggableMarker происходит обратное геокодирование (через Nominatim), и в не редактируемое поле выводится полученный адрес.

Ключевые моменты реализации:

- используются компоненты MapContainer, TileLayer из библиотеки react-leaflet для отображения карты;
- компонент RecenterMap следит за изменением позиции и устанавливает новый центр карты;
- при перемещении маркера вызывается функция обратного геокодирования, которая получает строку адреса;
- форма для добавления нового адреса включает поле с временем работы;
- вместе с адресом отправляется запрос через API-метод addSellerAddress.

Компонент AccountSeller – это кабинет продавца, где отображаются данные о продавце, его товары, описание, и предоставляются возможности для обновления информации (например, описание компании) и управления товарами.

Ключевые моменты реализации:

– загрузка данных продавца. Вызывается метод `getSellerData`, после чего данные записываются в состояние `seller` и описание (`state description`) синхронизируется с данными продавца;

– управление товарными позициями. После загрузки данных продавца запускается запрос для получения его товаров с помощью `getProductsBySeller`. Полученные товары сохраняются в состоянии `products`. Каждый товар отображается как карточка с изображением, именем, ценой, количеством и кнопкой для удаления (через `deleteProduct`);

– редактирование описания и управление адресами. Для редактирования описания используется текстовое поле с кнопкой, которая вызывает API-метод `updateSellerDescription`. Также есть кнопка для открытия модального окна управления адресами (`AddressModal`);

– загрузка логотипа. Отдельный компонент `SellerLogoUpload` используется для загрузки логотипа и обновления информации на сервере.

Компоненты `Catalog` и `Main` являются страницами, предназначенными для отображения товаров:

– `catalog`. Отображает в трёх колонках – категории (слева), каталог товаров (в центре) и дополнительную информацию (справа, например, карту или иной контент). Здесь для каждого товара реализуются карточки с изображением, названием, продавцом, ценой и рейтингом;

– `main`. Представляет собой главную страницу, где также показываются категории и товары. Он схож с `Catalog` по структуре, но может иметь чуть другой дизайн или функциональность.

Ключевые моменты реализации:

– использование `layout` на `Flexbox` для создания колонок;

– получение данных из API методов (например, `getCategories` и `getNearbyProducts`);

– обработка клика по товару для перехода на страницу товара (с помощью функции `handleProductClick`).

На рисунке 11 представлена UML-диаграмма компонентов.

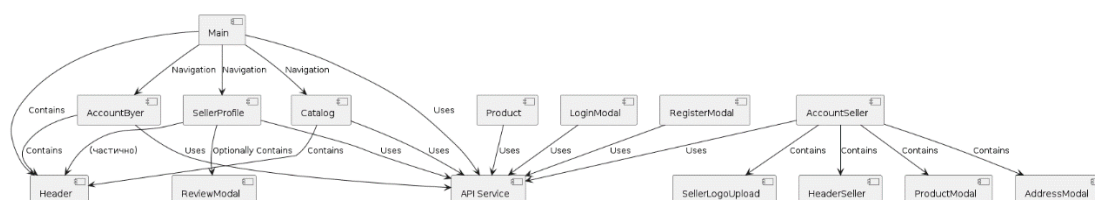


Рисунок 11 – UML-диаграмма компонентов

5.4 Описание интерфейса приложения

При запуске платформы перед пользователем сразу появляется выбор: войти в уже созданный аккаунт либо зарегистрироваться, если аккаунта ещё нет. На странице авторизации вы вводите свою электронную почту и пароль, после чего, при корректности данных, система направляет вас либо в каталог товаров (если вы зашли как покупатель), либо в личный кабинет (если вы продавец).

В каталоге отображаются карточки товаров, присутствуют фильтры для удобного поиска, а также интегрирована интерактивная карта, облегчающая навигацию. Пока покупатель не авторизуется, доступ к подробной информации по карточкам товаров для него недоступен.

Личный кабинет продавца доступен ему только после авторизации. Находясь в нем, производитель может менять описание и лого его хозяйства, управлять ассортиментом и адресами мест сбыта своих товаров. На рисунке 12 представлено окно авторизации пользователя.

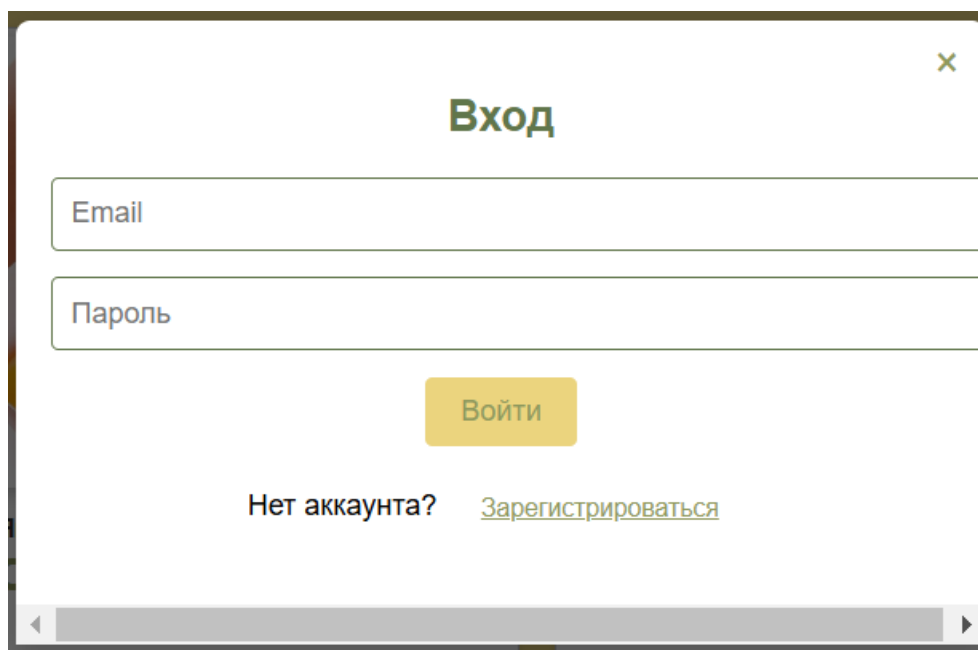


Рисунок 12 – Экран авторизации пользователя

В каталоге пользователь может выбрать любую категорию в левом блоке и каталог отфильтруется и выдаст список товаров, относящихся только к этой категории. Отфильтрованный каталог изображен на рисунке 13.

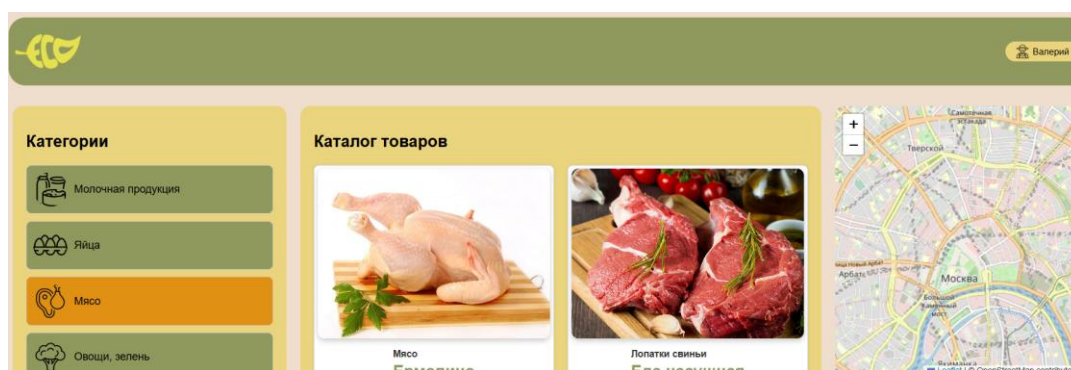


Рисунок 13 – Отфильтрованный каталог

При нажатии на любой товар, открывается его карточка с подробностями. Тут можно увидеть полное описание товара, перейти на карточку продавца. На рисунке 14 представлена страница карточки товара.

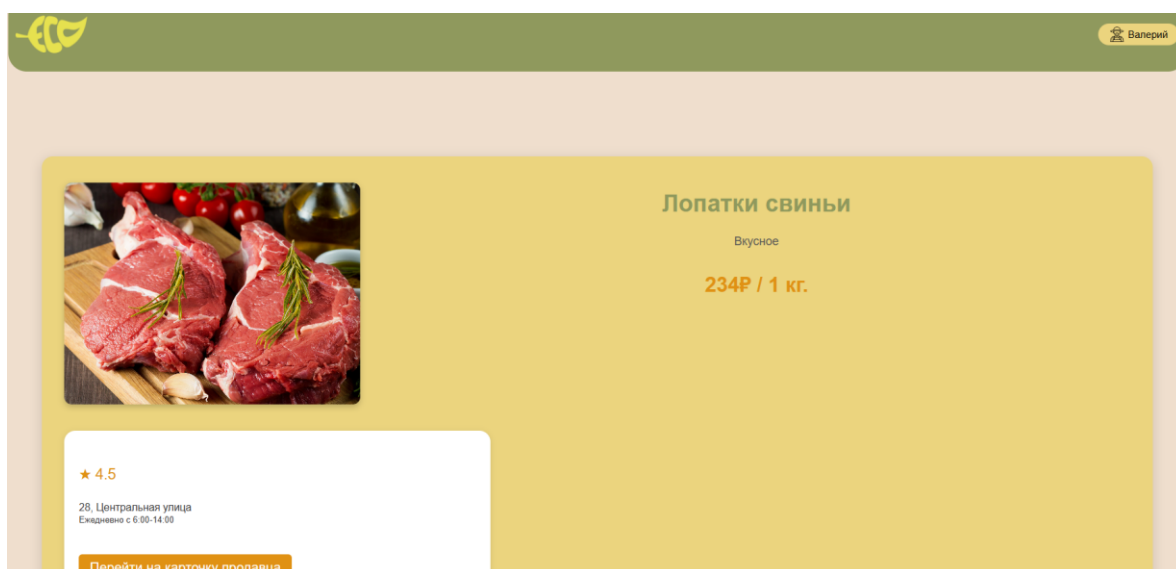


Рисунок 14 – Карточка с подробной информацией о товаре

Через кнопку перехода на карточку поставщика мы можем ближе узнать продавца, увидеть другие его товары, ознакомиться с описанием, увидеть адреса сбыта, а также открывается возможность написать отзыв и поставить рейтинг конкретному фермеру. На рисунке 15 представлена карточка продавца.

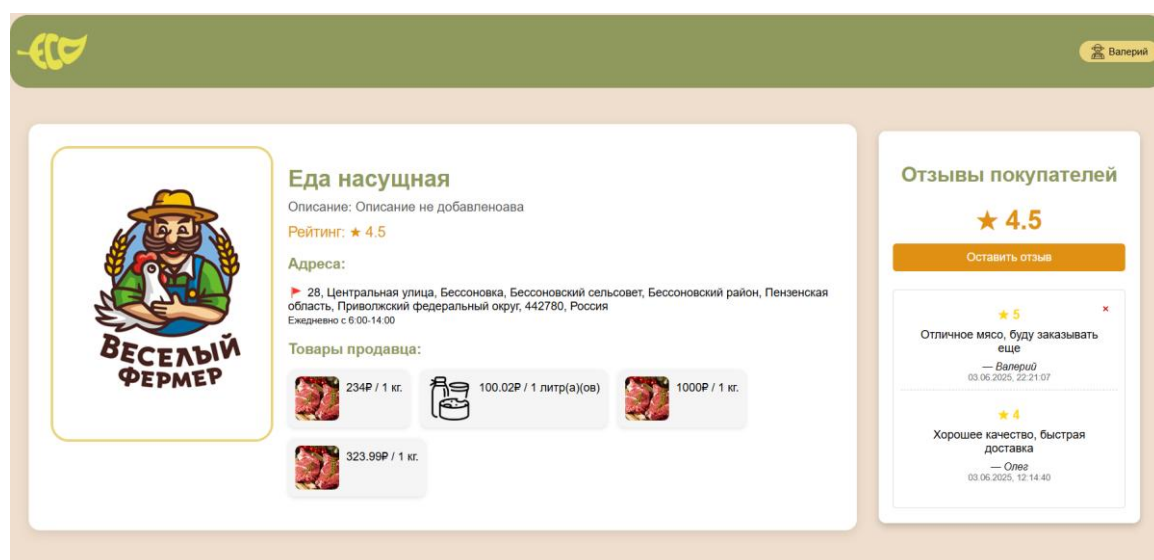


Рисунок 15 – Карточка продавца

У пользователей с ролью «продавец» предоставляется доступ к многофункциональному личному кабинету. Производитель может изменять

лого производства, а также изменять описание. Фермер может в том числе управлять своими адресами сбыта и ассортиментом. На рисунке 16 изображен личный кабинет производителя.

