Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра информационных и управляющих систем
Направление подготовки 09.04.04 Программная инженерия
Направленность (профиль) образовательной программы Управление разработкой программного обеспечения

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой
_____ А.В. Бушманов
«___» _____ 2025 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему: Проектирование и реализация веб-платформы Neighbor Flowers с использованием ASP.Net и Vue.js

Исполнитель студент группы 3105-ом2	_____ (подпись, дата)	Е Сухао
Руководитель доцент, канд. техн. наук	_____ (подпись, дата)	Т.А. Галаган
Руководитель научного содержания программы магистратуры профессор, доктор техн. наук	_____ (подпись, дата)	И.Е. Ерёмин
Нормоконтроль инженер кафедры	_____ (подпись, дата)	В.Н. Адаменко
Рецензент доцент, канд. техн. наук	_____ (подпись, дата)	С.Г. Самохвалова

Благовещенск, 2025

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра информационных и управляющих систем

УТВЕРЖДАЮ
Зав. кафедрой
_____ А.В. Бушманов
«___» _____ 2025 г.

З А Д А Н И Е

К магистерской диссертации студента группы 3105-ом2 Е Сухао

1. Тема магистерской диссертации: Проектирование и реализация веб-платформы Neighbor Flowers с использованием ASP.Net и Vue.js

(Утверждено приказом от 06.03.2025 № 609-уч)

2. Срок сдачи студентом законченной работы (проекта): 10.06.2025
3. Исходные данные к магистерской диссертации: документация разработчиков, интернет-ресурсы, учебная литература, отчёт по практической подготовке
4. Содержание магистерской диссертации (перечень подлежащих разработке вопросов): проектирование алгоритма решения задачи, разработка базы данных
5. Перечень материалов приложения (наличие чертежей, таблиц, графиков, схем, программных продуктов, иллюстративного материла и т.п.): нет
6. Рецензент магистерской диссертации: Самохвалова С.Г., доцент, кандидат технических наук
7. Дата выдачи задания 17.01.25
8. Руководитель выпускной квалификационной работы : Т.А. Галаган, доцент, канд. техн. наук

(фамилия, имя, отчество, должность, уч.степень, уч.звание)

Заявление принял к исполнению _____

РЕФЕРАТ

Магистерская диссертация содержит 83 страницы, 30 рисунков, 44 источника, 23 таблицы

ВЕБ-ПЛАТФОРМА, ЦВЕТОЧНЫЙ МАГАЗИН, ASP.NET, VUE.JS, ИНФОРМАЦИОННАЯ СИСТЕМА, АВТОМАТИЗАЦИЯ

Целью работы является проектирование и реализация веб-платформы для продажи цветов с использованием ASP.Net и Vue.js.

В качестве технологий разработки использовались ASP.Net для серверной части и Vue.js для клиентского интерфейса.

Разрабатываемое программное обеспечение представляет собой информационную систему, включающую модули регистрации и авторизации пользователей, каталог товаров, корзину покупок, оформление заказов, модуль отзывов.

Задачи:

- изучение предметной области и детализация объекта исследования;
- обеспечение разнообразного выбора товаров;
- удовлетворение индивидуальных потребностей покупателей;
- автоматизация системы продаж.

Разработанная платформа Neighbor Flowers ориентирована на удовлетворение актуальных требований малого и среднего бизнеса в сфере электронной коммерции. Практическая значимость проекта заключается в возможности его внедрения для цифровизации процессов продаж и повышения качества клиентского обслуживания.

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

БД	база данных
ИБ	информационная безопасность
ИС	информационная система
НФ	нормальная форма
ООП	объектно-ориентированное программирование
ОС	операционная система
ПО	программное обеспечение
СУБД	система управления базами данных
API	программный интерфейс приложения
CRM	управление взаимоотношениями с клиентами
REST	передача состояния представления
SOA	сервис-ориентированная архитектура
SSL	уровень защищенных сокетов
TLS	безопасность на транспортном уровне

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На современном этапе развития цифровых технологий наблюдается стремительное распространение онлайн-торговли в сегменте цветочного магазина. Электронная коммерция оказывает значительное влияние на поведение потребителей, позволяя им быстро находить и приобретать необходимые товары в удобное время, а компаниям – оперативно реагировать на спрос, управлять ассортиментом и взаимодействовать с клиентами.

Актуальность данного исследования обусловлена необходимостью создания удобных, надежных и функциональных веб-платформ, ориентированных на потребности пользователей и способных обеспечить эффективное управление продажами. В условиях высокой конкуренции и роста требований к качеству онлайн-сервисов особую важность приобретают удобство интерфейса, производительность системы, возможности анализа продаж и поддержки пользователей.

Платформа Neighbor Flowers разрабатывается как специализированная веб-система, объединяющая функции интернет-магазина с возможностями администрирования, статистического анализа и взаимодействия с пользователями. Для реализации проекта выбраны современные технологические решения: фреймворк ASP.Net на стороне сервера и библиотека Vue.js для построения динамичного клиентского интерфейса. Такой технологический стек позволяет обеспечить модульность, масштабируемость и высокую производительность приложения.

Целью является проектирование и реализация веб-платформы для продажи цветов с использованием ASP.Net и Vue.js

Для достижения цели необходимо решить следующие задачи:

- изучение предметной области и детализация объекта исследования;
- обеспечение разнообразного выбора товаров;
- удовлетворение индивидуальных потребностей покупателей;
- автоматизация системы продаж.

1 ПРЕДМЕТНАЯ ОБЛАСТЬ ИССЛЕДОВАНИЯ

1.1 Описание характерных особенностей предмета исследования

В эпоху быстрого развития компьютерных сетей в Интернете появляются различные компьютерные программы. Люди все чаще обнаруживают, что покупать товары в Интернете гораздо удобнее, чем ходить в обычные магазины. Затраты продавцов на получение и обработку заказов также возрастают гораздо ниже, чем при открытии магазина. Стоимость физического магазина.

Веб-сайт цветочного магазина имеет основные функции часто используемых веб-сайтов продаж, такие как просмотр продуктов, вход и регистрация пользователя, онлайн-покупки, расчеты, управление внутренней базой данных и т. д. Эти функции можно использовать для лучшего управления информацией о продажах цветов.

Пользователи, использующие этот веб-сайт, в основном делятся на две роли:

- клиенты, имеющие возможность просматривать, покупать цветы, изменять информацию в корзине покупок, запрашивать заказы и оценивать продукты;

- менеджеры по продажам имеющие возможность просматривать информацию о клиентах.

Система должна обладать следующими основными функциями. Для клиентов она должна предоставлять возможность регистрации и авторизации, просмотра доступных товаров (цветов и букетов), использования функции поиска для удобного нахождения нужной продукции, добавления товаров в корзину, оформления заказов с указанием способа доставки и оплаты, а также предоставления отзывов и оценок на купленные товары.

Для менеджеров система должна обеспечивать функционал управления ассортиментом, включая добавление, редактирование и удаление товаров, просмотр заказов клиентов и их статуса, управление информацией о клиентах, а

также предоставление аналитических данных, таких как отчеты о продажах и востребованных товарах.

Кроме того, система должна включать возможности администрирования, такие как управление учетными записями пользователей, настройка доступов и резервное копирование данных. Важной частью функционала является обеспечение безопасности данных и конфиденциальности информации клиентов, а также стабильная работа системы при высокой нагрузке.

С быстрым развитием общества люди продолжают уделять больше внимания духовным потребностям, их вкус к жизни растет с каждым днем. Сформировалась тенденция превращения цветов в предмет повседневного потребления. Однако из-за быстрого развития информационных технологий традиционные цветочные магазины также столкнулись с беспрецедентными проблемами. Рост онлайн-продаж цветов еще больше удовлетворил потребности потребителей. Люди могут легко совершить покупку цветов через Интернет. Поэтому необходимо разработать сайт цветочного магазина на базе ASP.Net+Vue для удовлетворения повседневных потребностей людей.

Со временем время и стоимость сделки можно существенно сократить, воспользовавшись интернет-магазином продажи и покупки цветов. В современном мире уровень жизни и потребности людей постепенно меняются, и все больше и больше людей надеются удовлетворить свои потребности, покупая инновационные товары. Во многих случаях, таких как поздравления с Новым годом, свадьбами и днями рождения, цветы играют жизненно важную роль. Хотя может быть трудно найти подходящий цветочный магазин, с развитием современных социальных технологий сеть онлайн-маркетинга цветов создала большое удобство для клиентов, а также создала больше возможностей для торговцев, сокращая традиционные каналы. Это значительно снижает эксплуатационные расходы. улучшает покупательский опыт клиентов и оказывает положительное влияние на социальное развитие.

Для обеспечения системы необходимыми функциями следует выполнить следующие шаги:

Проектирование архитектуры системы – выделение клиентской и серверной части (frontend и backend), проектирование базы данных для хранения информации о пользователях, товарах, заказах и отзывах

Реализация пользовательского интерфейса (frontend) – использование Vue.js для создания удобного и интуитивно понятного интерфейса, разработка страницы регистрации, авторизации, просмотра товаров, корзины и оформления заказа, реализация адаптивного дизайна для поддержки разных устройств.

Реализация серверной логики (backend) – использование ASP.Net для обработки запросов от клиентов, управления бизнес-логикой и взаимодействия с базой данных, реализация RESTful API для связи между клиентской и серверной частями.

Проектирование базы данных – использование SQL Server для хранения структурированных данных, создание таблицы для товаров, пользователей, заказов, отзывов, корзины и статистики, обеспечение целостности данных с помощью связей между таблицами и ограничений.

Обеспечение безопасности – реализация защиты данных клиентов через SSL/TLS для шифрования трафика, использование механизмов авторизации и аутентификации, таких как JWT (JSON Web Tokens) или OAuth, обеспечение регулярного резервного копирования данных.

Функционал администрирования – создание интерфейса для менеджеров, позволяющего управлять товарами, заказами и клиентскими данными, реализация инструментов для аналитики, таких как отчеты о продажах и статистику посещений.

Тестирование и оптимизация – проведение функционального и нагрузочного тестирования, чтобы убедиться в корректной работе системы, оптимизировать производительность базы данных и кода.

Интеграция дополнительных возможностей – подключение платежных систем для онлайн-оплаты, реализация уведомлений через email или SMS для подтверждения заказов и обновлений статусов.

1.2 Анализ современного состояния проблемы исследования

1.2.1 Современные технологии разработки интернет-магазинов

Разработка интернет-магазинов как одного из ключевых компонентов электронной коммерции требует комплексного подхода, включающего выбор архитектуры, реализацию безопасной аутентификации, интеграцию с платёжными и логистическими системами, а также обеспечение высокой производительности и масштабируемости.

В современной практике используются следующие технологические решения:

- клиент-серверная архитектура на основе REST или GraphQL API, обеспечивающая разделение фронтенда и бэкенда;

- фронтенд-фреймворки, такие как Vue.js, React и Angular, которые позволяют создавать отзывчивые, динамичные и удобные пользовательские интерфейсы;

- бэкенд-платформы, включая ASP.Net Core, Django, Node.js и Laravel, обеспечивающие высокую надёжность, защиту и удобную работу с базами данных;

- системы управления базами данных как реляционные (PostgreSQL, MySQL), так и нереляционные (MongoDB) применяются для хранения информации о продуктах, пользователях и заказах;

- интеграция с внешними сервисами – платёжные системы (ЮKassa, Tinkoff, Сбербанк API), службы доставки (СДЭК, Boxberry), а также CRM-системы;

- контейнеризация и DevOps-подходы – использование Docker, CI/CD-инструментов (GitHub Actions, GitLab CI) и облачных сервисов (например, Yandex Cloud, VK Cloud Solutions) позволяет обеспечивать надёжную доставку обновлений и масштабируемость проекта;

- инструменты аналитики и мониторинга (например, Яндекс.Метрика, Google Analytics, Sentry) позволяют анализировать поведение пользователей и вовремя устранять ошибки.