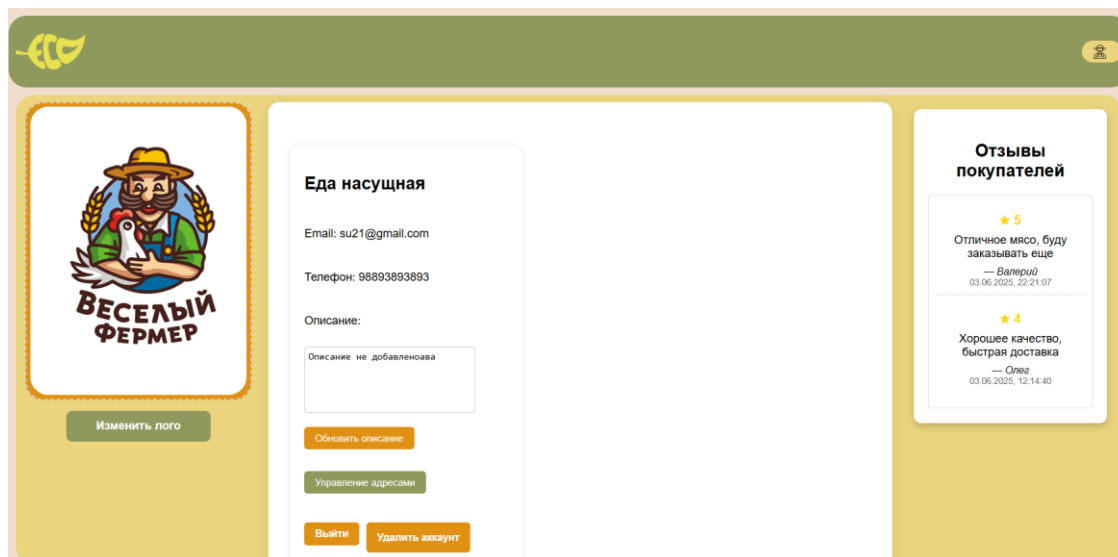


Рисунок 16 – Личный кабинет производителя

При нажатии на кнопку «Управление адресами» открывается модальное окно, где пользователь может добавить новый адрес, либо удалить один из адресов, который есть в списке всех добавленных производителем мест сбыта ранее. На рисунке 17 изображено модальное окно управления адресами.

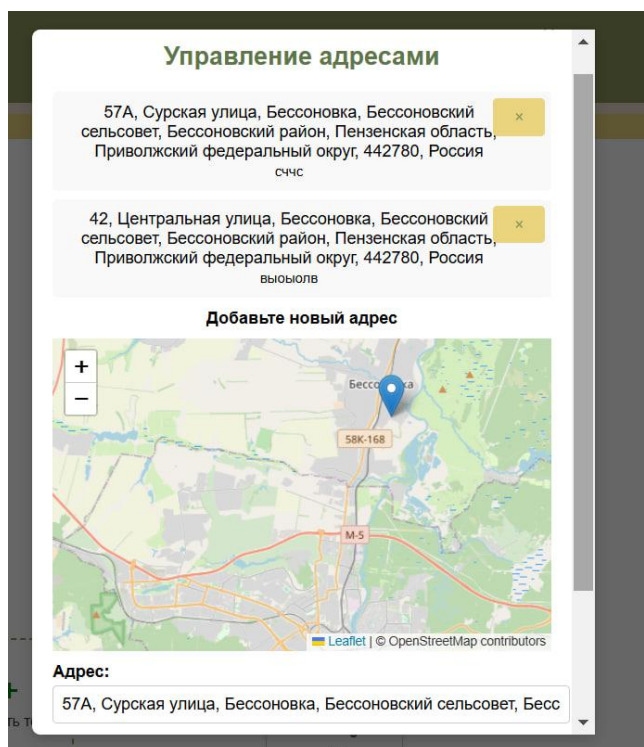


Рисунок 17 – Модальное окно для управления адресами

При нажатие на кнопку «Добавить товар» открывается модальное окно для добавления новых товаров, дается заполнить обязательные поля (фото, выбрать категорию, название товара, описание товара, цена, количество, единица измерения (кг, шт и так далее)), затем убедившись в правильности введенных данных можно нажать на кнопку сохранить, после чего добавленный товар отобразиться в сетки всех товаров этого производителя. На рисунке 18 изображено модальное окно для добавления нового товара.

Добавить товар

Фото товара:

Выберите файл

птица.jpg



Категория:

Мясо

Название товара:

Мясо цыпленка

Описание товара:

Домашнее мясо, отличное, вкусное.

Цена:

Сохранить

Рисунок 18 – Модальное окно для добавления нового товара

5.5 Тестирование приложения

При разработке любого приложения одной из важнейших задач является тестирование. Это позволяет заранее выявить ошибки в коде, недочеты в работе функций и моменты, где интерфейс может быть неудобен для пользователя. Благодаря тщательному тестированию мы можем улучшить стабильность работы приложения и повысить качество взаимодействия с ним. Для этого планируется проверить следующие ключевые функции приложения:

- авторизация;
- регистрация;
- добавление новых товаров;
- добавление нового адреса;
- добавление отзыва.

5.5.1 Авторизация

При нажатии на кнопку войти открывается модальное (рисунок 19) окно двумя полями для ввода email и пароль. При вводе всех данных пользователь может нажать на кнопку войти.

Рисунок 19 – Окно авторизации

Если пользователь введет неверные данные в одно из полей, сразу появится предупреждение о допущенной ошибке. Как видно на рисунке 20, при вводе неправильно оформленного email, система уведомляет пользователя о необходимости исправления.

Рисунок 20 – Сообщение ошибки при вводе данных

После успешного входа в систему, пользователь попадает либо в каталог товаров, либо в личный кабинет – выбор зависит от его роли.

5.5.2 Регистрация

Если у пользователя ещё нет аккаунта, ему необходимо зарегистрироваться. Для начала регистрации достаточно нажать на кнопку «Зарегистрироваться», расположенную под основной кнопкой входа. После нажатия открывается модальное окно, в котором пользователь должен выбрать, под какой ролью он будет зарегистрирован – покупателя или продавца. Далее нужно заполнить все обязательные поля: имя, фамилию, email, телефон и пароль. Для продавцов предусмотрено дополнительное поле, где указывается наименование производства. Пример окна регистрации можно увидеть на рисунке 21.

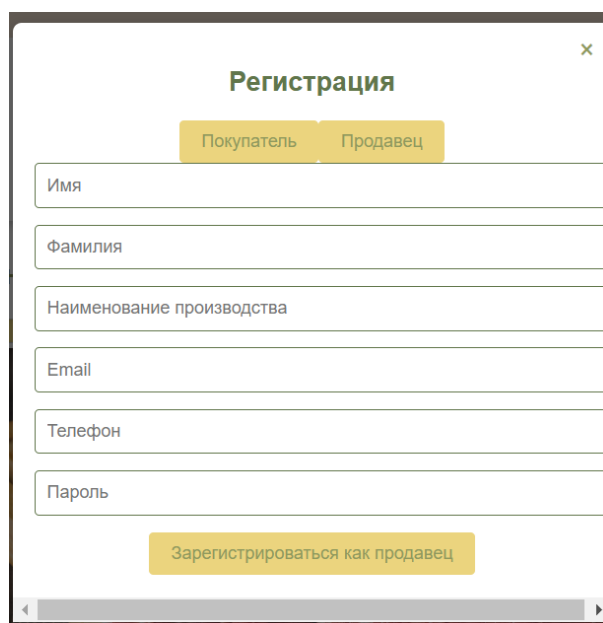


Рисунок 21 – Окно регистрации

После регистрации пользователь переходит либо в каталог, либо в личный кабинет (в зависимости от роли).

5.5.3 Добавление новых товаров

В личном кабинете продавец может добавлять к своему ассортименту новые товары. При нажатии на кнопку «добавить товар» открывается модальное окно, где предлагается заполнить необходимые поля (фото, категорию, название товара, описание товара, цена, количество, единица измерения (кг, шт и так далее)), после сохранения добавленный товар отобразится в сетки всех товаров этого производителя. На рисунке 22 изображено модальное окно для добавления нового товара.

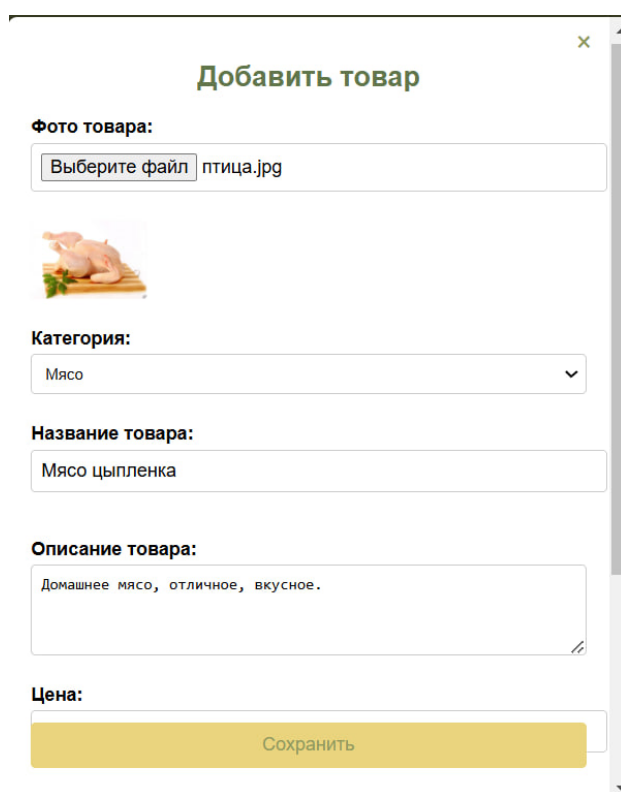


Рисунок 22 – Модальное окно для добавления нового товара

На рисунке 23 изображена сетка товаров продавца.



Рисунок 23 – Сетка товаров продавца

5.5.4 Добавление нового адреса

Перейдя по кнопке, пользователь увидит модальное окно, где сверху будет список имеющихся адресов сбыта с возможностью их удалить. Под списком будет карта с меткой для добавления новых адресов, при добавлении нового адреса он попадает в список к действующим адресам данного производителя (рисунок 24).

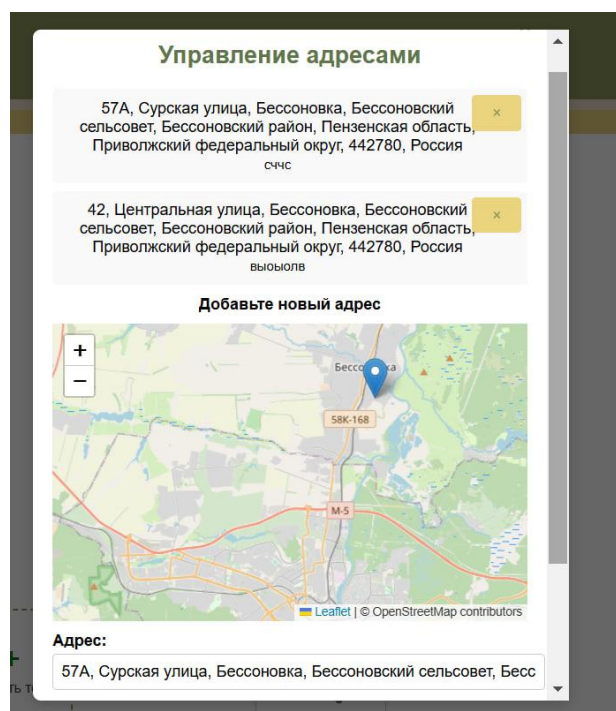


Рисунок 24 – Модальное окно для управления адресами

6 Безопасность жизнедеятельности

Несмотря на всю кажущуюся простоту использования современных компьютеров и смартфонов, работа с ними сопряжена с определёнными рисками. Об этом свидетельствует стандарт ГОСТ 12.0.003-2015, в котором подробно описаны вредные и опасные факторы, влияющие на здоровье при работе с компьютерной техникой. Соблюдение специальных рекомендаций и нормативов помогает защитить пользователя от негативных последствий как для организма, так и для имущества [5].

6.1 Правила и техника безопасности при работе с ПК/ ноутбуком

Согласно стандарту, основные вредные факторы включают:

- перегрев – отдельные компоненты компьютера выделяют много тепла, что повышает температурный фон в помещении;
- монотонность – задачи могут приводить к усталости и снижению концентрации;
- зрительная нагрузка – длительное использование экрана вызывает переутомление глаз;
- статическое электричество – случайный контакт с некоторыми элементами техники может привести к электрическим ударам;
- недостаточное освещение – плохое освещение рабочей зоны может способствовать утомлению;
- электромагнитные поля – работающая техника создаёт электрические и магнитные поля, что потенциально вредно для организма;
- высокая яркость и контрастность – чрезмерный блеск экрана негативно влияет на зрение при длительном обращении.

Чтобы минимизировать эти риски, рекомендуется организовать рабочее место с учётом принципов эргономики:

					<i>ПГУ1.090301.14.001 ПЗ</i>	Лист
						45
Изм.	Лист	№ докум.	Подпись	Дата		

– правильная посадка: сидите прямо с поддержкой спины, руки должны быть на уровне клавиатуры, а ноги – устойчиво стоять на полу. Монитор лучше располагать на расстоянии вытянутой руки с небольшим наклоном (около 30°), чтобы снизить нагрузку на шею и глаза;

– хорошее освещение: рабочее место должно быть достаточно освещено, но без резких бликов на мониторе, чтобы избежать утомления глаз.

Кроме того, важно следить за исправностью техники, не использовать устройства при наличии повреждений и не оставлять их включёнными без присмотра. Такие меры помогают снизить воздействие электромагнитных полей, что также рекомендовано различными источниками по безопасности труда.

6.2 Правила и техника безопасности при работе со смартфоном

Смартфоны стали неотъемлемой частью нашей жизни, однако их эксплуатация также требует соблюдения правил безопасности:

– не разбирайте устройство самостоятельно. Попытки модифицировать, отремонтировать или вскрывать смартфон могут привести к повреждениям и снизить его работоспособность;

– защита от физических повреждений: избегайте ударов, сгибания, воздействия огня или воды;

– используйте оригинальные аксессуары: для зарядки смартфона следует использовать только оригинальные зарядные устройства и кабели. Это поможет избежать короткого замыкания, перегрева и других проблем. Заряжать телефон в местах с плохой вентиляцией или под подушкой не рекомендуется.

Смартфоны также излучают радиоэнергию, однако современные исследования пока не обнаружили доказательств серьёзного вреда для здоровья при стандартном использовании. Тем не менее, в целях безопасности

рекомендуется не держать телефон слишком близко к телу во время его работы, а также часто менять положение устройства при звонках [6].

Соблюдение этих рекомендаций помогает не только продлить срок службы техники, но и обеспечить ваше здоровье, позволяя безопасно и эффективно использовать устройства в повседневной жизни.

					ПГУ1.090301.14.001 ПЗ	Лист
						47
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В ходе выполнения выпускной квалификационной работы была создана веб-платформа «Электронный рынок фермерской продукции». Для её разработки использовались Flask на Python для серверной части и React.js для создания динамического интерфейса. Хранение данных организовано с помощью реляционной базы данных MySQL.

Разработанная платформа позволяет покупателям находить местных фермеров через интерактивную карту, просматривать каталог свежих фермерских продуктов с подробными описаниями, возможность оставлять отзывы и оценки продавцам. А фермерам, в свою очередь, предлагается возможность создавать карточки своих товаров, управлять ассортиментом и ценами, получать обратную связь от покупателей.

Одним из достоинств разработанной платформы является наличие удобного и интуитивно понятного для пользователя интерфейса с адаптивным дизайном. А система рейтингов и отзывов и гибкие фильтры поиска обеспечивает прозрачность и простоту использования платформы.

В дальнейшем можно усовершенствовать разработанное приложение, внедрить системы подписок на регулярные поставки, разработать мобильное приложение для iOS/Android, интеграция с платежными системами, создание чата, а добавление системы рекомендаций на основе предпочтений и разработка программы лояльности позволят расширить количество потенциальных пользователей.

Список использованных источников

1. Tastesofrussia.ru: [Электронный ресурс]. URL: <https://www.tastesofrussia.ru/novosti/objom-onlajn-prodazh-fermerskih-produktov-vyrastet-do-50-mlrd-k-2024/> (дата обращения: 19.05.2025)
2. Habr.com: [Электронный ресурс]. URL: <https://habr.com/ru/companies/ruvds/articles/428077/> (дата обращения: 20.05.2025)
3. React.dev: [Электронный ресурс]. URL: <https://reactdev.ru/learn/describing-the-ui/#conditional-rendering> (дата обращения: 20.05.2025)
4. Habr.com: [Электронный ресурс]. URL: <https://habr.com/ru/articles/485618/> (дата обращения: 24.05.2025)
5. Internet-law.ru: [Электронный ресурс]. URL: <https://internet-law.ru/gosts/gost/62075/> (дата обращения: 26.05.2025).
6. Bbc.com: [Электронный ресурс]. URL: <https://www.bbc.com/russian/features-43226267> (дата обращения: 26.05.2025)

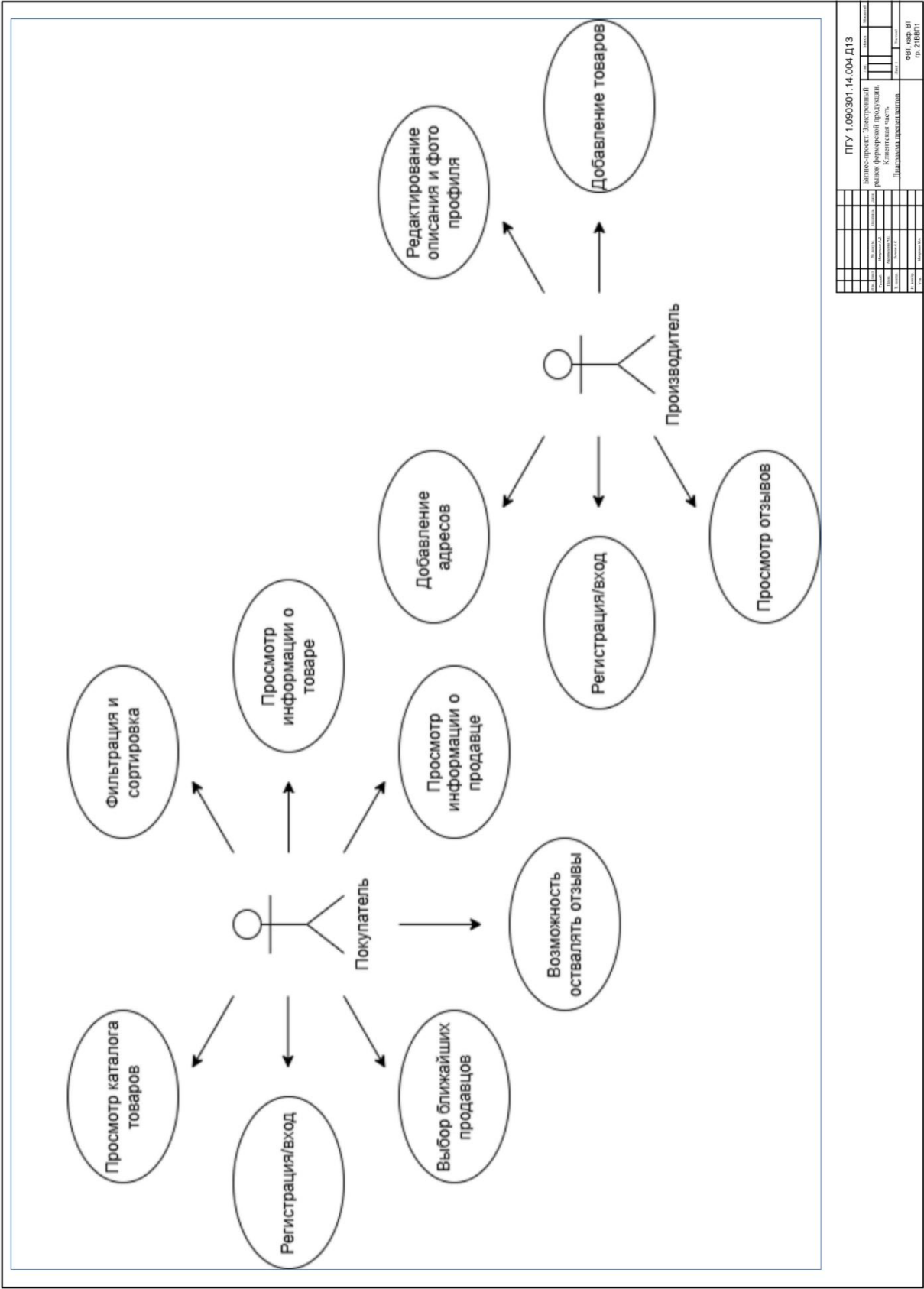
Приложение А

(Обязательное)

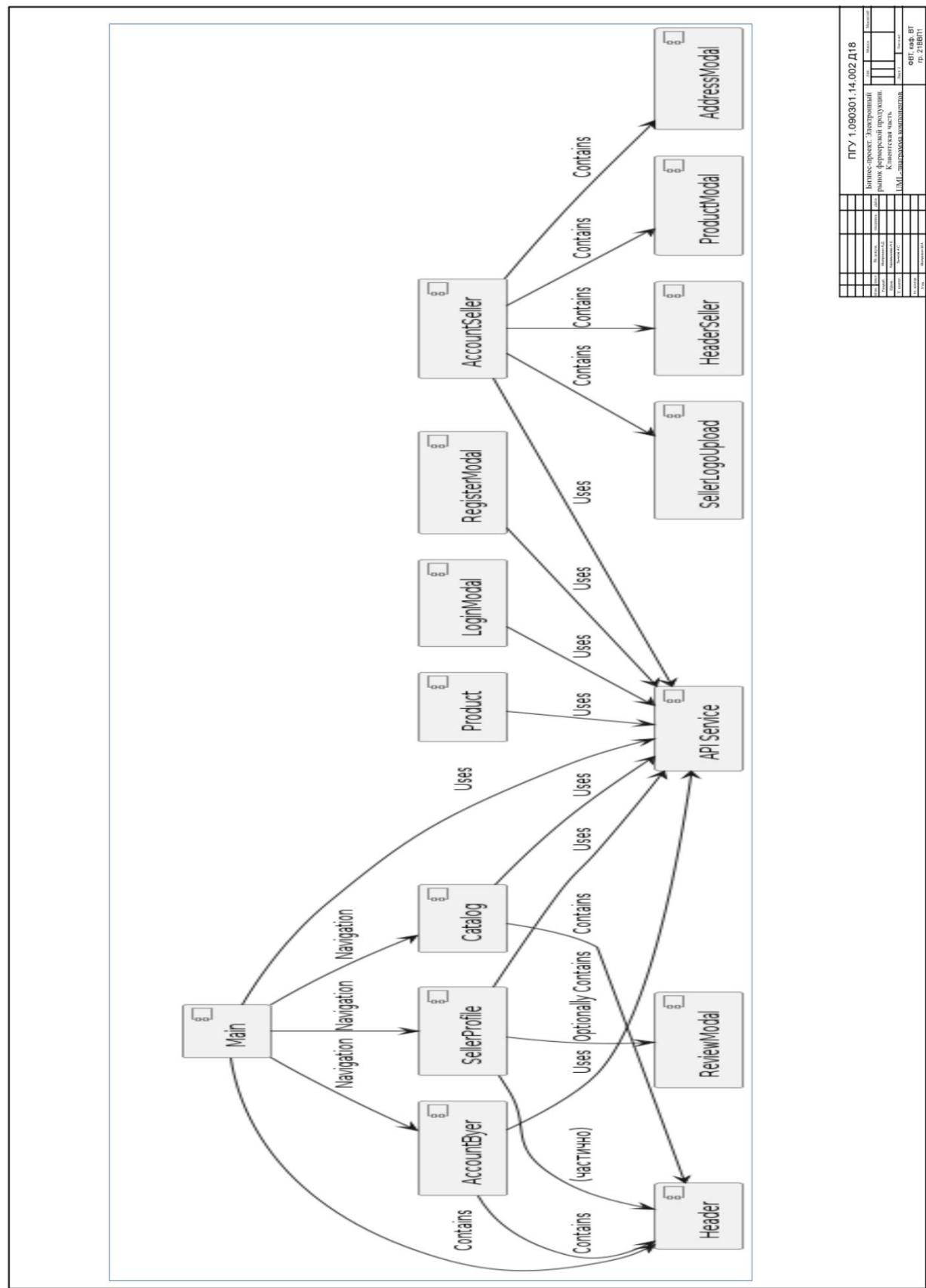
Бизнес-проект. Электронный рынок фермерской продукции. Клиентская часть
UML-диаграммы

					ПГУ1.090301.14.001 ПЗ	Лист
						50
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1-Диаграмма прецедентов