Особое внимание в последние годы уделяется обеспечению информационной безопасности, быстрой загрузке страниц, адаптивному дизайну, а также применению технологий персонализации от рекомендаций товаров до таргетированной рекламы.

Таким образом, современные технологии веб-разработки обеспечивают мощный инструментарий для создания эффективных и устойчивых онлайн-магазинов, включая цветочные платформы, с широкими возможностями кастомизации, масштабирования и взаимодействия с пользователем.

1.2.2 Развитие интернет-коммерции

Рынок интернет-торговли, включая онлайн-продажи цветов, активно развивается, особенно в крупных городах. За последние годы растет популярность онлайн-магазинов цветов, благодаря их удобству и возможности быстрой доставки.

Компании уделяют внимание следующим аспектам:

- технологии и платформы, наиболее популярными инструментами для создания сайтов являются платформы с открытым исходным кодом (например, 1С-Битрикс, OpenCart), а также использование фреймворков, таких как Laravel или Django. Активно используются интеграции с банковскими системами и агрегаторами доставки, такими как СДЭК и Boxberry;

- локализация и сезонность, отмечается сильная зависимость от сезонных факторов (например, праздники 8 Марта, День учителя). Поэтому разработки включают анализ сезонного спроса и специальные предложения. Интеграция с локальными курьерскими службами для обеспечения своевременной доставки является важной составляющей;

- развитие омниканальности, важное направление исследований связано с омниканальными решениями, которые позволяют клиентам переключаться между веб-сайтом, мобильными приложениями и офлайн-магазинами, сохраняя единый опыт покупки. Примеры российских онлайн-магазинов цветов, таких как Flowwow и UFL, демонстрируют успешные решения по объединению платформ;

- инновации в клиентском опыте, в России появляются платформы, предлагающие нестандартные услуги, такие как подписка на цветы или виртуальные букеты.

Основные направления исследований:

- персонализация и искусственный интеллект, использование AI для создания персонализированных рекомендаций на основе данных о предыдущих покупках и интересах клиентов. Применение чат-ботов для автоматизации общения с клиентами и поддержки на всех этапах покупки;

- мобильные приложения, во многих странах акцент сделан на развитие мобильных решений, позволяющих совершать покупки в несколько кликов;

- экологичность и устойчивость, исследования направлены на внедрение экологичных методов упаковки и поставок. Например, во многих странах популярны решения, минимизирующие использование пластика и предлагающие букеты из локально выращенных цветов;

- быстрая доставка, компании за рубежом активно используют дроны и автоматизированные системы доставки для сокращения времени доставки;

- интеграция с другими сервисами, когда цветочные магазины объединяются с другими услугами, такими как подарочные корзины, шоколад или доставка напитков.

Таким образом, исследования развиваются в сторону улучшения клиентского опыта, автоматизации процессов, а также экологизации и персонализации сервисов. Эти тенденции создают возможности для внедрения инноваций в разработку интернет-магазинов.

1.3 Обзор существующих решений задачи

Детальный и полный анализ требований, общая структура проекта обеспечивает скорость разработки системы, поэтому хорошее построение системы проекта и технико-экономический анализ имеют важное значение для проекта. В этой статье подробно разбираются функциональные и нефункциональные требования веб-сайта, а также подробно анализируется каждая часть.

Представляем ключевое программное обеспечение и технологии, использованные при разработке сайта Neighbor Flowers:

- основанная на платформе веб-разработки ASP.NET, которая может предоставить разработчикам богатые модели программирования, сервисы для создания мощного сетевого приложения и программное обеспечение;

- разработанная Microsoft СУБД SQL Server и представляющая собой мощный, простой в использовании и гибкий веб-сайт управления базами данных, на котором можно обеспечить хорошую интеграцию данных и реализовать множество приложений;

- полный инструмент разработки Microsoft Visual Studio, который может помочь разработчикам быстро и эффективно реализовать различные функции.

- поддерживающий различные соединения, включая MySQL, Oracle Navicat. Его функции мощные и могут удовлетворить различные потребности профессиональных разработчиков, а также его очень легко использовать новичкам. Пользовательский интерфейс Navicat имеет изысканный дизайн, позволяющий легко реализовать безопасные, эффективные и удобные функции создания, управления, доступа и обмена информацией.

Таким образом, важно провести детальный анализ функциональных и нефункциональных требований, чтобы понять нужды пользователей и обеспечить соответствие конечного продукта ожиданиям.

Существующие решения позволяют легко адаптироваться к изменяющимся требованиям проекта и обеспечивают возможность интеграции новых функций по мере необходимости.

В целом, выбор правильных технологий и инструментов в сочетании с тщательным планированием и анализом требований является ключевым для успешной реализации веб-платформы Neighbor Flowers.

1.4 Цель и задачи

Цель задаёт общий вектор развития проекта и позволяет сфокусироваться на главном результате, которого нужно достичь. Это помогает избежать распыления усилий и сосредоточиться на ключевых аспектах.

Задачи позволяют разбить цель на конкретные шаги, которые необходимо выполнить. Это делает процесс разработки управляемым и последовательным.

Чётко сформулированные цели и задачи позволяют оценить результат: достигнута ли цель, выполнены ли все задачи, насколько успешно решены поставленные проблемы.

Чётко сформулированные цели и задачи позволяют оценить результат достигнута ли цель, выполнены ли все задачи, насколько успешно решены поставленные проблемы.

Целью данной работы является проектирование и реализация веб-платформы для продажи цветов с использованием технологий ASP.Net и Vue.js, обеспечивающей удобную навигацию, широкое разнообразие товаров и эффективную автоматизацию процессов покупки и управления заказами.

Для достижения поставленной цели необходимо решить следующие задачи:

- провести анализ предметной области и уточнить объект исследования, выявив современные требования к онлайн-торговле цветами;

- обеспечить возможность широкого выбора товаров с индивидуальным подбором композиций под вкусы и предпочтения клиентов;

- реализовать инструменты, ориентированные на удовлетворение индивидуальных потребностей покупателей, включая поиск, фильтрацию и персонализацию;

- автоматизировать ключевые процессы системы продаж от оформления заказа до управления клиентской и товарной информацией.

Таким образом, цель и задачи являются неотъемлемой частью научной и проектной документации. Они помогают другим участникам (научным руководителям, экспертам, заказчикам) лучше понять суть проекта и подход к его

выполнению. Эти задачи направлены на создание эффективного инструмента, который упростит процесс продажи и покупки цветов, а также обеспечит удобство работы для всех категорий пользователей.

1.5 Описание методов и инструментов разработки

Создание веб-сайта Neighbor Flowers требует использования современных методов проектирования и разработки, а также соответствующих инструментов, обеспечивающих надежность, производительность и удобство использования.

1.5.1 Методы разработки

Для реализации проекта применяются следующие методы:

- модульный подход, проект разделяется на отдельные функциональные модули (например, регистрация, каталог товаров, корзина покупок). Такой подход облегчает тестирование и внесение изменений;

- методология Agile, проект реализуется итерационно, с возможностью оперативного внесения изменений на основе обратной связи. Подход подходит для работы с динамичными требованиями и позволяет реализовывать функционал поэтапно;

- архитектура REST, используется для взаимодействия клиентской и серверной частей. Реализация REST API обеспечивает масштабируемость системы и возможность интеграции с внешними сервисами;

- методы обеспечения безопасности, использование шифрования данных (SSL/TLS) для защиты пользовательской информации. Внедрение механизмов проверки подлинности и авторизации для защиты от несанкционированного доступа;

- объектно-ориентированное программирование (ООП), этот метод позволяет разбить проект на логические модули и компоненты, упрощая их разработку и последующее сопровождение. Используется для реализации основных функций веб-сайта, таких как управление заказами, данными клиентов и каталогом продуктов.

На рисунке 1 представлены основные принципы ООП.

Рисунок 1 – Основные принципы ООП

Основные принципы объектно-ориентированного программирования (ООП) представляют собой подход к разработке программ, в котором все компоненты рассматриваются как объекты с определенными свойствами и поведением. Эти принципы обеспечивают модульность, повторное использование кода и гибкость систем.

1.5.2 Средства разработки

Для реализации проекта используются современные инструменты, обеспечивающие высокое качество разработки и удобство работы.

Средства для серверной части:

- платформа для создания высокопроизводительных и кроссплатформенных веб-приложений ASP.Net Core. Позволяет реализовать серверную логику, включая обработку запросов, работу с базой данных и безопасность;

- реляционная база данных для хранения информации о продуктах, заказах и пользователях SQL Server. Обеспечивает интеграцию с ASP.Net Core и высокую производительность.

Средства для клиентской части:

- фреймворк для создания динамичных и интерактивных пользовательских интерфейсов Vue.js. Его модульность и простота позволяют эффективно разрабатывать компоненты веб-приложения;

- CSS-фреймворк для адаптивного дизайна Bootstrap, обеспечивающий удобное отображение сайта на различных устройствах.

Инструменты для разработки и тестирования:

- интегрированная среда разработки (IDE) Microsoft Visual Studio, поддерживающая ASP.Net Core, позволяющая эффективно разрабатывать, отлаживать и тестировать приложение;

- инструмент для тестирования REST API Postman, необходимый для проверки корректности взаимодействия между клиентской и серверной частями.

Средства управления базами данных:

- удобный инструмент для управления базой данных SQL Server Navicat, обеспечивающий простоту создания, редактирования и анализа данных.

Средства обеспечения совместной работы и контроля версий:

- система контроля версий для управления изменениями в коде проекта Git;

- платформа для хранения кода, командной работы и интеграции с другими инструментами разработки GitHub.

Средства для тестирования производительности и безопасности:

- инструмент для проверки производительности и нагрузки на веб-сайт JMeter;

- средство для тестирования безопасности веб-приложений и выявления уязвимостей OWASP ZAP.

Эти методы и инструменты позволят создать удобный, производительный и безопасный веб-сайт, удовлетворяющий современным требованиям электронной коммерции.

1.5.3 Объектно-ориентированное программирование

Инкапсуляция заключается в объединении данных (свойств) и методов (функций) для работы с ними в одном объекте. Доступ к данным осуществляется через методы, что позволяет контролировать их использование.

Преимущества:

- защищает данные от несанкционированного доступа;

- упрощает изменение внутренней реализации объекта без влияния на другие части системы.

Пример использования инкапсуляции представлен на рисунке 2.

```
public class Flower {
    private string name;  // Закрытое поле
    public string GetName() {
        return name;  // Доступ через метод
    }
    public void SetName(string newName) {
        name = newName;
    }
}
```

Рисунок 2 – Использование инкапсуляции

Наследование позволяет создавать новые классы на основе существующих, заимствуя их свойства и методы. Это помогает избегать дублирования кода.

Пример использования наследования представлен на рисунке 3.

```
public class Product {
    public string Name { get; set; }
    public decimal Price { get; set; }
}
public class Flower : Product {
    public string Color { get; set; }
}
```

Рисунок 3 – Использование наследования

Преимущества:

- повторное использование кода;

- упрощение поддержки и расширения системы.

Полиморфизм позволяет объектам одного типа обрабатывать данные по-разному в зависимости от конкретной реализации. Это достигается через перегрузку методов и их переопределение в наследуемых классах.

Пример использования полиморфизма представлен на рисунке 4.

```
public class Product {
    public virtual void DisplayInfo() {
        Console.WriteLine("Product information");
    }
}
public class Flower : Product {
    public override void DisplayInfo() {
        Console.WriteLine("Flower information");
    }
}
```

Рисунок 4 – Использование полиморфизма

Абстракция фокусируется на определении только тех характеристик объекта, которые имеют значение для конкретной задачи, игнорируя второстепенные детали.

Пример использования абстракции представлен на рисунке 5.

```
public abstract class Product {
    public string Name { get; set; }
    public abstract void DisplayInfo(); // Метод без реализации
}
public class Flower : Product {
    public override void DisplayInfo() {
        Console.WriteLine("Flower information");
    }
}
```

Рисунок 5 – Использование абстракции

Преимущества:

- упрощает проектирование системы;

- сокращает количество избыточной информации, с которой работает разработчик.