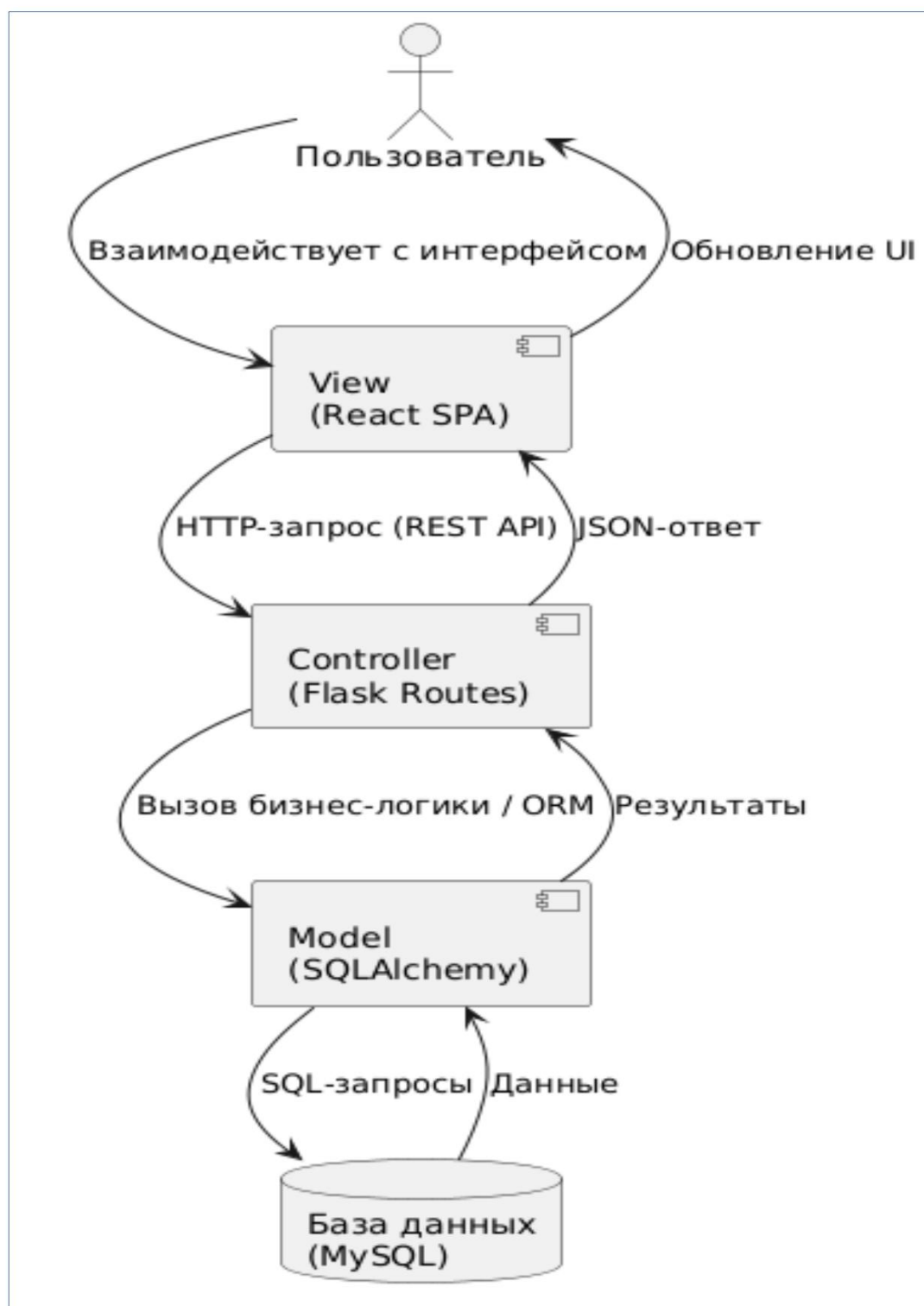


Приложение А.2-Диаграмма компонентов



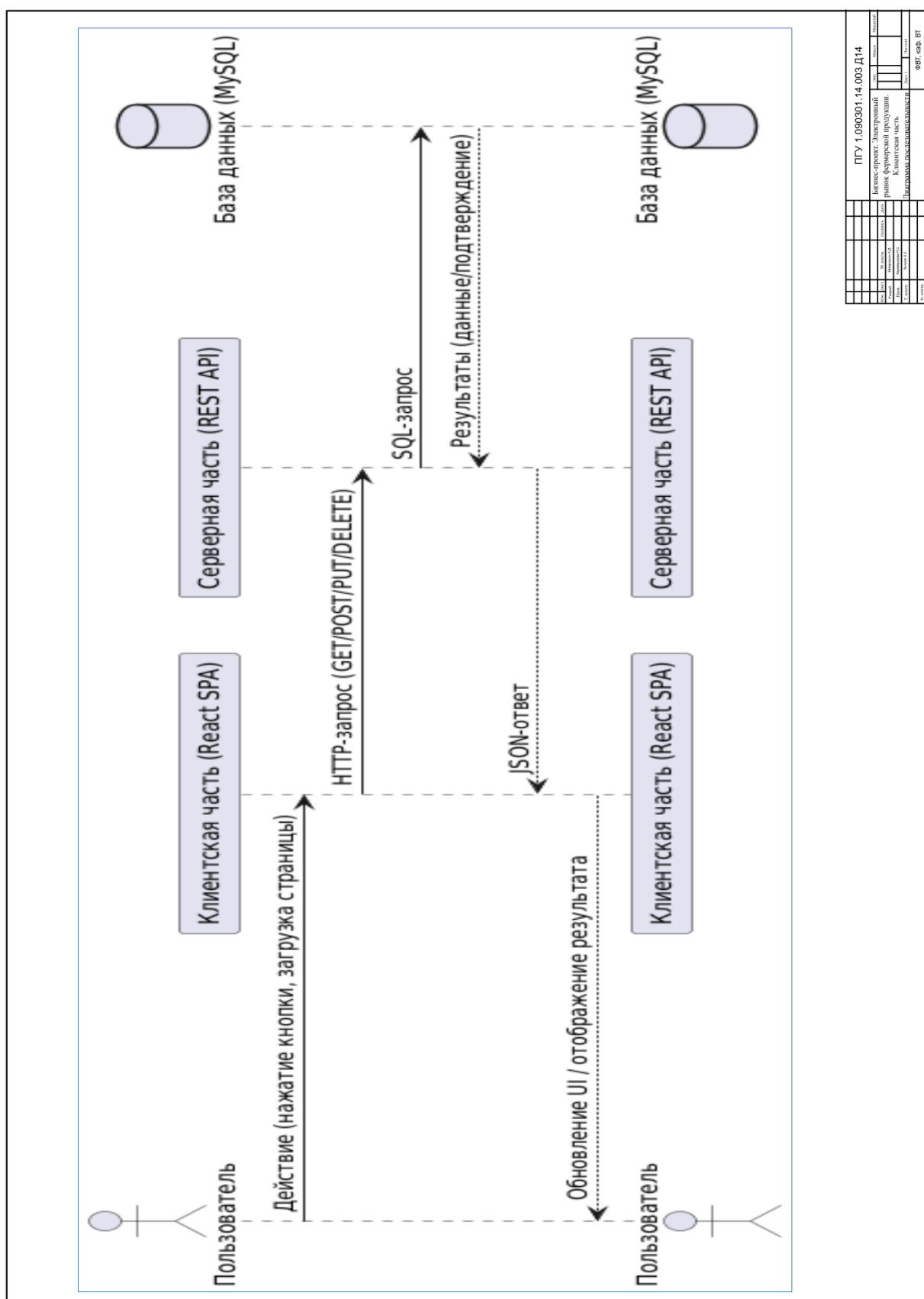
ПУУ 1.090301.14.002 Д18										№ п/п		№ документа		Дата		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Исполнитель		Испол	
-------------------------	--	--	--	--	--	--	--	--	--	-------	--	-------------	--	------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------------	--	-------	--

Приложение А.3-Диаграмма компонентов

[illegible]

Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.4-Диаграмма последовательности

[illegible]

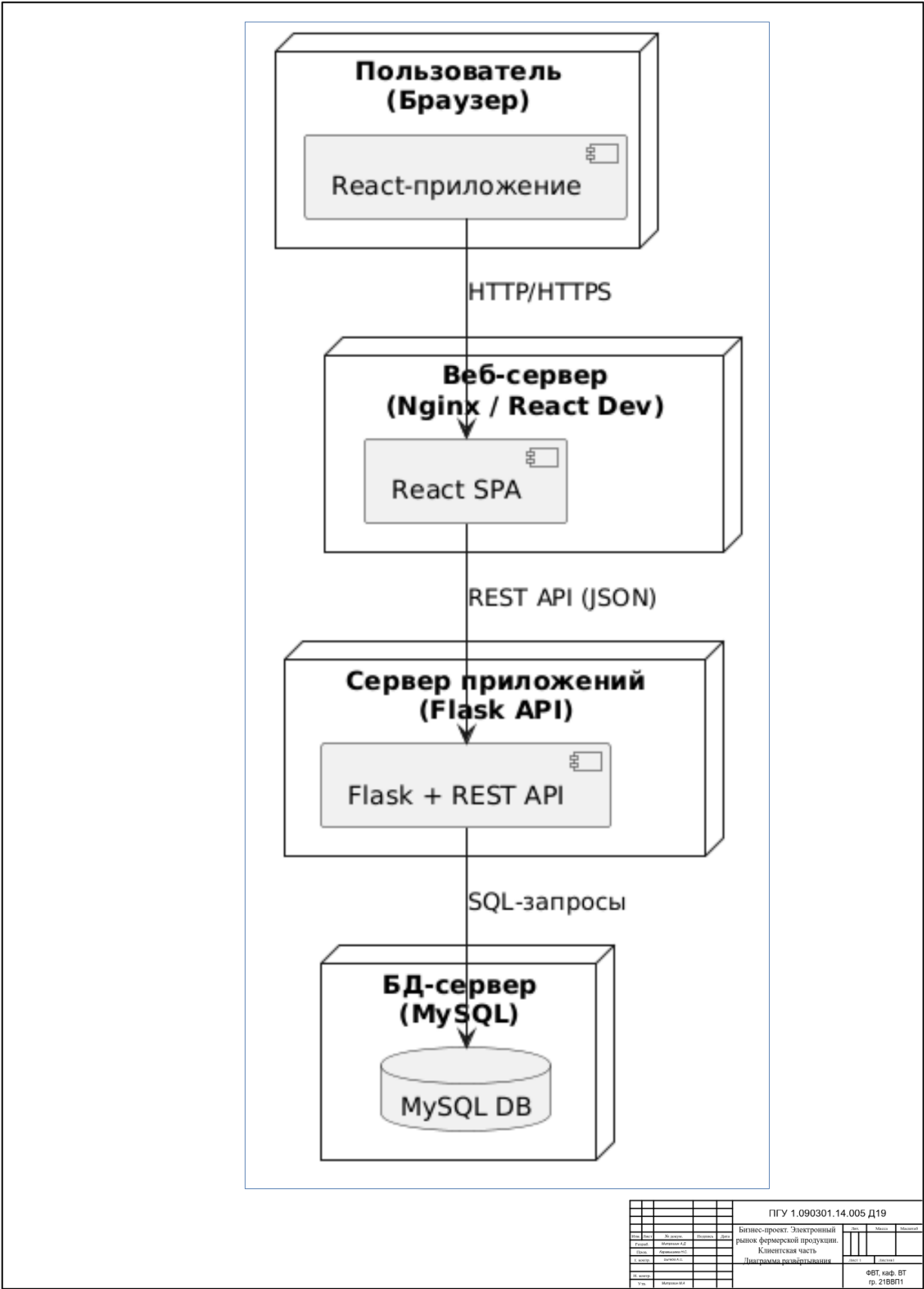
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.14.001 ПЗ

Лист

54

Приложение А.5-Диаграмма развертывания



Изм.	Лист	№ докум.	Подпись	Дата

Приложение Б

(Обязательное)

Бизнес-проект. Электронный рынок фермерской продукции. Клиентская часть
Листинг программы

					ПГУ1.090301.14.001 ПЗ	Лист
						56
Изм.	Лист	№ докум.	Подпись	Дата		

Файл «AppContent.js»

```
import React, { useState, useEffect } from "react";
import { Routes, Route, useLocation, useNavigate } from
"react-router-dom";
import Main from "../pages/Main";
import Product from "../pages/Product";
import AccountByer from "../pages/AccountByer";
import AccountSeller from "../pages/AccountSeller";
import Catalog from "../pages/Catalog";
import SellerProfile from "../pages/SellerProfile";
import LoginModal from "../components/LoginModal";
import RegisterModal from "../components/RegisterModal";
import Header from "../components/Header";
import HeaderBuyer from "../components/HeaderBuyer";
import { getBuyerData } from "../services/api";
import "../styles/style.css";

function AppContent() {
  // Глобальное состояние аутентификации, имени покупателя и
  модальных окон
  const [isAuth, setIsAuth] =
  useState(!localStorage.getItem("access_token"));
  const [buyerName, setBuyerName] = useState("");
  const [isLoginOpen, setIsLoginOpen] = useState(false);
  const [isRegisterOpen, setIsRegisterOpen] =
  useState(false);

  const location = useLocation();
  const navigate = useNavigate();

  // Если токен есть, получаем данные покупателя с сервера
  useEffect(() => {
    const token = localStorage.getItem("access_token");
    if (token) {
      getBuyerData()
        .then((res) => {
          if (res.data && res.data.status === "success" &&
  res.data.buyer) {
            setIsAuth(true);
            setBuyerName(res.data.buyer.first_name);
          } else {
            setIsAuth(false);
            setBuyerName("");
          }
        })
        .catch((error) => {
          console.error("Ошибка получения данных
  покупателя:", error);
          setIsAuth(false);
          setBuyerName("");
        });
    }
  });
}
```

					ПГУ1.090301.14.001 ПЗ	Лист
						57
Изм.	Лист	№ докум.	Подпись	Дата		

```

        });
    } else {
        setIsAuth(false);
        setBuyerName("");
    }
}, []);

// Функция выхода – удаление токена и обновление
глобального состояния
const handleLogout = () => {
    localStorage.removeItem("access_token");
    setIsAuth(false);
    setBuyerName("");
    setIsLoginOpen(false);
    setIsRegisterOpen(false);
    navigate("/");
};

const hideGlobalHeaderPaths = ["/accountSeller"];

return (
    <>
        {!hideGlobalHeaderPaths.includes(location.pathname) &&
            (isAuth ? (
                <HeaderBuyer buyerName={buyerName}
onLogout={handleLogout} />
            ) : (
                <Header setIsLoginOpen={setIsLoginOpen} />
            ))
        }

        {/* Модальные окна для входа и регистрации */}
        {isLoginOpen && (
            <LoginModal
                isOpen={isLoginOpen}
                onClose={() => setIsLoginOpen(false)}
                onOpenRegister={() => setIsRegisterOpen(true)}
            />
        )}
        {isRegisterOpen && (
            <RegisterModal
                isOpen={isRegisterOpen}
                onClose={() => setIsRegisterOpen(false)}
            />
        )}

        <Routes>
            <Route path="/" element={<Main />} />
            <Route path="/product/:id" element={<Product />} />
        </Routes>
    </>

```

```

        <Route path="/account" element={<AccountByer
onLogout={handleLogout} />} />
        <Route path="/accountSeller" element={<AccountSeller
/>} />
        <Route path="/catalog" element={<Catalog />} />
        <Route path="/sellerProfile/:sellerId"
element={<SellerProfile />} />
    </Routes>
</>
);
}

export default AppContent;

```

Файл «api.js»

```

import axios from "axios";

const API_BASE_URL =
"https://farmmarket.drunkenhedgehog.ru/api";

const api = axios.create({
  baseURL: API_BASE_URL,
});

api.interceptors.request.use((config) => {
  const token = localStorage.getItem("access_token");
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});

export const registerBuyer = async (data) =>
api.post("/buyers/register", data);
export const loginBuyer = async (data) =>
api.post("/buyers/login", data);
export const registerSellers = async (data) =>
api.post("/sellers/register", data);
export const loginSellers = async (data) =>
api.post("/sellers/login", data);
export const updateMainImage = async (data) =>
api.patch("/sellers/update_main_image", data);
export const updateSellerDescription = async (data) =>
api.patch("/sellers/update_description", data);
export const getBuyerData = async () =>
api.get("/buyers/get_data");
export const getSellerData = async () =>
api.get("/sellers/get_data");
export const getSellerAddresses = async (sellerId) =>
api.get(`/sellers/addresses/${sellerId}`);

```

```

    export const addSellerAddress = async (data) =>
api.post("/sellers/add_address", data);
    export const getCategories = async () =>
api.get("/categories/get");
    export const getNearbyProducts = async (latitude, longitude,
radius = 30) =>

api.get(`/products/nearby?latitude=${latitude}&longitude=${longitu
de}&radius=${radius}`);
    export const deleteSellerAddress = async (addressId) =>
api.delete(`/sellers/delete_address/${addressId}`);
    export const getProducts = async () => api.get("/products");
    export const getSellerPublicData = (sellerId) =>
api.get(`/sellers/get_public_data/${sellerId}`);
    export const createProduct = async (data) =>
api.post("/products/create", data);
    export const getProductsBySeller = async (sellerId) =>
api.get(`/products/seller/${sellerId}`);
    export const deleteProduct = async (productId) =>
api.delete(`/products/delete/${productId}`);
    export const getFavorites = async () =>
api.get("/favorites");
    export const addToFavorites = async (productId) =>
api.post("/favorites", { productId });
    export const createReview = async (data) =>
api.post("/reviews/create", data);
    export const getSellerReviews = async (sellerId) =>
api.get(`/reviews/seller/${sellerId}`);
    export const deleteReview = async (reviewId) =>
api.delete(`/reviews/delete/${reviewId}`);
    export const deleteBuyerAccount = async () =>
api.delete("/buyers/delete");
    export const getNearbySellers = async (latitude, longitude,
radius = 30) =>

api.get(`/sellers/nearby?latitude=${latitude}&longitude=${longitud
e}&radius=${radius}`);
    export const deleteSellerAccount = async () =>
api.delete("/sellers/delete");

    export default api;
    export const removeProductFromFavorites = async (productId)
=> {
    const response = await
api.delete(`/favorites/${productId}`);
    return response.data;
};

```

Файл «Main.js»

```
import React, { useState, useEffect } from "react";
```

					ПГУ1.090301.14.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

```

import { getCategories, getNearbyProducts } from
"../services/api";
import NearbySellersMap from
"../components/NearbySellersMap";
import "../styles/style.css";

const Main = () => {
  const [categories, setCategories] = useState([]);
  const [products, setProducts] = useState([]);
  // По умолчанию координаты Москвы
  const [location, setLocation] = useState([55.755826,
37.6173]);
  const radius = 30; // Радиус по умолчанию для товаров

  // Новое состояние для выбранной категории
  const [selectedCategoryId, setSelectedCategoryId] =
useState(null);

  const handleUnauthorizedClick = () => {
    alert("Войдите или зарегистрируйтесь, чтобы посмотреть
товар");
  };

  // Пытаемся получить геолокацию пользователя
  useEffect(() => {
    if (navigator.geolocation) {
      navigator.geolocation.getCurrentPosition(
        ({ coords }) => {
          const lat = coords.latitude;
          const lon = coords.longitude;
          if (lat >= -90 && lat <= 90 && lon >= -180 && lon
<= 180) {
            setLocation([lat, lon]);
          } else {
            console.error("Некорректные координаты:", lat,
lon);
          }
        },
        (err) => console.warn("Ошибка геолокации:",
err.message)
      );
    }
  }, []);

  // Загружаем категории
  useEffect(() => {
    async function fetchCategories() {
      try {
        const res = await getCategories();
        if (res.data && Array.isArray(res.data)) {
          setCategories(res.data);
        }
      } catch (err) {
        console.error("Ошибка загрузки категорий:", err);
      }
    }
    fetchCategories();
  });
}

```

```

        } else if (res.data.categories) {
            setCategories(res.data.categories);
        }
    } catch (error) {
        console.error("Ошибка получения категорий:", error);
    }
}
fetchCategories();
}, []);

// Загружаем товары поблизости
useEffect(() => {
    async function fetchProducts() {
        try {
            const [latitude, longitude] = location;
            console.log("Запрос товаров с параметрами:", {
latitude, longitude, radius });
            const res = await getNearbyProducts(latitude,
longitude, radius);
            if (res.data.status === "success" &&
res.data.products) {
                setProducts(res.data.products);
            } else {
                console.error("Ошибка получения товаров
поблизости:", res.data);
            }
        } catch (error) {
            console.error("Ошибка получения товаров:", error);
        }
    }
    fetchProducts();
}, [location]);

// Фильтруем товары по выбранной категории (если выбрана)
const visibleProducts = selectedCategoryId
    ? products.filter(product => product.category_id ===
selectedCategoryId)
    : products;

return (
    <div className="main-container">
        {/* Левый блок: Категории */}
        <div className="left-panel">
            <h2>Категории</h2>
            <div className="categories-list">
                {categories.map((cat) => (
                    <div
                        key={cat.id}
                        className={`category-card ${selectedCategoryId
=== cat.id ? 'selected' : ''}`}
                        onClick={() => setSelectedCategoryId(cat.id)}

```

```

        >
        <img src={cat.image_url} alt={cat.name}
className="category-image" />
        <p>{cat.name}</p>
    </div>
    )})
</div>
</div>

{/* Центральный блок: Каталог товаров поблизости */}
<div className="center-panel">
    <h2>Каталог товаров</h2>
    <div className="catalog-block">
        {visibleProducts.length > 0 ? (
            visibleProducts.map((product) => (
                <div
                    key={product.id}
                    className="product-card"
                    onClick={() => handleUnauthorizedClick()}
                >
                    <div className="product-image-container">
                        <img src={product.main_image_url}
alt={product.name} className="product-image" />
                    </div>
                    <div className="product-details">
                        <div className="left-details">
                            <p className="product-
name"><strong>{product.name}</strong></p>
                            <p className="seller-
name">{product.seller_name}</p>
                        </div>
                        <div className="right-details">
                            <p className="product-price">
                                {product.price} {product.quantity_unit}
                            </p>
                            <p className="product-rating">★ 0.0</p>
                        </div>
                    </div>
                </div>
            ))
        ) : (
            <p>Нет товаров поблизости</p>
        )}
    </div>
</div>

<div className="right-panel">
    <NearbySellersMap userLocation={location} radius={10}
/>
</div>

```

```

        </div>
    );
};

export default Main;
Файл «Catalog.js»

import React, { useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";
import { getCategories, getNearbyProducts } from
"../services/api";
import "../styles/style.css";

const Catalog = () => {
    const navigate = useNavigate();
    const [categories, setCategories] = useState([]);
    const [products, setProducts] = useState([]);
    const [location, setLocation] = useState([55.755826,
37.6173]);
    const radius = 30;

    // Получаем геолокацию пользователя
    useEffect(() => {
        if (navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(
                ({ coords }) => {
                    const lat = coords.latitude;
                    const lon = coords.longitude;
                    if (lat >= -90 && lat <= 90 && lon >= -180 && lon
<= 180) {
                        setLocation([lat, lon]);
                    } else {
                        console.error("Некорректные координаты:", lat,
lon);
                    }
                },
                (err) => console.warn("Ошибка геолокации:",
err.message)
            );
        }
    }, []);

    // Загружаем категории
    useEffect(() => {
        async function fetchCategories() {
            try {
                const res = await getCategories();
                if (res.data.categories) {
                    setCategories(res.data.categories);
                }
            } catch (error) {

```

```

        console.error("Ошибка получения категорий:", error);
    }
}
fetchCategories();
}, []);

// Загружаем товары поблизости
useEffect(() => {
    async function fetchProducts() {
        try {
            const [latitude, longitude] = location;
            const res = await getNearbyProducts(latitude,
longitude, radius);
            if (res.data.status === "success" &&
res.data.products) {
                setProducts(res.data.products);
            }
        } catch (error) {
            console.error("Ошибка получения товаров:", error);
        }
    }
    fetchProducts();
}, [location]);

const handleProductClick = (productId) => {
    navigate(`/product/${productId}`);
};

return (
    <div className="main-container">
        <div className="left-panel">
            <h2>Категории</h2>
            <div className="categories-list">
                {categories.map((cat) => (
                    <div key={cat.id} className="category-card">
                        <img src={cat.image_url} alt={cat.name}
className="category-image" />
                        <p>{cat.name}</p>
                    </div>
                ))}
            </div>

            <div className="center-panel">
                <h2>Каталог товаров</h2>
                <div className="catalog-block">
                    {products.length > 0 ? (
                        products.map((product) => (
                            <div key={product.id} className="product-card"
onClick={ () => handleProductClick(product.id)}>
                                <div className="product-image-container">