Таким образом, эти четыре принципа лежат в основе ООП и позволяют создавать системы, которые легко поддерживать, масштабировать и модифицировать.

1.6 Обзор существующих методов решения аналогичных задач

Для создания веб-сайтов электронной коммерции, подобных Neighbor Flowers, уже существуют различные подходы и методы разработки. Эти методы варьируются в зависимости от технологий, используемых инструментов и требований к функциональности. В данном разделе показаны основные существующие методы и их особенности.

Сравнение основных методов решения аналогичных задач представлено в таблице 1.

Таблица 1 – Сравнение методов решения

Критерий	Готовые платформы (Shopify, WooCommerce)	CMS (Joomla, Drupal)	Индивидуальная разработка (ASP.Net, Vue.js)	Облачные технологии (AWS, Azure)
1	2	3	4	5
Время разработки	Короткое	Среднее	Долгое	Зависят от используемой технологии
Стоимость	Низкая или средняя	Средняя	Высокая	Платежи зависят от использования ресурсов
Гибкость	Ограниченная	Средняя	Максимальная	Высокая
Техническая сложность	Низкая	Средняя	Высокая	Высокая
Масштабируемость	Ограниченная	Средняя	Высокая	Очень высокая
Функциональность	Стандартная	Расширяемая через модули	Полностью настраиваемая	Расширяемая

Продолжение таблицы 1

Зависимость от платформы	Высокая	Средняя	Низкая	Высокая
Поддержка и обслуживание	Включены в сервис	Требуется дополнительная работа	Зависит от команды разработки	Включена в облачный сервис
Подходит для новичков	Да	В определенной степени	Нет	Нет
Использование современных технологий	Среднее	Среднее	Высокое	Высокое

Вывод:

- готовые платформы подходят для быстрого запуска небольших магазинов;

- CMS предоставляют больше возможностей для настройки, но требуют большего времени и опыта;

- индивидуальная разработка оптимальна для сложных и уникальных решений, но требует больше ресурсов;

- облачные технологии усиливают любой из подходов за счет высокой масштабируемости и надежности.

Для Neighbor Flowers выбрана индивидуальная разработка на базе ASP.Net и Vue.js, что обеспечивает максимальную гибкость и позволяет реализовать уникальный функционал.

1.6.1 Использование готовых платформ электронной коммерции

На сегодняшний день существует множество готовых платформ для создания онлайн-магазинов.

Преимущества:

- быстрое развертывание без необходимости глубокого погружения в программирование;

- наличие встроенных функций: управление каталогом, корзина, оформление заказов, обработка платежей;

- возможность масштабирования и интеграции с другими сервисами.

Популярные платформы:

- простая в использовании платформа с широким выбором интеграций и шаблонов Shopify;

- плагин для WordPress, который позволяет добавлять функции интернет-магазина к существующему сайту WooCommerce;

- открытая система управления интернет-магазинами, которая поддерживает гибкую настройку и разработку дополнительных модулей OpenCart.

Ограничения:

- меньшая гибкость в реализации уникальных функций;

- зависимость от ограничений платформы и лицензирования.

1.6.2 Разработка с использованием готовых CMS

Системы управления контентом (CMS) предоставляют удобный способ создания веб-сайтов, включая интернет-магазины.

Популярные CMS:

- обеспечивает базовые функции интернет-магазина и поддерживает кастомизацию Joomla с VirtueMart;

- мощная система для разработки сложных сайтов с возможностью интеграции интернет-магазина Drupal с модулем Commerce.

Преимущества:

- более гибкая настройка, чем на готовых платформах;

- поддержка множества плагинов и модулей для расширения функционала.

Ограничения:

- необходимость дополнительных знаний для настройки;

- ограниченная производительность при обработке большого объема данных.

1.6.3 Индивидуальная разработка с использованием современных фреймворков

Метод индивидуальной разработки обеспечивает максимальную гибкость и позволяет адаптировать систему под конкретные нужды бизнеса.

Технологии серверной части:

- предоставляет широкие возможности для создания мощных и безопасных веб-приложений ASP.Net;

- фреймворк с богатым набором инструментов для ускорения разработки Django (Python);

- используется для разработки высокопроизводительных приложений, работающих в режиме реального времени Node.js.

Технологии клиентской части:

- легкий фреймворк для создания динамических пользовательских интерфейсов Vue.js;

- популярная библиотека для разработки одностраничных приложений React.js;

- комплексное решение для построения масштабируемых приложений Angular.

Преимущества:

- полная свобода в выборе архитектуры и реализации функций;

- возможность оптимизации производительности и удобства использования.

Ограничения:

- более высокие временные и финансовые затраты на разработку;

- необходимость наличия квалифицированной команды разработчиков.

1.6.4 Использование облачных технологий

Облачные сервисы предлагают гибкие решения для размещения и поддержки веб-приложений.

Примеры:

- платформа для развертывания, масштабирования и управления веб-приложениями Amazon Web Services (AWS);

- удобна для приложений на базе ASP.Net благодаря полной совместимости Microsoft Azure;

- предлагает надежные и высокопроизводительные решения для размещения приложений Google Cloud Platform.

Преимущества:

- обеспечивают масштабируемость и надежность;
- упрощают управление инфраструктурой.

Ограничения:

- постоянные затраты на использование облачных ресурсов;
- зависят от качества интернет-соединения.

Таким образом, методы, основанные на использовании готовых платформ и CMS, подходят для быстрого создания стандартных интернет-магазинов, но ограничивают гибкость. Индивидуальная разработка обеспечивает создание уникальных решений, но требует больше времени и ресурсов. Использование облачных технологий дополняет оба подхода, повышая производительность и доступность системы.

Для реализации Neighbor Flowers выбран метод индивидуальной разработки с использованием ASP.Net и Vue.js, что позволяет создать гибкую и масштабируемую систему с уникальным функционалом.

2 АЛГОРИТМИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ РЕШЕНИЯ ЗАДАЧИ ПРОДАЖ

2.1 Технико-экономическое обоснование

Детальный и полный анализ требований, общая структура проекта обеспечивает скорость разработки системы, поэтому хорошее построение системы проекта и технико-экономический анализ имеют важное значение для проекта. В этой статье подробно разбираются функциональные и нефункциональные требования веб-сайта, а также подробно анализируется каждая часть.

Целью технико-экономического обоснования является превышение экономической выгоды от разработки сайта над затратами на разработку. Прежде всего, стоимость разработки этого веб-сайта очень низкая: стоимость онлайн-продаж эквивалентна устранению затрат на офлайн-рекламу. Во-вторых, для размещения заказов пользователям достаточно зайти на этот сайт с ПК, избегая транспортных расходов, связанных с поездками. Интернет может рекламировать предприятия в течение всего дня. Удобство Интернета можно использовать для своевременного устранения разницы во времени и географических препятствий, тем самым увеличивая продажи и увеличивая популярность. Он может предоставлять пользователям удобные услуги в режиме реального времени и эффективно.

В условиях все более продвинутого информационного развития Интернета люди склонны полагаться на онлайн-продажи и удобство. Путем создания относительно разумного макета, четких изображений цветов и дружественного интерфейса взаимодействия с пользователем потребности людей в покупке товаров могут быть в основном удовлетворены. Клиентам и администраторам в процессе использования данного веб-сайта нет необходимости заботиться и понимать конкретную внутреннюю структуру и методы реализации данного веб-сайта, для этого им достаточно нажать соответствующую кнопку в интерфейсе ПК. функциональность сайта.

На этом веб-сайте планируется использовать язык C#, один из самых популярных языков в настоящее время. Языковая платформа C# – ASP.NET использует технологию .NET для обеспечения более высокого уровня безопасности и эффективности разработки. Разработчики могут легко создавать эффективные приложения. SQL Сервер – это популярная база данных, которая проста в использовании, очень гибка и хорошо совместима с другим программным обеспечением. Она может удовлетворить потребности веб-сайта Neighbor Flowers в хранении больших объемов данных.

2.2 Анализ требований

2.2.1 Функциональные требования

Веб-платформа Neighbor Flowers предусматривает роли администратора и пользователя. Каждая роль имеет свои функциональные возможности, которые реализуются через следующие модули:

- модуль входа и регистрации, пользователи и администраторы должны пройти процесс регистрации, указав необходимые данные (имя, электронная почта, пароль и т.д.). После регистрации пользователь обязан войти в систему, используя свои учетные данные (имя пользователя и пароль), которые проверяются на соответствие данным в базе данных. Администраторы используют отдельные учетные данные для входа, что обеспечивает доступ к расширенным функциям управления платформой;

- модуль управления продуктами, пользователи могут просматривать информацию о продуктах, переходя на страницу сведений о товаре. Администраторы имеют возможность добавлять новые продукты, редактировать существующие и удалять их с виртуальных «полок»;

- модуль управления заказами, пользователи могут размещать заказы, просматривать информацию о получателе, общую стоимость заказа и его статус. Администраторы имеют доступ к управлению заказами: просмотр, обновление статуса заказа и контроль процесса доставки;

- модуль управления сообщениями, пользователи могут публиковать сообщения на платформе после успешного входа в систему. Администраторы могут

просматривать сообщения пользователей и отвечать на них, обеспечивая обратную связь;

- модуль личного кабинета, пользователи могут управлять своими данными, просматривать историю заказов и редактировать личную информацию.

На рисунке 6 показана схема с распределением функционала между пользователями и администраторами веб-платформы.

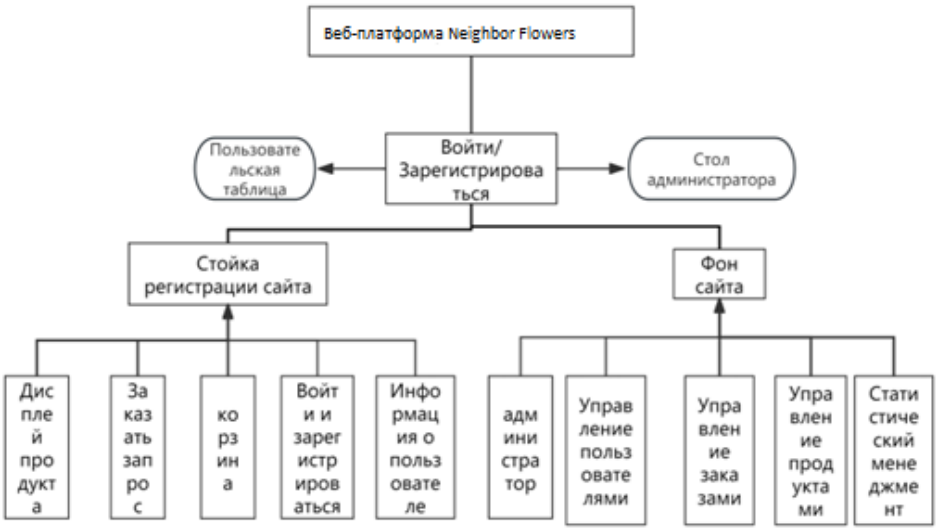

Рисунок 6 – Схема с распределением функционала между пользователями и администраторами

Схема включает следующие основные компоненты:

- пользовательская таблица, этот модуль отвечает за управление данными пользователей, включая их регистрацию и вход в систему;

- войти/зарегистрироваться функциональный блок, позволяющий пользователям войти в систему или зарегистрироваться, обеспечивая доступ к персонализированным функциям сайта;

- стол администратора – модуль предназначен для администраторов, предоставляя им доступ к управлению пользователями, заказами и продуктами;

- стойка регистрации сайта компонент, отвечающая за процесс регистрации новых пользователей на сайте;

- фон сайта – элемент, который обеспечивает визуальное оформление и дизайн сайта, создавая привлекательный интерфейс для пользователей;

- дисплей продукта – модуль, который отображает информацию о продуктах, доступных на сайте, позволяя пользователям просматривать и выбирать товары;

- заказать запрос функция, позволяющая пользователям оформлять заказы на продукты, выбранные на сайте;

- корзина – модуль, отвечающий за корзину продуктов;

- информация о пользователе модуль, который предоставляет данные о зарегистрированных пользователях, включая их профили и историю заказов;

- управление пользователями функционал для администраторов, позволяющий управлять учетными записями пользователей;

- управление заказами модуль, который позволяет администраторам отслеживать и управлять заказами, размещенными пользователями;

- управление продуктами компонент, отвечающий за добавление, редактирование и удаление продуктов на сайте;

- статистический менеджмент – модуль, который предоставляет аналитику и статистику по деятельности сайта, помогая администраторам принимать решения.