```

```

        <img src={product.main_image_url}
alt={product.name} className="product-image" />
      </div>
      <div className="product-details">
        <div className="left-details">
          <p className="product-
name"><strong>{product.name}</strong></p>
          <p className="seller-
name">{product.seller_name}</p>
        </div>
        <div className="right-details">
          <p className="product-
price">{product.price} {product.quantity_unit}</p>
          <p className="product-rating">★ 0.0</p>
        </div>
      </div>
    </div>
  ))
) : (
  <p>Нет товаров поблизости</p>
)
</div>
</div>

<div className="right-panel">
  <h2>Дополнительная информация</h2>
  <div className="right-block-content"></div>
</div>
</div>
);
};

```

export default Catalog;

Файл «AccountSeller.js»

```

import React, { useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";
import HeaderSeller from "../components/HeaderSeller";
import SellerLogoUpload from
"../components/SellerLogoUpload";
import ProductModal from "../components/ProductModal";
import AddressModal from "../components/AddressModal";
import {
  getSellerData,
  updateSellerDescription,
  getProductsBySeller,
  deleteProduct,
  deleteSellerAccount,
  getSellerReviews // функция для получения отзывов
} from "../services/api";

```

```

import "../styles/accountSeller.css";

function AccountSeller() {
  const [seller, setSeller] = useState(null);
  const [description, setDescription] = useState("");
  const [isProductModalOpen, setIsProductModalOpen] =
useState(false);
  const [isAddressModalOpen, setIsAddressModalOpen] =
useState(false);
  const [products, setProducts] = useState([]);

  // Состояния для отзывов
  const [reviews, setReviews] = useState([]);
  const [loadingReviews, setLoadingReviews] = useState(true);

  const navigate = useNavigate();

  // Получение данных продавца
  useEffect(() => {
    async function fetchSellerData() {
      try {
        const response = await getSellerData();
        if (response.data.status === "success") {
          setSeller(response.data.seller);
          setDescription(response.data.seller.description ||
"Описание не добавлено");
        } else {
          console.error("Продавец не найден");
        }
      } catch (error) {
        console.error("Ошибка получения данных продавца",
error);
      }
    }
    fetchSellerData();
  }, []);

  // Получаем товары продавца после загрузки данных
  useEffect(() => {
    async function fetchProducts() {
      try {
        const res = await getProductsBySeller(seller.id);
        if (res.data.status === "success") {
          setProducts(res.data.products);
        } else {
          console.error("Ошибка получения продуктов",
res.data);
        }
      } catch (error) {
        console.error("Ошибка получения продуктов", error);
      }
    }
  });

```

```

    }
    if (seller) {
        fetchProducts();
    }
}, [seller]);

// Получаем отзывы продавца (для отображения в правой
колонке)
useEffect(() => {
    async function fetchReviews() {
        try {
            setLoadingReviews(true);
            const res = await getSellerReviews(seller.id);
            if (res.data && res.data.reviews) {
                setReviews(res.data.reviews);
            } else {
                console.error("Ошибка получения отзывов:",
res.data);
            }
        } catch (error) {
            console.error("Ошибка получения отзывов:", error);
        } finally {
            setLoadingReviews(false);
        }
    }
    if (seller) {
        fetchReviews();
    }
}, [seller]);

const handleDescriptionUpdate = async () => {
    try {
        const response = await updateSellerDescription({
description });
        if (response.data.status === "success") {
            setSeller({ ...seller, description });
        } else {
            console.error("Ошибка обновления описания");
        }
    } catch (error) {
        console.error("Ошибка запроса:", error);
    }
};

// Функция для удаления товара
const handleDeleteProduct = async (productId) => {
    if (!window.confirm("Удалить этот товар?")) return;
    try {
        const res = await deleteProduct(productId);
        if (res.data.status === "success") {

```

```

        setProducts((prev) => prev.filter((prod) => prod.id
!= productId));
    } else {
        console.error("Ошибка удаления товара", res.data);
    }
} catch (error) {
    console.error("Ошибка запроса на удаление товара:",
error.response?.data || error);
}
};

// Функция для удаления аккаунта продавца
const handleAccountDeletion = async () => {
    if (
        !window.confirm(
            "Вы уверены, что хотите удалить аккаунт продавца? Это
действие нельзя отменить."
        )
    )
        return;
    try {
        const res = await deleteSellerAccount();
        if (res.data.status === "success") {
            alert("Аккаунт продавца удалён");
            localStorage.removeItem("access_token");
            window.location.href = "/";
        } else {
            alert("Ошибка удаления аккаунта");
        }
    } catch (error) {
        console.error("Ошибка при удалении аккаунта продавца:",
error);
        alert("Ошибка при удалении аккаунта");
    }
};

// Функция для выхода из аккаунта
const handleLogout = () => {
    localStorage.removeItem("access_token");
    window.location.href = "/";
};

return (
    <div className="account-seller-page">
        <HeaderSeller />
        <div className="seller_page">
            {/* Левая колонка: логотип */}
            <div className="left-column">
                <SellerLogoUpload
                    initialImageUrl={seller?.main_image_url}

```

```

        onUpdate={ (newUrl) => setSeller({ ...seller,
main_image_url: newUrl })}
      />
    </div>

    { /* Центральная колонка: информация о продавце */ }
    <div className="center-column">
      <div className="seller-info">
        <h2>{seller?.production_name || "Название
компании"}</h2>
        <p>Email: {seller?.email}</p>
        <p>Телефон: {seller?.phone}</p>
        <p>Описание:</p>
        <textarea
          value={description}
          onChange={ (e) =>
setDescription(e.target.value)}
        />
        <button
onClick={handleDescriptionUpdate}>Обновить описание</button>
        <div className="address-management">
          <button onClick={ () =>
setIsAddressModalOpen(true)}>
            Управление адресами
          </button>
        </div>
        { /* Блок кнопок в правом нижнем углу центральной
колонки */ }

        <div className="account-actions">
          <button
            className="account-action-btn logout-btn"
            onClick={handleLogout}
          >
            Выйти
          </button>
          <button
            className="account-action-btn delete-account-
btn"
            onClick={handleAccountDeletion}
          >
            Удалить аккаунт
          </button>
        </div>
      </div>
    </div>

    { /* Правая колонка: окно с отзывами */ }
    <div className="right-column">
      <div className="reviews-block">
        <h3>Отзывы покупателей</h3>
        <div className="reviews-container">

```

```

        {loadingReviews ? (
          <p>Загрузка отзывов...</p>
        ) : reviews.length > 0 ? (
          reviews.map((review) => (
            <div key={review.id} className="review-
item">
              <p className="review-rating">★
{review.rating}</p>
              <p className="review-
comment">{review.comment}</p>
              <p className="review-buyer">–
{review.buyer_first_name}</p>
              <p className="review-date">
                {new
Date(review.created_at).toLocaleString()}
              </p>
            </div>
          ))
        ) : (
          <p>Нет отзывов</p>
        )}
      </div>
    </div>
  </div>
</div>

{/* Новый блок – жёлтый прямоугольник для товаров */}
<div className="products_section">
  <h3>Ваши товары</h3>
  <div className="products-grid">
    {/* Кнопка "добавить товар" как элемент сетки */}
    <div
      className="add-product-button"
      onClick={() => setIsProductModalOpen(true)}
    >
      <span className="plus-icon">+</span>
      <span className="button-text">добавить
товар</span>
    </div>
    {products.length > 0 ? (
      products.map((prod) => (
        <div key={prod.id} className="product-card">
          <button
            className="delete-product-btn"
            onClick={() =>
handleDeleteProduct(prod.id)}
          >
            &times;
          </button>
          <img

```

```

        src={prod.main_image_url}
        alt={prod.name}
        className="product-image"
      />
      <div className="product-info">
        <h4 className="product-
name">{prod.name}</h4>
        <p className="product-price">
          {prod.price} {prod.quantity_unit}
        </p>
        <p className="product-
count">x{prod.unit_count}</p>
      </div>
    </div>
  ))
) : (
  <p className="no-products">Нет товаров</p>
)
</div>
</div>

{isProductModalOpen && (
  <ProductModal onClose={() =>
setIsProductModalOpen(false)} />
)}
{isAddressModalOpen && seller && (
  <AddressModal sellerId={seller.id} onClose={() =>
setIsAddressModalOpen(false)} />
)}
</div>
);
}

export default AccountSeller;

```

Файл «Product.js»

```

import React, { useEffect, useState } from "react";
import { useParams, useNavigate } from "react-router-dom";
import { getNearbyProducts, getSellerAddresses,
getSellerPublicData } from "../services/api";
import "../styles/product.css";

const Product = () => {
  const { id } = useParams();
  const navigate = useNavigate();

  const [product, setProduct] = useState(null);
  const [addresses, setAddresses] = useState([]);
  const [sellerInfo, setSellerInfo] = useState(null);

```

```

const [loading, setLoading] = useState(true);
const [location, setLocation] = useState(null);
const [unitCount] = useState(1);
const radius = 100;

// Функция для форматирования единицы измерения
const formatQuantityUnit = (unit) => {
  switch (unit) {
    case "kg":
      return "кг.";
    case "units":
      return "шт.";
    case "liter":
      return "литр(а) (ов)";
    default:
      return unit;
  }
};

// Получаем текущее местоположение пользователя
useEffect(() => {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(
      ({ coords }) => {
        const lat = coords.latitude;
        const lon = coords.longitude;
        if (lat < -90 || lat > 90 || lon < -180 || lon >
180) {
          console.error("Некорректные координаты:", lat,
lon);
        } else {
          setLocation([lat, lon]);
        }
      },
      (err) => {
        console.warn("Ошибка определения местоположения:",
err.message);
        setLocation([55.755826, 37.6173]); // запасное
значение: Москва
      }
    );
  } else {
    setLocation([55.755826, 37.6173]);
  }
}, []);

// Получаем список товаров и ищем нужный товар по id
useEffect(() => {
  async function fetchProducts() {
    if (!location) return; // ждем, пока не получим
координаты

```

					ПГУ1.090301.14.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		73

```

        try {
            const [latitude, longitude] = location;
            console.log("Запрос товаров с параметрами:", {
latitude, longitude, radius });
            const res = await getNearbyProducts(latitude,
longitude, radius);
            if (res.data.status === "success" &&
res.data.products) {
                console.log("Список товаров, полученных с
сервера:", res.data.products);
                // Ищем товар с совпадающим id
                const foundProduct = res.data.products.find(item =>
String(item.id) === id);
                if (foundProduct) {
                    setProduct(foundProduct);
                } else {
                    alert("Товар не найден");
                    navigate("/catalog");
                }
            } else {
                alert("Ошибка получения списка товаров");
                navigate("/catalog");
            }
        } catch (error) {
            console.error("Ошибка получения данных товаров:",
error);

            alert("Ошибка получения данных товара");
            navigate("/catalog");
        } finally {
            setLoading(false);
        }
    }
    fetchProducts();
}, [id, location, navigate]);

// Получаем список адресов производителя по seller_id
useEffect(() => {
    async function fetchAddresses() {
        if (product && product.seller_id) {
            try {
                const res = await
getSellerAddresses(product.seller_id);
                if (res.data.status === "success" &&
res.data.addresses) {
                    setAddresses(res.data.addresses);
                } else {
                    console.error("Ошибка получения адресов
продавца:", res.data);
                }
            } catch (error) {

```

```

        console.error("Ошибка получения адресов продавца:",
error);
    }
    }
    }
    fetchAddresses();
}, [product]);

// Запрос для получения публичных данных продавца по
seller_id
useEffect(() => {
    async function fetchSellerPublicData() {
        if (product && product.seller_id) {
            try {
                const res = await
getSellerPublicData(product.seller_id);
                if (res.data.status === "success" &&
res.data.seller) {
                    setSellerInfo(res.data.seller);
                } else {
                    console.error("Ошибка получения публичных данных
продавца:", res.data);
                }
            } catch (error) {
                console.error("Ошибка получения публичных данных
продавца:", error);
            }
        }
    }
    fetchSellerPublicData();
}, [product]);

if (loading) {
    return <p>Загрузка...</p>;
}

if (!product) {
    return null;
}

return (
    <div className="product-container">
        <div className="product-block">
            {/* Верхний блок - фото товара и описание */}
            <div className="product-info">
                <div className="product-image-wrapper">
                    <img
                        src={product.main_image_url}
                        alt={product.name}
                        className="product-image"
                    />