Эта схема демонстрирует взаимодействие различных модулей, обеспечивающих функциональность и удобство использования веб-сайта.

2.2.2 Нефункциональные требования

Требования делятся на следующие виды:

- требования к безопасности, веб-платформа использует базу данных SQL Server для хранения всех данных, включая учетные записи пользователей, информацию о продуктах и заказах. Для обеспечения безопасности данных используются пароли и шифрование соединений (HTTPS). Реализована защита от несанкционированного доступа к данным, включая механизмы аутентификации и авторизации;

- требования к производительности, веб-сайт должен обеспечивать стабильную работу при одновременном использовании большим количеством пользователей. Время отклика системы не должно превышать 2–3 секунды для

большинства операций. Платформа должна быть оптимизирована для работы на различных устройствах и браузерах;

- требования к надежности, веб-сайт должен быть доступен 24/7, за исключением планового технического обслуживания. В случае возникновения ошибок или сбоев система должна предоставлять пользователям понятные сообщения об ошибках и инструкции по дальнейшим действиям;

- требования к удобству использования, интерфейс платформы должен быть интуитивно понятным и удобным для пользователей. На всех этапах взаимодействия с платформой пользователи должны получать оперативную обратную связь (например, подтверждение успешной регистрации, уведомления о статусе заказа и т.д.).

2.3 Анализ отрасли и спроса

2.3.1 Характеристики цветочной розничной торговли

Цветочная розничная торговля имеет ряд уникальных характеристик, которые влияют на стратегию управления бизнесом и разработку платформы Neighbor Flowers:

- чувствительность ко времени, цветы имеют короткий срок хранения, в среднем около 3 дней. Это требует быстрой оборачиваемости запасов, чтобы минимизировать потери и максимизировать прибыль. Средний показатель оборачиваемости запасов в отрасли составляет около 65 %;

- сценарии потребления, большая часть заказов (78 %) поступает с самовывозом из офлайн-магазинов. Однако доля онлайн-заказов постоянно растет и уже достигает 22 %, что подчеркивает важность развития цифровых каналов продаж;

- нестабильность спроса, объем заказов значительно увеличивается в праздничные дни, такие как День святого Валентина и Китайский день святого Валентина. В такие периоды объем заказов может превышать обычный уровень в 5–8 раз, что требует гибкости и масштабируемости системы.

На основе анализа характеристик отрасли были определены следующие цели для платформы Neighbor Flowers:

- создание единой многоканальной платформы продаж;

- интеграция онлайн и офлайн каналов продаж для обеспечения удобства и гибкости для клиентов;

- динамичный и точный контроль запасов;

- реализация системы управления запасами, которая позволяет отслеживать и обновлять информацию о наличии товаров в реальном времени, минимизируя потери из-за короткого срока хранения цветов;

- увеличение показателя повторных покупок клиентов;

- достижение показателя повторных покупок более 35 % за счет улучшения качества обслуживания, персонализации предложений и эффективной системы лояльности.

2.3.2 Модель анализа поведения клиентов

Для повышения уровня повторных покупок и улучшения взаимодействия с клиентами была разработана модель анализа поведения клиентов на основе матрицы RFM (Recency, Frequency, Monetary). Этот метод позволяет сегментировать клиентов на основе их поведения и принимать данные-ориентированные решения для оптимизации маркетинговых стратегий.

Матрица RFM – инструмент анализа клиентской базы, который основывается на трех ключевых показателях:

- recency (давность последней покупки) как давно клиент совершал покупку. Клиенты, которые покупали недавно, с большей вероятностью отреагируют на маркетинговые предложения;

- frequency (частота покупок) как часто клиент совершает покупки. Клиенты с высокой частотой покупок считаются более лояльными;

- monetary (денежная ценность) – сколько денег клиент тратит за определенный период. Клиенты с высокой денежной ценностью приносят больше прибыли.

На рисунке 7 представлен пример использования RFM матрицы в производстве.

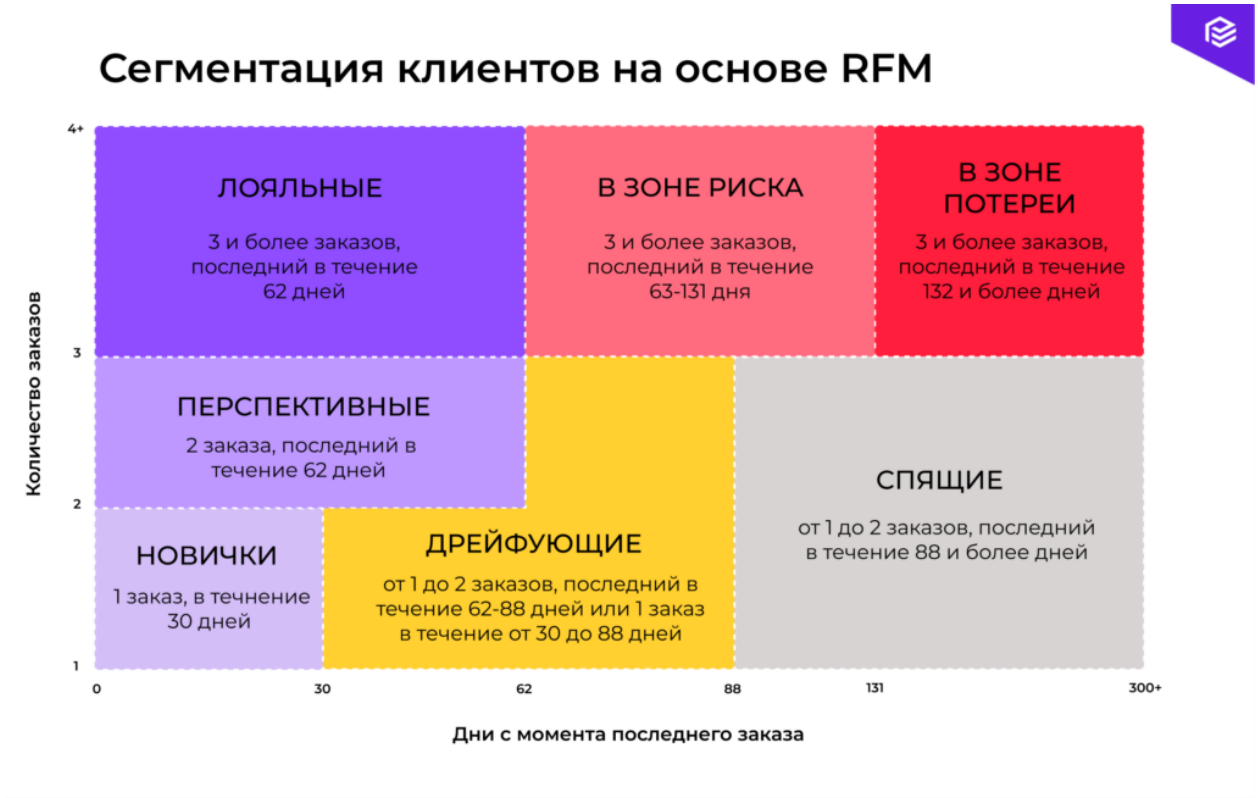

Рисунок 7 – Пример использования RFM матрицы

На основе этих параметров клиенты делятся на группы, такие как:

- лояльные клиенты – часто покупают, тратят много и недавно совершали покупки;

- потерянные клиенты – давно не покупали, несмотря на прошлую активность;

- потенциально ценные клиенты – тратят много, но покупают редко;

- новые клиенты – совершили первую покупку недавно, но их частота и сумма покупок пока неизвестны.

Для наглядности сегментации клиентов на платформе Neighbor Flowers была построена матрица значений RFM.

В таблице 2 представлены основные типы клиентов и их характеристики.

Таблица 2 – Матрица значений RFM

Тип клиента	Recency (давность)	Frequency (частота)	Monetary (денежная ценность)	Характеристики
Лояльные клиенты	Высокий	Высокий	Высокий	Часто покупают, тратят много, недавно совершали покупки.
Потерянные клиенты	Низкий	Низкий	Низкий	Давно не покупали, низкая активность и сумма покупок.
Потенциально ценные клиенты	Высокий	Низкий	Высокий	Тратят много, но покупают редко.
Новые клиенты	Высокий	Низкий	Низкий	Совершили первую покупку недавно, но их частота и сумма покупок пока неизвестны.

На основе анализа данных платформы Neighbor Flowers были выделены следующие группы клиентов:

- лояльные клиенты (высокий Recency, Frequency, Monetary) им были предложены эксклюзивные скидки и доступ к новым продуктам;

- потерянные клиенты (низкий Recency, Frequency, Monetary) для них была запущена кампания с персональными скидками и напоминаниями о платформе;

- новые клиенты (высокий Recency, низкий Frequency, Monetary) им была предложена скидка на следующую покупку, чтобы стимулировать повторные заказы.

Использование матрицы RFM позволило платформе Neighbor Flowers более эффективно управлять клиентской базой, повышать уровень лояльности и увеличивать прибыль. Этот инструмент стал важной частью стратегии анализа поведения клиентов и оптимизации маркетинговых усилий.

2.3.3 Сравнение планов улучшения

Для выбора оптимальной модели прогнозирования и анализа данных были рассмотрены три типа моделей: ARIMA, LSTM и гибридная модель. Каждая из этих моделей имеет свои особенности, которые делают их подходящими для различных сценариев использования. Ниже приведено подробное описание каждой модели, их преимущества и недостатки.

ARIMA – это статистическая модель, используемая для анализа и прогнозирования временных рядов. Она состоит из трех компонентов:

- зависимость текущего значения от предыдущих значений AR (Autoregressive);

- дифференцирование для стабилизации временного ряда (устранения тренда) I (Integrated);

- зависимость текущего значения от ошибок прогноза предыдущих значений MA (Moving Average).

Преимущества:

- высокая объяснимость, результаты модели ARIMA легко интерпретировать, что делает её полезной для бизнес-аналитики;

- эффективность для линейных данных хорошо работает с временными рядами, где зависимости между данными линейны;

- простота настройки, параметры модели (p, d, q) могут быть подобраны с использованием стандартных методов, таких как поиск по сетке.

Недостатки:

- ограниченность для нелинейных данных, ARIMA плохо справляется с данными, где присутствуют сложные нелинейные зависимости;

- чувствительность к шуму, модель может быть чувствительной к выбросам и шуму в данных.

Применение в проекте:

- используется для прогнозирования спроса на популярные товары, где временные ряды имеют стабильные линейные зависимости ARIMA.

- тип рекуррентной нейронной сети (RNN), предназначенный для работы с последовательными данными. LSTM способна запоминать долгосрочные зависимости в данных, что делает её особенно полезной для анализа временных рядов LSTM.

Преимущества:

- эффективность для нелинейных данных, LSTM отлично справляется с данными, где присутствуют сложные нелинейные зависимости;

- учет долгосрочных зависимостей, модель способна запоминать информацию из далекого прошлого, что полезно для прогнозирования;

- гибкость, LSTM может быть адаптирована для различных задач, таких как классификация, прогнозирование и анализ последовательностей.

Недостатки:

- сложность интерпретации, результаты LSTM сложно объяснить, что делает её менее подходящей для задач, где важна прозрачность модели;

- высокие вычислительные затраты, обучение LSTM требует значительных ресурсов и времени;

- требовательность к данным, для эффективного обучения LSTM требуется большой объем данных.

Применение в проекте:

- используется LSTM для прогнозирования спроса на товары с нелинейными зависимостями, где традиционные методы, такие как ARIMA, не справляются.

Гибридная модель сочетает в себе преимущества ARIMA и LSTM. Например, ARIMA может использоваться для обработки линейных компонент временного ряда, а LSTM для учета нелинейных зависимостей.

Преимущества:

- глобальный оптимум, гибридная модель способна учитывать как линейные, так и нелинейные зависимости, что повышает точность прогнозирования;

- баланс между точностью и объяснимостью, в отличие от LSTM, гибридная модель может быть частично интерпретируемой благодаря использованию ARIMA;

- гибкость, модель может быть адаптирована для различных сценариев, где требуется учет как линейных, так и нелинейных компонент.

Недостатки:

- разработка и настройка гибридной модели требует глубоких знаний в области машинного обучения и статистики;

- как и LSTM, гибридная модель требует значительных ресурсов для обучения.

Применение в проекте:

- гибридная модель была использована для прогнозирования спроса на товары, где присутствуют как линейные, так и нелинейные зависимости. Это позволило достичь высокой точности прогнозов.

В таблице 3 представлено сравнение моделей ARIMA, LSTM и гибридной модели по ключевым параметрам:

Таблица 3 – Сравнение моделей ARIMA, LSTM и гибридной модели

Параметр	ARIMA	LSTM	Гибридная модель
Возможность захвата характеристик	Линейные данные	Нелинейные данные	Линейные и нелинейные данные
Эффективность обучения	Средняя	Высокая	Высокая
Объяснимость	Высокая	Низкая	Средняя
Сложность реализации	Низкая	Высокая	Высокая
Вычислительные затраты	Низкие	Высокие	Высокие

Каждая из рассмотренных моделей имеет свои преимущества и недостатки. ARIMA подходит для анализа линейных данных и обладает высокой объяснимостью, LSTM эффективна для работы с нелинейными зависимостями, а гибридная модель сочетает в себе преимущества обоих подходов. В рамках проекта Neighbor Flowers выбор модели зависит от конкретной задачи: ARIMA

используется для прогнозирования спроса на популярные товары, LSTM для анализа нелинейных данных, а гибридная модель для комплексных задач, где требуется учет как линейных, так и нелинейных зависимостей.

2.4 Обзор возможностей профильного программного обеспечения

2.4.1 Платформы и фреймворки

Spring Framework – это фреймворк с открытым исходным кодом, который предоставляет контейнер с функциями инверсии управления (Inversion of Control, IoC). Хотя Spring не ограничивает модель программирования, его популярность в экосистеме Java привела к тому, что он стал дополнением или даже заменой модели EJB (Enterprise Java Beans).

Vue.js – это прогрессивный JavaScript-фреймворк для создания пользовательских интерфейсов. Он отличается простотой интеграции, высокой производительностью и гибкостью.

Преимущества:

- имеет интуитивно понятный API, что делает его доступным для разработчиков с разным уровнем опыта;

- автоматически обновляет DOM при изменении данных, что упрощает разработку динамических интерфейсов;

- позволяет создавать переиспользуемые компоненты, что упрощает поддержку и масштабирование кода;

- легко интегрируется с существующими проектами и библиотеками, такими как Vuex для управления состоянием и Vue Router для маршрутизации.

Vue.js может быть использован для создания интерактивного и отзывчивого пользовательского интерфейса платформы Neighbor Flowers. Он позволяет реализовать следующие компоненты системы:

- отображение списка товаров с фильтрацией и сортировкой;

- реализация корзины покупок с динамическим обновлением;

- управление профилем пользователя и заказами.

Express.js – минималистичный фреймворк для создания веб-приложений и API на платформе Node.js. Он предоставляет простой и гибкий способ разработки серверной части приложений.

Преимущества:

- не накладывает строгих ограничений, что позволяет разработчикам гибко настраивать приложение;

- благодаря асинхронной природе Node.js, Express.js обеспечивает высокую скорость обработки запросов;

- позволяет легко создавать RESTful API и управлять маршрутами;

- поддерживает использование промежуточного ПО (middleware) для обработки запросов, что упрощает реализацию таких функций, как аутентификация, логирование и обработка ошибок.

Express.js может быть использован для создания RESTful API, который будет взаимодействовать с фронтендом на Vue.js. К примеру, можно реализовать следующий функционал:

- обработка запросов на регистрацию и авторизацию пользователей;

- управление заказами и продуктами через API;

- интеграция с платежными системами.

Django – высокоуровневый фреймворк для создания веб-приложений на языке Python. Он предоставляет множество встроенных функций.

Преимущества:

- предоставляет встроенные решения для аутентификации, администрирования, ORM (Object-Relational Mapping) и других задач;

- включает механизмы для защиты от распространенных уязвимостей, таких как SQL-инъекции и XSS;

- подходит для создания как небольших, так и крупных проектов;

- с помощью Django REST Framework можно легко создавать API для взаимодействия с фронтендом.

Django может быть использован для создания серверной части платформы Neighbor Flowers. Например:

- управление пользователями и продуктами через административную панель;

- создание RESTful API для взаимодействия с фронтендом;

- реализация системы аутентификации и авторизации.

React – библиотека JavaScript для создания пользовательских интерфейсов, разработанная Facebook. Она широко используется для создания одностраничных приложений (SPA).

Преимущества:

- позволяет создавать переиспользуемые компоненты, что упрощает разработку и поддержку;

- использует виртуальный DOM для оптимизации обновлений интерфейса, что повышает производительность;

- имеет множество библиотек и инструментов, таких как Redux для управления состоянием и React Router для маршрутизации;

- может быть использован для создания изоморфных приложений, которые рендерятся как на сервере, так и на клиенте.

React может быть использован для создания динамического и отзывчивого интерфейса платформы Neighbor Flowers. Например:

- отображение списка товаров с возможностью фильтрации и сортировки;

- реализация корзины покупок с динамическим обновлением;

- управление профилем пользователя и заказами.

Flask – микрофреймворк для создания веб-приложений на Python. Он предоставляет минималистичный набор инструментов, что делает его гибким и легковесным.

Преимущества:

- не накладывает строгих ограничений, что позволяет разработчикам выбирать необходимые компоненты;

- подходит для создания небольших приложений и микросервисов;

- может быть использован для создания API с минимальными усилиями;

- поддерживает множество расширений для работы с базами данных, аутентификацией и другими задачами.

Flask может быть использован для создания RESTful API или микросервисов в рамках платформы Neighbor Flowers. Например:

- обработка запросов на регистрацию и авторизацию;

- управление заказами и продуктами через API;

- интеграция с платежными системами.

Angular – фреймворк для создания веб-приложений на TypeScript, разработанный Google. Он предоставляет полный набор инструментов для разработки сложных приложений.

Преимущества:

- включает встроенные решения для маршрутизации, управления состоянием и работы с формами;

- использование TypeScript повышает надежность и поддерживаемость кода;

- автоматически синхронизирует данные между моделью и представлением;

- имеет множество библиотек и инструментов для решения различных задач.

Angular может быть использован для создания сложного и масштабируемого интерфейса платформы Neighbor Flowers. Например:

- управление профилем пользователя и заказами;

- реализация корзины покупок с динамическим обновлением;

- отображение списка товаров с возможностью фильтрации и сортировки.

2.4.2 Среды разработки

IntelliJ IDEA – мощная среда разработки для Java и других языков программирования, разработанная JetBrains. Она поддерживает множество фреймворков, включая Spring, Hibernate и другие.

Интерфейс IntelliJ IDEA представлен на рисунке 8.

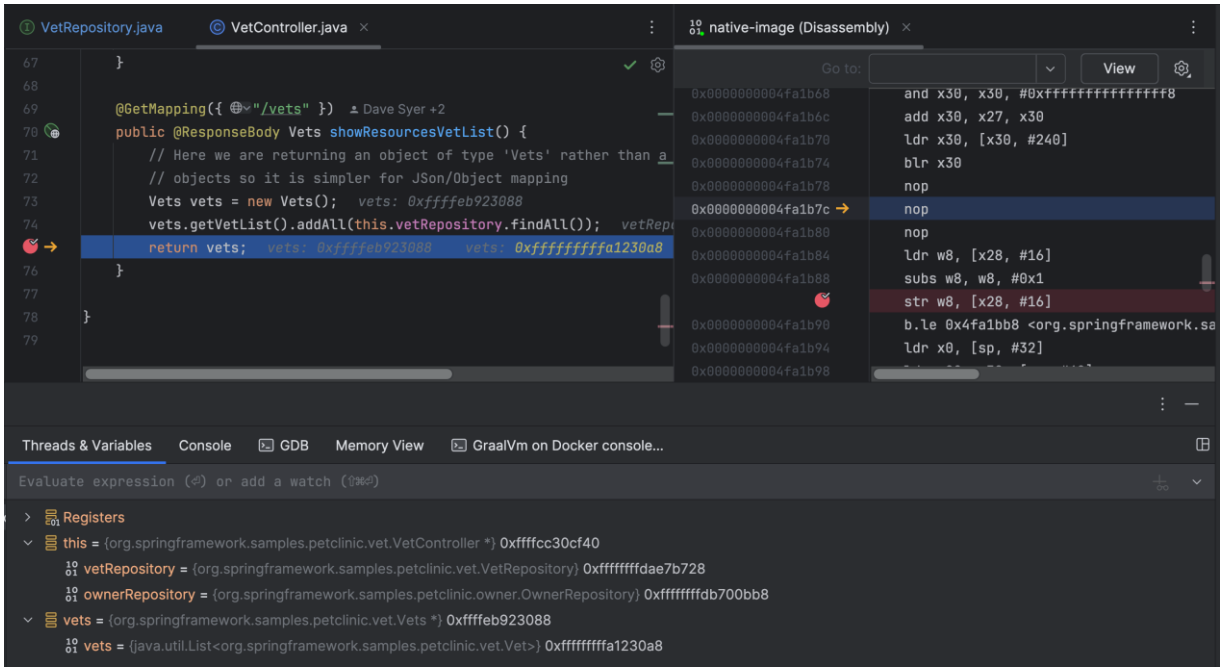

Рисунок 8 – Интерфейс IntelliJ IDEA

Преимущества:

- предоставляет встроенные инструменты для работы с Spring Framework, такие как автодополнение, рефакторинг и отладка;

- встроенные инструменты для работы с базами данных, такие как DataGrip;

- поддерживает JavaScript, TypeScript, Python, Kotlin и другие языки;

- широкая экосистема плагинов для расширения функциональности.

Недостатки:

- может потреблять значительное количество оперативной памяти;

- полнофункциональная версия IDE является платной, хотя существует бесплатная Community Edition.

Visual Studio Code – легковесный, но мощный редактор кода от Microsoft, который поддерживает множество языков программирования и фреймворков.

Преимущества:

- работает быстро даже на маломощных машинах;

- поддержка JavaScript, TypeScript, Python, Java, C++ и другие;

- встроенные инструменты для работы с системой контроля версий;

- огромное количество плагинов для расширения функциональности, включая поддержку Vue.js, React, Spring и других технологий.

Недостатки:

- для работы с Java требуется установка дополнительных расширений, таких как Java Extension Pack;

- по сравнению с полноценными IDE, такими как IntelliJ IDEA, VS Code требует больше настроек.

Eclipse – популярная среда разработки для Java, которая также поддерживает другие языки программирования через плагины.

Преимущества:

- полностью бесплатен и имеет открытый исходный код;

- предоставляет инструменты для работы с Spring Framework и другими Java-технологиями;

- широкая экосистема плагинов для расширения функциональности.

Недостатки:

- может быть сложным в настройке и использовании для новичков;

- как и IntelliJ IDEA, Eclipse может потреблять много оперативной памяти.

PyCharm – среда разработки для Python, разработанная JetBrains. Она поддерживает Django, Flask и другие Python-фреймворки.

Преимущества:

- предоставляет встроенные инструменты для работы с Python, включая автодополнение, отладку и рефакторинг;

- встроенная поддержка Django для создания веб-приложений;

- также поддерживает JavaScript, TypeScript, HTML и CSS.

Недостатки:

- может потреблять значительное количество оперативной памяти;

- полнофункциональная версия IDE является платной, хотя существует бесплатная Community Edition.