```

```

        </div>
        <div className="product-details">
            <div className="product-header">
                <h1 className="product-
title">{product.name}</h1>
            </div>
            <p className="product-
description">{product.description}</p>
            <p className="product-price-large">
                {product.price}₽ / {unitCount}
                {formatQuantityUnit(product.quantity_unit)}
            </p>
            </div>
        </div>

        {/* Нижний блок - информация о продавце */}
        <div className="seller-info">
            <h2 className="seller-name">{sellerInfo ?
sellerInfo.name : product.seller_name}</h2>
            <p className="seller-rating">
                ★ {sellerInfo ? sellerInfo.rating :
(product.seller_rating || "0.0")}
            </p>

            {addresses.length > 0 && (
                <ul className="seller-addresses">
                    {addresses.map((addr) => {
                        const parts = addr.address.split(",");
                        const shortAddress = parts.slice(0,
2).join(",");

                        return (
                            <li key={addr.id}>
                                {shortAddress} <br />
                                <small>{addr.working_hours}</small>
                            </li>
                        );
                    })}
                </ul>
            )}

            <button
                className="seller-button"
                onClick={() =>
navigate(`/SellerProfile/${product.seller_id}`)}
            >
                Перейти на карточку продавца
            </button>
        </div>
    </div>

```

```

        </div>
    );
};
export default Product;

```

Файл «SellerProfile.js»

```

import React, { useState, useEffect } from "react";
import { useParams } from "react-router-dom";
import {
    getSellerPublicData,
    getProductsBySeller,
    getSellerAddresses,
    getSellerReviews,
    getBuyerData,
    deleteReview
} from "../services/api";
import ReviewModal from "../components/ReviewModal";
import "../styles/sellerProfile.css";

const formatQuantityUnit = (unit) => {
    switch (unit) {
        case "kg":
            return "кг.";
        case "units":
            return "шт.";
        case "liter":
            return "литр(а) (ов)";
        default:
            return unit;
    }
};

const SellerProfile = () => {
    const { sellerId } = useParams();

    const [seller, setSeller] = useState(null);
    const [products, setProducts] = useState([]);
    const [addresses, setAddresses] = useState([]);
    const [reviews, setReviews] = useState([]);
    const [loadingReviews, setLoadingReviews] = useState(true);
    const [isReviewModalOpen, setIsReviewModalOpen] =
    useState(false);
    const [currentBuyerId, setCurrentBuyerId] = useState(null);

    // Получаем публичные данные продавца
    useEffect(() => {
        const fetchSeller = async () => {
            try {
                const res = await getSellerPublicData(sellerId);

```

					ПГУ1.090301.14.001 ПЗ	Лист
						77
Изм.	Лист	№ докум.	Подпись	Дата		

```

        if (res.data.status === "success") {
            setSeller(res.data.seller);
        } else {
            console.error("Продавец не найден:", res.data);
        }
    } catch (error) {
        console.error("Ошибка получения публичных данных
продавца:", error);
    }
};
fetchSeller();
}, [sellerId]));

// Получаем товары продавца
useEffect(() => {
    const fetchProducts = async () => {
        try {
            const res = await getProductsBySeller(sellerId);
            if (res.data.status === "success") {
                setProducts(res.data.products);
            } else {
                console.error("Ошибка получения товаров продавца:",
res.data);
            }
        } catch (error) {
            console.error("Ошибка получения товаров продавца:",
error);
        }
    };
    fetchProducts();
}, [sellerId]);

// Получаем адреса продавца
useEffect(() => {
    const fetchAddresses = async () => {
        try {
            const res = await getSellerAddresses(sellerId);
            if (res.data.status === "success" &&
res.data.addresses) {
                setAddresses(res.data.addresses);
            } else {
                console.error("Ошибка получения адресов продавца:",
res.data);
            }
        } catch (error) {
            console.error("Ошибка получения адресов продавца:",
error);
        }
    };
    fetchAddresses();
}, [sellerId]);

```

```

// Получаем отзывы продавца
useEffect(() => {
  const fetchReviews = async () => {
    try {
      setLoadingReviews(true);
      const res = await getSellerReviews(sellerId);
      if (res.data && res.data.reviews) {
        setReviews(res.data.reviews);
      } else {
        console.error("Ошибка получения отзывов:",
res.data);
      }
    } catch (error) {
      console.error("Ошибка получения отзывов:", error);
    } finally {
      setLoadingReviews(false);
    }
  };
  fetchReviews();
}, [sellerId]);

// Получаем данные покупателя (если авторизован)
useEffect(() => {
  if (localStorage.getItem("access_token")) {
    const fetchBuyer = async () => {
      try {
        const res = await getBuyerData();
        if (res.data && res.data.buyer &&
res.data.buyer.id) {
          setCurrentBuyerId(res.data.buyer.id);
        } else if (res.data && res.data.id) {
          setCurrentBuyerId(res.data.id);
        }
        console.log("Current buyer id:", res.data.buyer ?
res.data.buyer.id : res.data.id);
      } catch (error) {
        console.error("Ошибка получения данных
покупателя:", error);
      }
    };
    fetchBuyer();
  }
}, []);

// Обработчик удаления отзыва
const handleDeleteReview = async (reviewId) => {
  if (!window.confirm("Удалить этот отзыв?")) return;
  try {
    const res = await deleteReview(reviewId);
    if (res.data.status === "success") {

```

```

        setReviews((prev) => prev.filter((r) => r.id !==
reviewId));
    } else {
        console.error("Ошибка удаления отзыва:", res.data);
    }
} catch (error) {
    console.error("Ошибка при удалении отзыва:", error);
}
};

if (!seller) {
    return <p>Загрузка...</p>;
}

return (
    <div className="seller-profile-page">
        <div className="seller-profile-container">
            {/* Центральная колонка: объединённая информация о
продавце */}
            <div className="center-column">
                <div className="seller-info-layout">
                    <div className="seller-logo-container">
                        <img
                            src={seller.main_image_url}
                            alt="Логотип продавца"
                            className="seller-logo"
                        />
                    </div>
                    <div className="seller-details">
                        <h2>{seller.production_name || "Название
компании"}</h2>
                        <p className="seller-desc">
                            Описание: {seller.description || "Описание не
добавлено"}
                        </p>
                        <p className="seller-rating-info">
                            Рейтинг: ★ {seller.rating || "0.0"}
                        </p>

                        {/* Список адресов */}
                        {addresses.length > 0 && (
                            <div className="addresses-list">
                                <h3>Адреса:</h3>
                                <ul>
                                    {addresses.map((addr) => (
                                        <li key={addr.id}>
                                            <span className="address-
flag"> 🚩 </span>
                                            <span className="address-
text">{addr.address}</span>

```

```

        <br />
        <small>{addr.working_hours}</small>
    </li>
    )})
</ul>
</div>
)}

{/* Список товаров продавца */}
<div className="products-list">
    <h3>Товары продавца:</h3>
    {products.length > 0 ? (
        <ul>
            {products.map((prod) => (
                <li key={prod.id}>
                    <img
                        src={prod.main_image_url}
                        alt={prod.name}
                        className="product-thumb"
                    />
                    <span className="product-info">
                        {prod.price}₽ / {1}
{formatQuantityUnit(prod.quantity_unit)}
                    </span>
                </li>
            )})
        </ul>
    ) : (
        <p>Нет товаров</p>
    )}
    </div>
</div>
</div>
</div>

{/* Правая колонка: блок с отзывами */}
<div className="right-column">
    <div className="reviews-block">
        <h2>Отзывы покупателей</h2>
        <p className="seller-rating-large">★
{seller.rating || "0.0"}</p>
        <button
            className="leave-review-btn"
            onClick={() => setIsReviewModalOpen(true)}
        >
            Оставить отзыв
        </button>
        <div className="reviews-container">
            {loadingReviews ? (
                <p>Загрузка отзывов...</p>

```

```

    ) : reviews.length > 0 ? (
      reviews.map((review) => (
        <div key={review.id} className="review-
item">
          {currentBuyerId &&
Number(review.buyer_id) === Number(currentBuyerId) && (
            <button
              className="delete-review-btn"
              onClick={() =>
handleDeleteReview(review.id)}
            >
              ×
            </button>
          )}
          <p className="review-rating">★
{review.rating}</p>
          <p className="review-
comment">{review.comment}</p>
          <p className="review-buyer">–
{review.buyer_first_name}</p>
          <p className="review-date">
            {new
Date(review.created_at).toLocaleString()}
          </p>
        </div>
      ))
    ) : (
      <p>Нет отзывов</p>
    )}
  </div>
</div>
</div>
</div>
{isReviewModalOpen && (
  <ReviewModal
    sellerId={seller.id}
    onClose={() => setIsReviewModalOpen(false)}
  />
)}
</div>
);
};

export default SellerProfile;

```

Файл «sellerProfile.css»

```

/* Общий контейнер страницы */
.seller-profile-page {
  padding: 20px;
}

```

					ПГУ1.090301.14.001 ПЗ	Лист
						82
Изм.	Лист	№ докум.	Подпись	Дата		

```

        background-color: #EFDECD; /* Тёплый бежевый фон */
        min-height: 100vh;
    }

    /* Основной контейнер для центральной и правой колонки */
    .seller-profile-container {
        display: flex;
        gap: 20px;
        padding: 20px;
    }

    .center-column {
        flex: 3;
        background-color: #fff;
        padding: 30px;
        border-radius: 15px;
        box-shadow: 0 4px 15px rgba(0, 0, 0, 0.1);
    }

    .seller-info-layout {
        display: flex;
        gap: 20px;
        align-items: flex-start;
    }

    /* Контейнер логотипа - размещается в крайнем левом углу */
    .seller-logo-container {
        flex-shrink: 0;
    }

    .seller-logo {
        width: 300px;
        height: 400px;
        object-fit: cover;
        border: 4px solid #EBD47E;
        border-radius: 25px;
    }

    /* Блок деталей о продавце */
    .seller-details {
        flex: 1;
        text-align: left;
    }

    .seller-details h2 {
        font-size: 32px;
        margin-bottom: 10px;
        color: #8F995D; /* Оливковый оттенок */
    }

    .seller-desc {
        font-size: 18px;
        color: #555;
        margin: 10px 0;
    }

```

```

}
.seller-rating-info {
    font-size: 20px;
    color: #E09113;
    margin: 5px 0;
}

/* Список адресов */
.addresses-list {
    margin-top: 20px;
}
.addresses-list h3 {
    margin-bottom: 10px;
    font-size: 20px;
    color: #8F995D;
}
.addresses-list ul {
    list-style: none;
    padding: 0;
}
.addresses-list li {
    margin-bottom: 10px;
    font-size: 16px;
}
.address-flag {
    margin-right: 5px;
}

.products-list {
    margin-top: 20px;
}
.products-list h3 {
    margin-bottom: 10px;
    font-size: 20px;
    color: #8F995D;
}
.products-list ul {
    list-style: none;
    padding: 0;
    display: flex;
    flex-wrap: wrap;
    gap: 15px;
}
.products-list li {
    display: flex;
    align-items: center;
    font-size: 16px;
    background: #F5F5F5;
    padding: 10px;
    border-radius: 10px;
    box-shadow: 0 2px 6px rgba(0, 0, 0, 0.1);
}

```

```

}
.product-thumb {
  width: 60px;
  height: 60px;
  object-fit: cover;
  border-radius: 8px;
  margin-right: 10px;
}

/* Правая колонка: блок с отзывами */
.right-column {
  flex: 1;
  display: flex;
  flex-direction: column;
  align-items: center;
  padding: 10px;
}
.reviews-block {
  background-color: #fff;
  padding: 20px;
  border-radius: 10px;
  width: 100%;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
  text-align: center;
}
.reviews-block h2 {
  font-size: 28px;
  margin-bottom: 10px;
  color: #8F995D;
}
}
.seller-rating-large {
  font-size: 36px;
  font-weight: bold;
  color: #E09113;
  margin: 10px 0;
}
}
.leave-review-btn {
  background-color: #E09113;
  color: #fff;
  padding: 10px 15px;
  border: none;
  border-radius: 5px;
  font-size: 16px;
  cursor: pointer;
  margin-bottom: 15px;
  transition: background-color 0.3s;
}
}
.leave-review-btn:hover {
  background-color: #d77c0e;
}
}
.reviews-container {

```

```

    max-height: 300px;
    overflow-y: hidden;
    border: 1px solid #ddd;
    padding: 10px;
    margin-top: 10px;
    border-radius: 5px;
    transition: overflow 0.3s ease;
}
.reviews-container:hover {
    overflow-y: auto;
}
.review-item {
    position: relative;
    padding: 10px;
    border-bottom: 1px dashed #ccc;
    margin-bottom: 10px;
}
.review-item:last-child {
    border-bottom: none;
}
.delete-review-btn {
    position: absolute;
    top: 5px;
    right: 5px;
    background: transparent;
    border: none;
    font-size: 18px;
    color: #d00;
    cursor: pointer;
}
.review-rating {
    font-weight: bold;
    color: gold;
    margin: 0;
    font-size: 18px;
}
.review-comment {
    margin: 8px 0;
    font-size: 16px;
}
.review-buyer {
    font-style: italic;
    font-size: 14px;
    margin: 0;
}
.review-date {
    font-size: 12px;
    color: #666;
    margin: 0;
}

```

```

/* Responsive */
@media (max-width: 768px) {
  .seller-profile-container {
    flex-direction: column;
  }
  .center-column, .right-column {
    flex: none;
    width: 100%;
  }
  .seller-info-layout {
    flex-direction: column;
    align-items: center;
  }
  .seller-logo {
    width: 100%;
    height: auto;
  }
  .products-list ul {
    flex-direction: column;
  }
}

```

Приложение В

(Обязательное)

Бизнес-проект. Электронный рынок фермерской продукции. Клиентская часть

Презентация

					<i>ПГУ1.090301.14.001 ПЗ</i>	Лист
						88
Изм.	Лист	№ докум.	Подпись	Дата		

Бизнес-проект. Электронный рынок фермерской продукции

Авторы работы: студенты гр.21ВВГ1 Митрошин А.Д. и Сущёв М.В.
Научные руководители: к.т.н. доцент кафедры ВТ Карамышева Н.С. и доцент кафедры ВТ Кузнецов А.В.