WebStorm – среда разработки для JavaScript и TypeScript, разработанная JetBrains. Она поддерживает Vue.js, React, Angular и другие фреймворки.

Преимущества:

- предоставляет встроенные инструменты для работы с JavaScript и TypeScript, включая автодополнение, отладку и рефакторинг;

- встроенная поддержка Vue.js, React, Angular и других фреймворков;

- встроенные инструменты для работы с системой контроля версий.

Недостатки:

- может потреблять значительное количество оперативной памяти;

- является платной IDE.

Показательное сравнение всех особенностей сред разработки представлено в таблице 4.

Таблица 4 – Сравнение сред разработки

Характеристика	IntelliJ IDEA	Visual Studio Code	Eclipse	PyCharm	WebStorm
1	2	3	4	5	6
Поддержка Java	Да	Через плагины	Да	Через плагины	Через плагины
Поддержка Spring	Да	Через плагины	Да	Нет	Нет
Поддержка Python	Через плагины	Да	Через плагины	Да	Нет
Поддержка JavaScript	Да	Да	Через плагины	Да	Да
Поддержка Vue.js	Через плагины	Да	Через плагины	Через плагины	Да
Поддержка React	Через плагины	Да	Через плагины	Через плагины	Да
Интеграция с Git	Да	Да	Да	Да	Да

Продолжение таблицы 4

1	2	3	4	5	6
Ресурсоемкость	Высокая	Низкая	Высокая	Высокая	Высокая
Стоимость	Платная (есть бесплатная версия)	Бесплатная	Бесплатная	Платная (есть бесплатная версия)	Платная

Выбор среды разработки зависит от конкретных задач проекта и предпочтений команды. Для разработки на Java и Spring Framework оптимальным выбором может быть IntelliJ IDEA. Если проект предполагает использование JavaScript, TypeScript и фреймворков, таких как Vue.js или React, то Visual Studio Code или WebStorm станут отличными вариантами. Для Python-разработки подойдет PyCharm, а для Java-проектов с ограниченным бюджетом Eclipse.

Каждая из этих сред разработки имеет свои преимущества и недостатки, поэтому выбор должен основываться на требованиях проекта, используемых технологиях и предпочтениях разработчиков.

2.4.3 Архитектурные стили

Архитектурные стили определяют набор принципов и шаблонов, которые используются для организации структуры программного обеспечения. Выбор подходящего архитектурного стиля для проекта Neighbor Flowers играет ключевую роль в обеспечении масштабируемости, поддерживаемости и производительности системы. Ниже приведен обзор популярных архитектурных стилей, которые могут быть использованы в проекте.

Монолитная архитектура предполагает, что все компоненты приложения (например, пользовательский интерфейс, бизнес-логика и доступ к данным) объединены в единое целое и выполняются в одном процессе.

Преимущества:

- легко разрабатывать и тестировать, так как все компоненты находятся в одном месте;

- развертывание приложения сводится к запуску одного исполняемого файла или контейнера;

- отсутствие необходимости в сложной инфраструктуре для взаимодействия между компонентами.

Недостатки:

- масштабирование требует запуска копии всего приложения, что может быть неэффективно;

- по мере роста приложения кодовая база становится сложной для понимания и изменения;

- ошибка в одном компоненте может привести к падению всего приложения.

Монолитная архитектура может быть использована на начальных этапах разработки, когда приложение имеет небольшую сложность и команда хочет быстро выпустить продукт.

Микросервисная архитектура предполагает разделение приложения на небольшие, независимые сервисы, каждый из которых выполняет определенную бизнес-функцию. Сервисы взаимодействуют друг с другом через API.

Преимущества:

- каждый сервис может масштабироваться независимо, что позволяет эффективно использовать ресурсы;

- возможность использовать разные технологии и языки программирования для разных сервисов;

- ошибка в одном сервисе не приводит к падению всего приложения;

- каждый сервис имеет небольшую кодовую базу, что упрощает разработку и тестирование.

Недостатки:

- требуется больше усилий для настройки взаимодействия между сервисами;

- необходимость в сложной инфраструктуре для оркестрации сервисов, мониторинга и логирования;

- каждый сервис требует отдельного процесса развертывания и управления.

Микросервисная архитектура может быть использована для крупных и сложных приложений, таких как Neighbor Flowers, где требуется высокая масштабируемость и гибкость.

Сервис-ориентированная архитектура (SOA) предполагает использование сервисов, которые предоставляют определенные бизнес-функции и взаимодействуют друг с другом через стандартные протоколы, такие как SOAP или REST.

Преимущества:

- сервисы могут быть использованы в разных приложениях и контекстах;
- возможность интеграции с различными системами и платформами;
- сервисы могут масштабироваться независимо.

Недостатки:

- требуется тщательное управление сервисами и их взаимодействием;
- использование стандартных протоколов может привести к увеличению накладных расходов;
- требуется больше усилий для настройки и интеграции сервисов.

SOA может быть использована для интеграции различных систем и сервисов в рамках платформы Neighbor Flowers, таких как платежные системы, системы доставки и другие.

Событийно-ориентированная архитектура предполагает, что компоненты системы взаимодействуют через события. Каждый компонент может генерировать события и реагировать на события, генерируемые другими компонентами.

Преимущества:

- компоненты могут быть легко добавлены или изменены без влияния на другие части системы;
- возможность обработки большого количества событий в реальном времени;
- компоненты могут работать независимо, что повышает производительность.

Недостатки:

- требуется тщательное проектирование системы событий и их обработки;
- отладка асинхронных систем может быть сложной;
- необходимость в инфраструктуре для обработки и маршрутизации событий.

Событийно-ориентированная архитектура может быть использована для обработки событий в реальном времени, таких как уведомления о заказах, обновления статусов и другие.

Слоистая архитектура предполагает разделение приложения на несколько слоев, таких как уровень представления, уровень бизнес-логики и уровень доступа к данным. Каждый слой взаимодействует только с соседними слоями.

Преимущества:

- четкое разделение ответственности между слоями упрощает разработку и тестирование;
- изменения в одном слое не влияют на другие слои;
- бизнес-логика может быть повторно использована в разных приложениях.

Недостатки:

- слоистая архитектура может быть менее гибкой по сравнению с микросервисной архитектурой;
- масштабирование требует масштабирования всего слоя, что может быть неэффективно.

Слоистая архитектура может быть использована для небольших и средних приложений, где требуется четкое разделение ответственности между компонентами.

Для отображения особенностей каждого архитектурного стиля необходимо использовать сравнительную таблицу. Сравнение архитектурных стилей представлено в таблице 5.

Таблица 5 – Сравнение архитектурных стилей

Характеристика	Монолитная архитектура	Микросервисная архитектура	SOA	Событийно-ориентированная архитектура	Слоистая архитектура
Масштабируемость	Низкая	Высокая	Средняя	Высокая	Средняя
Гибкость	Низкая	Высокая	Высокая	Высокая	Средняя
Сложность разработки	Низкая	Высокая	Высокая	Высокая	Низкая
Устойчивость к отказам	Низкая	Высокая	Средняя	Высокая	Средняя
Накладные расходы	Низкие	Высокие	Высокие	Высокие	Низкие
Применение	Небольшие проекты	Крупные и сложные проекты	Интеграция систем	Реальное время, события	Средние проекты

Выбор архитектурного стиля для проекта Neighbor Flowers зависит от требований к масштабируемости, гибкости и сложности системы.

Каждый из этих стилей имеет свои преимущества и недостатки, поэтому выбор должен основываться на конкретных требованиях проекта и долгосрочных целях.

Слоистая архитектура является оптимальным выбором для проекта Neighbor Flowers, так как обеспечивает необходимый баланс между простотой, расширяемостью и структурированностью системы. Она подходит для развития проекта, его поддержки и возможного масштабирования в будущем без излишних затрат на сложную инфраструктуру.

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРЕДЛАГАЕМОГО АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ ПРОДАЖ

3.1 Основные этапы практической разработки программного продукта

3.1.1 Проектирование модулей системы

Различные модули и функции, включенные во внешний и внутренний интерфейс всего веб-сайта, показаны на рисунке 9. Модуль входа включает в себя соответствующие операции, которые пользователи и администраторы могут выполнять для входа и регистрации на странице, а также пользователи могут изменять свою личную информацию. Модуль управления продуктами включает в себя то, что обычные пользователи могут только просматривать продукты, добавлять корзины покупок и фильтровать типы. Администраторы могут добавлять, удалять, изменять и проверять продукты. Модуль управления заказами включает в себя возможность пользователя изменять и отменять заказы после размещения заказа, а адрес заказа администратора также может управлять клиентами.

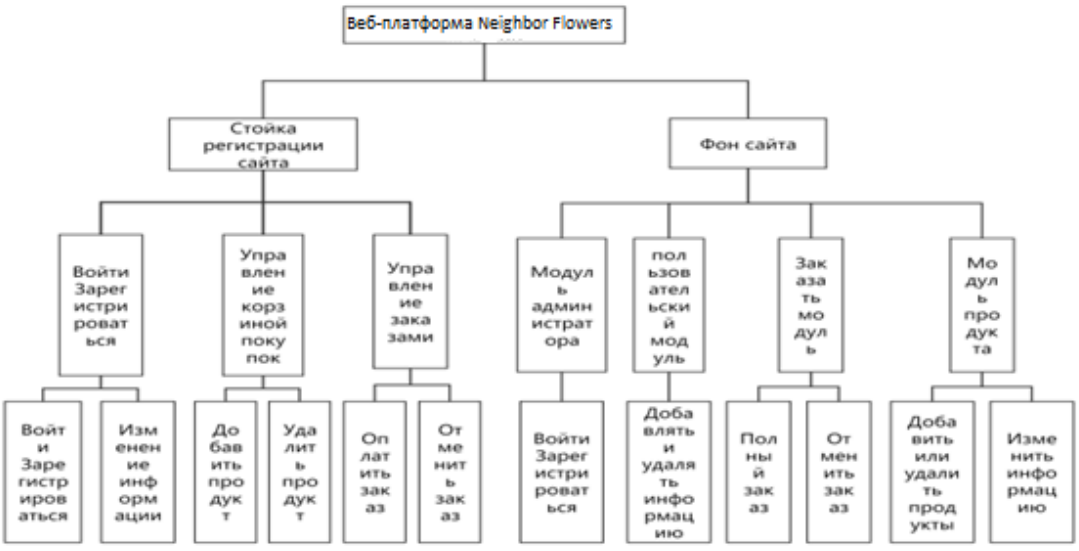

Рисунок 9 – Схема функциональной архитектуры пользовательского и административного интерфейса

В Модуле входа/регистрации пользователи могут завершить регистрацию, введя свою личную информацию, включая имя, контактную информацию,

пароль, код подтверждения и информацию о получателе, с помощью кнопки регистрации в правом верхнем углу страницы входа.

После входа в систему в модуле управления корзиной покупок пользователи могут просматривать личную информацию текущего пользователя в пользовательском центре, а также изменять личную информацию и пароли учетных записей.

В модуле управления заказами пользователь может просмотреть информацию о заказе в разделе «Мои заказы» или с помощью фильтрации типов и запроса заказа.

В модуль оценки пользователи могут оставлять сообщения о своих потребностях, предложениях и мнениях на доске объявлений. Содержимое сообщения включает заголовок и содержание.

В модуле новостей выполняются функции отображения всех продуктов, просмотра последних рекламных продуктов. Также есть возможность при нажатии на картинку войти в текущий интерфейс продукта, или добавить в корзину.

В администраторском модуле администратор может ввести правильную информацию об учетной записи и пароле на странице входа. После успешной проверки он автоматически перейдет на веб-сайт.

В модуле управления информацией о пользователях администратор может просмотреть личную информацию текущего пользователя.

В модуле управление заказами у пользователей есть доступ к заказам, мониторинг статуса заказов в режиме реального времени и отмена заказов.

В модуле оценки администраторы получают отзывы пользователей через доски объявлений и своевременно их обрабатывают.

В модуле управления продуктами администраторы могут быстро и точно выполнять операции добавления, изменения и удаления продуктов, вводя такую информацию, как названия продуктов, язык цветов, введение в классификацию и т. д. Среди них наиболее распространенной операцией является корректировка запасов.

В модуле управление статистики продаж администраторы могут получать и анализировать информацию о покупках и статистику продаж продуктов всех зарегистрированных пользователей в режиме реального времени посредством мониторинга и просмотра веб-сайта.

3.1.2 Проектирование и разработка базы данных

Сущность товар включает в себя идентификатор первичного ключа, идентификатор пользователя, имя пользователя, содержимое сообщения, содержимое ответа и атрибуты добавления времени.

Сущность пользователь включает в себя пол сущности, идентификатор пользователя, пароль для входа, имя для входа, номер администратора, рамку аватара, номер мобильного телефона, адрес электронной почты, баланс и время создания. Атрибуты сущности показаны на рисунке 10.

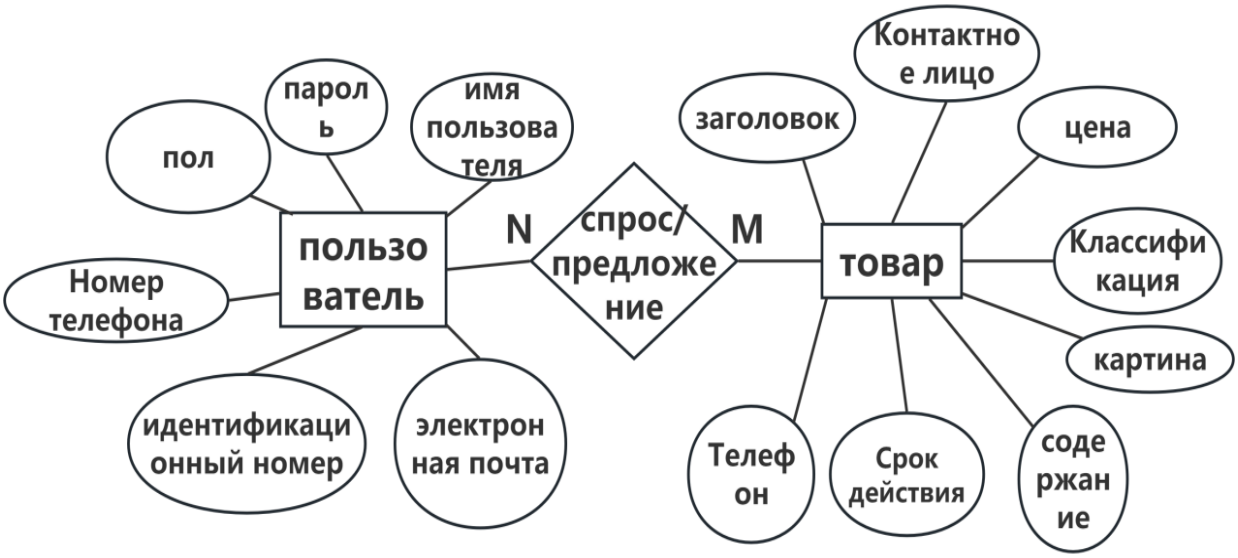

Рисунок 10 – Диаграмма отношений «пользователь товар»

Сущность «администратор» включает в себя идентификатор первичного ключа администратора, номер учетной записи, пароль, имя и другие атрибуты.

Диаграмма атрибутов сущности «администратор» представлена на рисунке 11.

Рисунок 11 – Диаграмма атрибутов сущности «администратор»

Сущность «заказ» включает в себя идентификатор пользователя, номер заказа, идентификатор первичного ключа, идентификатор продукта, время размещения заказа, цену заказа, возможность удаления заказа, статус заказа и атрибуты. Атрибуты сущности «заказ» показаны на рисунке 12.

Рисунок 12 – Диаграмма атрибутов сущности «заказ»

Сущность «товар» включает в себя идентификатор первичного ключа, название продукта, тип продукта, использование, материал цветка, объект, сок, цвет, детали.

Атрибуты сущности «товар» показаны на рисунке 13.

Рисунок 13 – Диаграмма атрибутов сущности «товар»

В рамках разработки веб-платформы Neighbor Flowers была спроектирована реляционная база данных, включающая несколько таблиц. Таблица пользователей в основном используется для хранения такой информации, как номер учетной записи, пароль, имя.

В таблице 6 представлена сущность пользователь (User) которая хранит информацию о зарегистрированных пользователях.

Таблица 6 – Сущность «Пользователь»

Имя поля	Тип данных	Длина	Описание
ID	VARCHAR	8	Первичный ключ
Username	VARCHAR	20	Логин пользователя
Password	VARCHAR	50	Хэшированный пароль
Name	VARCHAR	20	Имя пользователя
Gender	VARCHAR	2	Пол (М/Ж)
Avatar	VARCHAR	80	Ссылка на аватар
Email	VARCHAR	20	Электронная почта
Phone	VARCHAR	12	Номер телефона
Balance	FLOAT	-	Баланс счета
RegistrationDate	DATETIME	8	Дата регистрации

Благодаря проектированию концептуальной структуры базы данных была получена ER-диаграмма базы данных веб-сайта Neighbor Flowers.

Таблица администратора используется для хранения таких полей, как номера счетов и пароли. Ее структура показана в таблице 7.

Таблица 7 – Структура таблицы администратора

Имя поля	Тип данных	Длина	Описание
ID	VARCHAR	8	Первичный ключ
Username	VARCHAR	20	Логин администратора
Password	VARCHAR	50	Хэшированный пароль
Role	VARCHAR	100	Роль (например, "SuperAdmin")
RegistrationDate	DATETIME	8	Дата добавления
Email	VARCHAR	20	Электронная почта

Структура таблицы адресов хранит такие поля, как идентификатор пользователя, адрес, имя и т. д. Ее структура показана в Таблице 8.

Таблица 8 – Структура таблицы адресов

Имя поля	Тип данных	Длина	Описание
ID	VARCHAR	8	Первичный ключ
UserID	VARCHAR	20	Внешний ключ (User.ID)
Address	VARCHAR	100	Полный адрес
RecipientName	VARCHAR	20	Имя получателя
Phone	VARCHAR	30	Контактный телефон
IsDefault	VARCHAR	20	Адрес по умолчанию (Да/Нет)

Таблица сообщений используется для хранения идентификатора пользователя, содержимого сообщения, содержимого ответа и других полей.

Структура таблицы сообщений показана в Таблице 9.

Таблица 9 – Структура таблицы сообщений

Имя поля	Тип данных	Длина	Описание
ID	VARCHAR	8	Первичный ключ
UserID	INT	4	Внешний ключ (User.ID)
Nickname	VARCHAR	20	Имя отправителя
Content	TEXT	100	Текст сообщения
Reply	TEXT	100	Ответ администратора
CreatedAt	DATETIME	8	Дата отправки
Имя поля	Тип данных	Длина	Описание

Структура таблицы новостей хранит заголовок, содержание и другие поля информации. Структура показана в Таблице 10.

Таблица 10 – Структура таблицы новостей

Имя поля	Тип данных	Длина	Описание
ID	INT	4	Первичный ключ
Title	VARCHAR	20	Заголовок новости
Description	TEXT	100	Краткое описание
Image	VARCHAR	64	Ссылка на изображение
Content	TEXT	100	Полный текст новости
CreatedAt	DATETIME	8	Дата публикации

Структура таблицы заказов хранит такую информацию, как статус продукта, адрес заказа и т. д.

Структура таблицы заказов показана в Таблице 11.

Таблица 11 – Структура таблицы заказов

Имя поля	Тип данных	Длина	Описание
ID	INT	4	Первичный ключ
OrderNumber	VARCHAR	20	Номер заказа
UserID	INT	4	Внешний ключ (User.ID)
ProductID	INT	4	Внешний ключ (Product.ID)
UserName	VARCHAR	20	Имя пользователя
OrderPrice	FLOAT	4	Цена заказа
DiscountPrice	FLOAT	4	Цена со скидкой
TotalPrice	FLOAT	4	Итоговая стоимость
Status	VARCHAR	20	Статус заказа (например, "В обработке")
CreatedAt	DATETIME	8	Дата создания заказа

Таблица типов цветков используется для хранения идентификатора первичного ключа, типа цветка и времени добавления. Структура показана в Таблице 12.

Таблица 12 – Структура таблицы типов цветов

Имя поля	Тип данных	Длина	Описание
ID	VARCHAR	8	Первичный ключ
TypeName	VARCHAR	20	Название типа (например, "Свадебные")
CreatedAt	DATETIME	8	Дата добавления

В этой таблице хранятся идентификатор первичного ключа, использование цветка и время добавления.

Структура таблицы использования цветов показана в таблице 13.

Таблица 13 – Структура таблицы использования цветов

Имя поля	Тип данных	Длина	Описание
ID	VARCHAR	8	Первичный ключ
UsageName	VARCHAR	20	Назначение (например, "Подарок")
CreatedAt	DATETIME	8	Дата добавления

В таблице информации о цветах хранится такая информация, как названия цветов, типы цветов и их использование.

Структура таблицы цветов показана в Таблице 14.

Таблица 14 – Структура таблицы цветов

Имя поля	Тип данных	Длина	Описание
ID	VARCHAR	8	Первичный ключ
Name	VARCHAR	20	Название цветка
TypeID	VARCHAR	20	Внешний ключ (FlowerType.ID)
UsageID	VARCHAR	20	Внешний ключ (FlowerUsage.ID)
Image	VARCHAR	64	Ссылка на изображение
Material	VARCHAR	20	Материал (например, "Розы")
Color	VARCHAR	20	Цвет
Description	TEXT	100	Подробное описание
ClickCount	INT	4	Количество просмотров
Price	FLOAT	4	Цена
CreatedAt	DATETIME	8	Дата добавления

Обрабатывая ER-диаграмму, можно получить конкретную структуру таблицы. Логическая структура базы данных на этом веб-сайте в основном разрабатывает структуры таблиц для таких объектов, как продукты, пользователи, администраторы, заказы.

3.1.3 Проектирование структуры приложения

На рисунке 14 представлена Диаграмма классов приложения, его основные сущности.

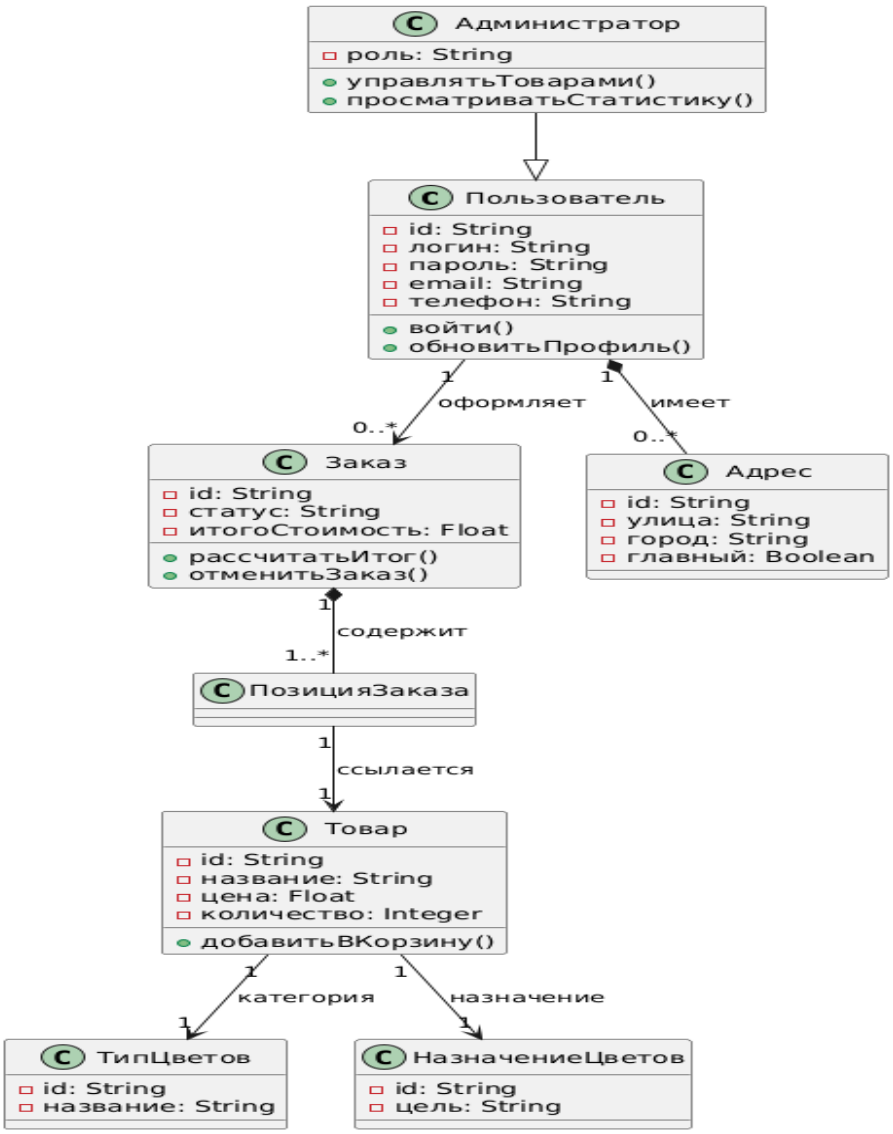

Рисунок 14 – Диаграмма классов приложения

Диаграмма последовательностей показывает пошаговое взаимодействие объектов при оформлении заказа.