Цель и актуальность проекта

Задача этой дипломной работы— создать удобную и практичную вебплатформу, которая станет мощным инструментом для развития электронной коммерции в сфере сельского хозяйства. Она поможет цифровизировать сельское хозяйство и улучшить взаимодействие между фермерами и покупателями.

2



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.14.001 ПЗ

Лист

89

Краткая постановка задачи

Обеспечить удобные инструменты для управления ассортиментом и обратной связью, что улучшит взаимодействие и доверие между продавцами и потребителями.

Создать электронную платформу, которая позволит фермерам продавать продукцию напрямую, минуя посредников, что повысит доходы и доступность товаров для покупателей.

Разработать функционал для быстрого и удобного поиска фермерских продуктов

3

Обзор аналогов

Торговые рынки обеспечивают прямой контакт продавцов и покупателей, а также возможность сразу оценить качество и свежесть продуктов. Основные ограничения — сезонность и ограниченный ассортимент.

01

АгроМП — специализированная платформа для агропродукции и оборудования с функциями создания заявок и информацией по сельскому хозяйству. Минусы — платные, отсутствие самовывоза, отсутствие системы рейтинга и отзывов и ориентированность на опт.

03

02

Авито — крупнейшая платформа объявлений в России, удобна для локального поиска товаров с поддержкой чата и рейтингов. Недостаток — отсутствие специализированной карты фермеров.

04

Яндекс Маркет предлагает удобную доставку и рейтинги продавцов, но не поддерживает специализированный поиск фермерских товаров и ограничен в ассортименте свежей продукции.

4

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.14.001 ПЗ

Лист

90



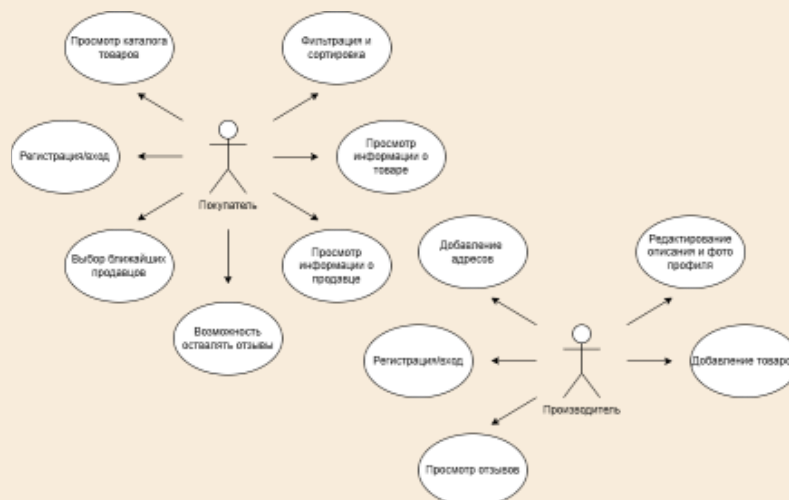
Описание функционала платформы

Платформа предлагает фермерам удобный интерфейс, который позволяет им размещать и управлять своим ассортиментом, а также отслеживать рейтинг и получать обратную связь. Платформа предоставляет удобный интерфейс для фермеров, позволяющий размещать и управлять своим ассортиментом. Отслеживать рейтинг и следить за обратной связью.

Клиенты могут ознакомиться с ассортиментом товаров, применить фильтр по категориям и найти ближайших продавцов на интерактивной карте. Кроме того, у покупателей есть возможность оставлять отзывы и оценивать продавцов. А покупатели могут просматривать каталог товаров, воспользоваться фильтром по категории, а так же искать на интерактивной карте ближайших продавцов. Покупатели так же могут оставлять отзывы и ставить рейтинги продавцам

5

Диаграмма прецедентов



6

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.14.001 ПЗ

Лист

91



Стек технологий

Клиентская часть

React позволяет создавать адаптивные интерфейсы с использованием компонентной архитектуры, что повышает эффективность и масштабируемость клиентской части приложения. Такой подход упрощает поддержку и развитие приложения.

React-компоненты отображают данные, полученные через REST API, и автоматически обновляются при изменении состояния, обеспечивая плавное взаимодействие с пользователем.

Серверная часть

Серверная часть приложения была разработана с применением языка программирования Python и микрофреймворка Flask. Flask выполняет функции маршрутизации HTTP-запросов, обработки бизнес-логики и взаимодействия с базой данных. API формирует и отправляет ответы в формате JSON, что обеспечивает лёгкую интеграцию с интерфейсом пользователя.

7

Диаграмма развертывания



8

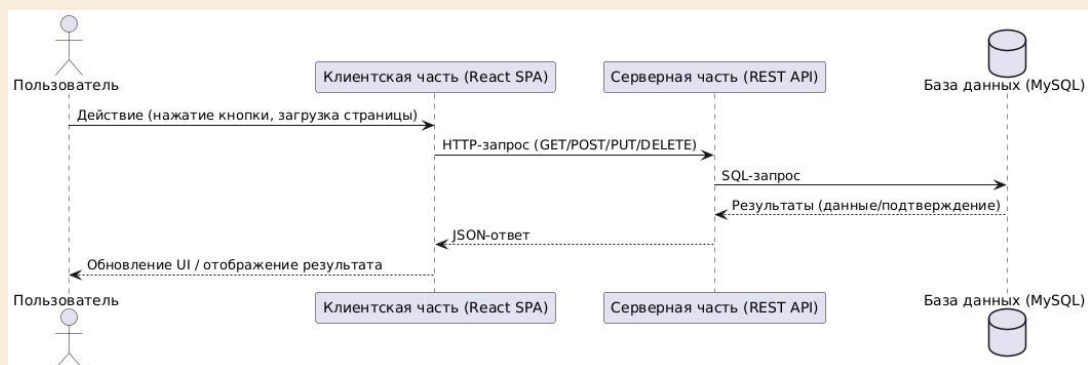
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.14.001 ПЗ

Лист

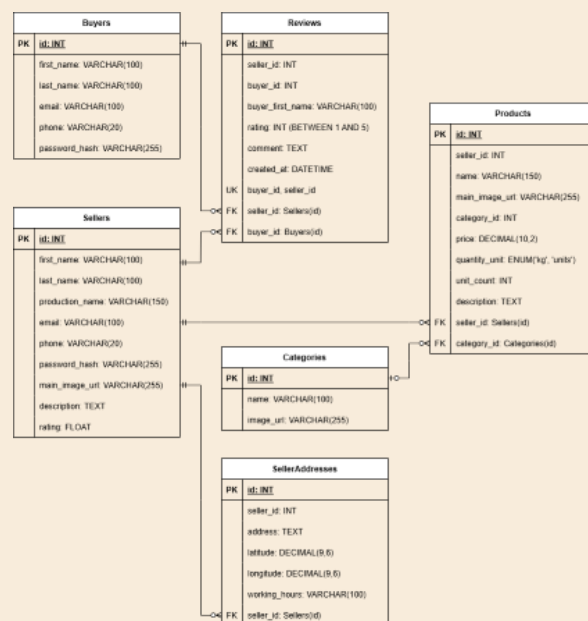
92

Диаграмма последовательности



9

ER-диаграмма



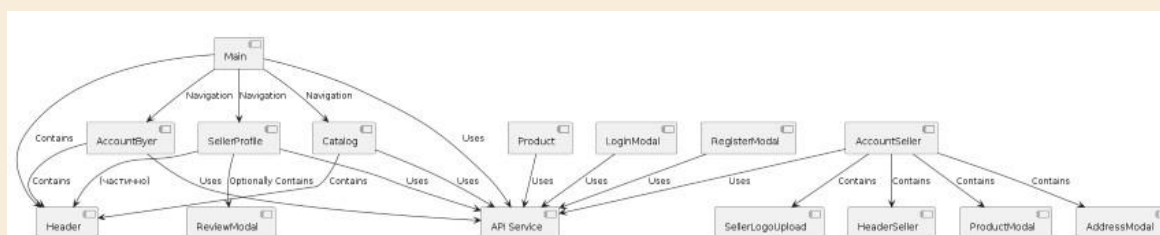
10

Диаграмма REST API эндпоинтов



11

Диаграмма компонентов React JS



12

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.14.001 ПЗ

Лист

94

Потоки доходов платформы

Эта диаграмма показывает, что основные доходы генерируются из рекламы, доставки и подписок, поддерживая финансовую устойчивость сервиса.

Разнообразие источников дохода снижает риски и способствует стабильности проекта, обеспечивая интерес как для фермеров, так и для инвесторов.



13

Каналы распространения платформы

01 Для роста платформы необходима активная рекламная кампания, направленная на привлечение новых пользователей, помимо технической поддержки и совершенствования приложения.

02 Партнёрские программы, например совместные акции с Министерством сельского хозяйства, укрепляют доверие фермеров и стимулируют участие новых производителей.

03 Основные инструменты продвижения включают SEO, контекстную рекламу и активное использование социальных сетей для максимального охвата целевой аудитории.

14

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.14.001 ПЗ

Лист

95

Расходы на разработку и запуск

Структура затрат по основным этапам разработки и внедрения электронной платформы.

Наибольшие затраты приходятся на backend разработку и frontend, что обусловлено необходимостью создания качественной и надежной системы.

Этап	Сумма (руб.)
Дизайн UX/UI и прототипирование	100 000
Frontend разработка	130 000
Backend разработка	150 000
Настройка серверов	70 000
Тестирование и отладка	60 000

15

Ежемесячные расходы на обслуживание

1

Хостинг и поддержка серверов требуют значительных вложений для обеспечения стабильной работы платформы и обработки пользовательских запросов без задержек.

2

Регулярное обновление и развитие функционала платформы обеспечивают соответствие современным требованиям рынка и повышают удобство использования для фермеров и покупателей.

3

Модерация контента направлена на поддержание качества и достоверности информации, а также оперативное разрешение возможных конфликтов между участниками торгов.

4

Маркетинговые мероприятия, в том числе реклама и продвижение в цифровых каналах, способствуют привлечению новых пользователей и увеличению объема продаж на площадке.

16

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.14.001 ПЗ

Лист

96

Расчет срока окупаемости проекта

Рост чистой прибыли отражает успешное развитие платформы и стабилизацию доходов на фоне снижения затрат.

Срок окупаемости составляет около пяти месяцев, после чего проект начинает приносить прибыль, обеспечивая финансовую устойчивость.



При расчете на 5000 активных пользователей

17

Итоги разработки платформы

Создана функциональная веб-платформа со зрелым интерфейсом, обеспечивающим интуитивное взаимодействие пользователей с каталогом.

Платформа оптимизирует коммуникацию между фермерами и конечными потребителями, сокращая количество посредников и повышая прозрачность сделок.

Поддержка локальных производителей усиливается через механизм маркетинга и аналитики, что способствует развитию сельского хозяйства на региональном уровне.

18

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.14.001 ПЗ

Лист

97



Поддержка локальных производителей

Площадка предоставляет отечественным фермерам доступ к широкой базе клиентов, расширяя возможности реализации продукции и укрепляя финансовую устойчивость хозяйств.

Наша платформа поможет производителям увеличивать базу своих клиентов и развивать бизнес в долгосрочной перспективе.

19

Заключение и перспективы развития

Проект успешно воплощён в жизнь, открывая новые перспективы для аграриев и потребителей.

В перспективе можно создать мобильное приложение для iOS и Android, интегрировать платформу с платёжными системами, разработать чат и внедрить систему рекомендаций, основанную на предпочтениях пользователей. Также разработка программы лояльности позволит привлечь новых клиентов.

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.14.001 ПЗ

Лист

98