Основные шаги:

- пользователь добавляет товар в корзину, проверка остатков (InventoryService);

- оформление заказа, обработка платежа (PaymentService);

- обновление остатков товара после успешной оплаты.

Критические точки:

- взаимодействие с внешними сервисами (платежи, инвентаризация);

- обработка ошибок (например, недостаточный остаток).

Диаграмма последовательностей для оформления заказа представлена на рисунке 15.

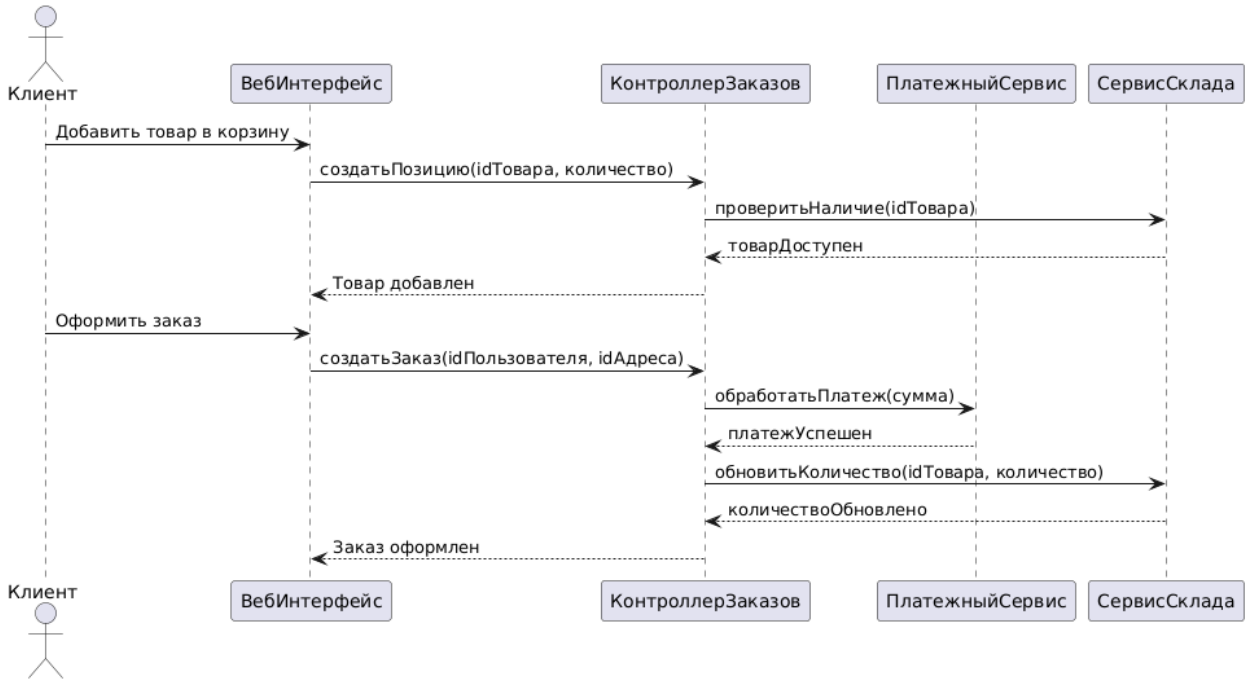

Рисунок 15 – Диаграмма последовательностей функции «Оформление заказа»

Диаграмма состояний описывает жизненный цикл заказа.

Состояния:

- после оплаты;

- упаковка;

- ошибка оплаты;

- доставлен;

- возврат.

Диаграмма состояний представлена на рисунке 16.

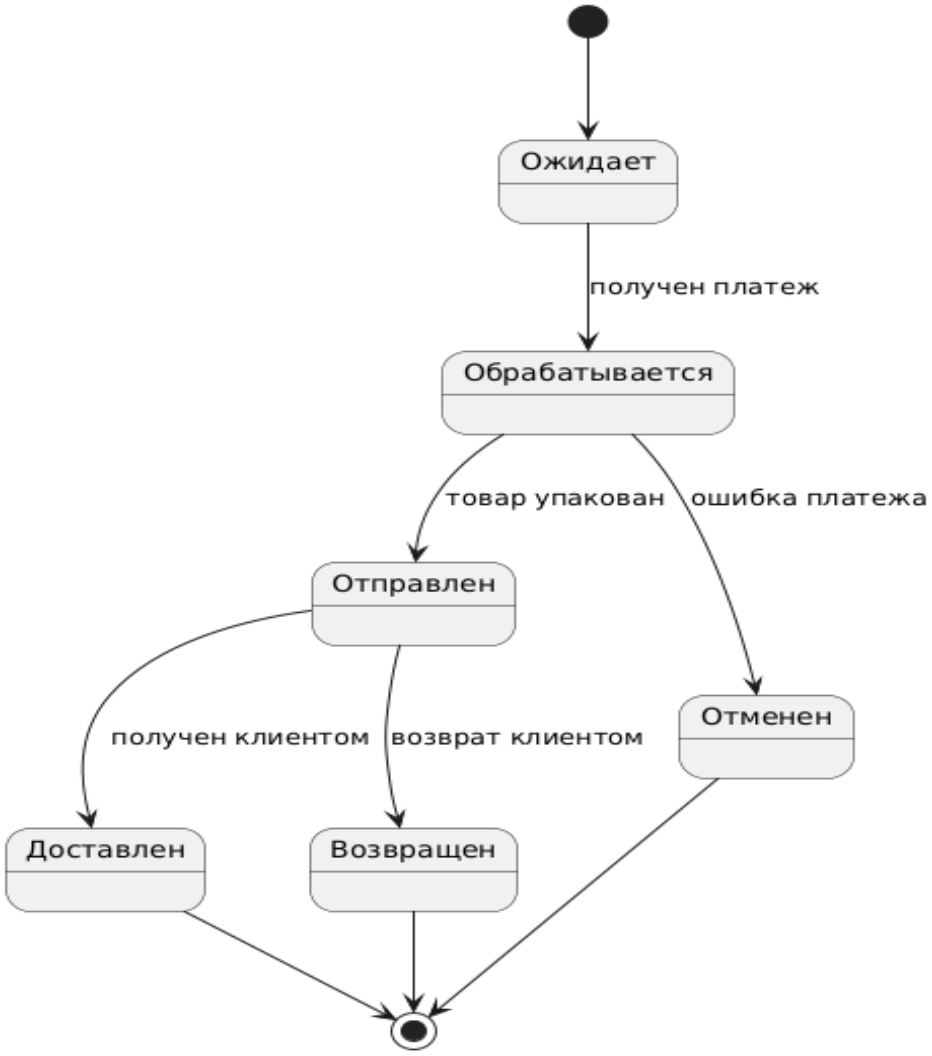

Рисунок 16 – Диаграмма состояний

Диаграмма компонентов отображает физическую архитектуру системы.

Компоненты:

- фронтенд Vue.js (интерфейс пользователя);
- бэкенд ASP.NET Core (API Gateway);
- база данных PostgreSQL;
- внешние сервисы платежная система (Stripe).

Диаграмма компонентов представлена на рисунке 17.

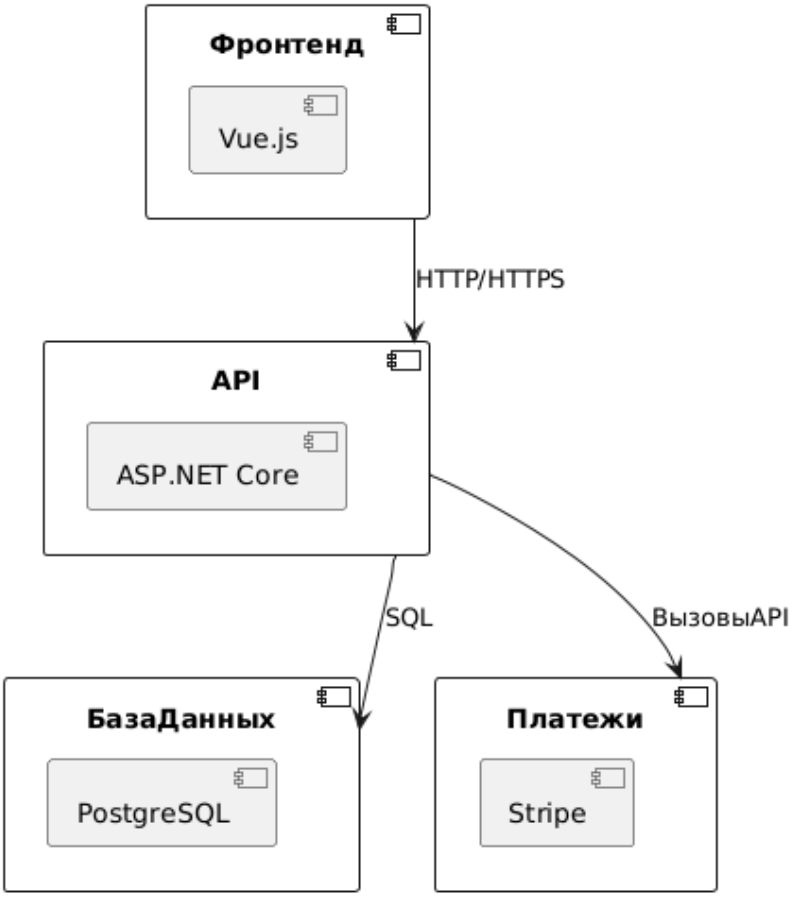

Рисунок 17 – Диаграмма компонентов

Диаграмма вариантов использования описывает функциональные возможности системы с точки зрения пользователей.

Акторы:

- покупатель, просмотр каталога, оформление заказов;

-администратор, управление товарами и пользователями.

Ключевые сценарии:

- управление товарами;

- управление пользователями;

- просмотр каталога;

- оформление заказов.

Диаграмма вариантов использования представлена на рисунке 18.

Рисунок 18 – Диаграмма вариантов использования

Для разработчиков:

- диаграмма классов помогает понять структуру БД;

- диаграммы последовательностей полезны при реализации API.

Для тестировщиков:

- диаграмма состояний определяет тестовые сценарии для заказов.

Для архитекторов:

- диаграмма компонентов показывает масштабируемость системы.

Таким образом, все диаграммы синхронизированы между собой и покрывают все ключевые аспекты системы от бизнес-логики до физической реализации.

3.2. Примеры фактического тестирования программного продукта

3.2.1 Структура пользовательского интерфейса

Структура пользовательского интерфейса – набор элементов, которые обеспечивают комфортное взаимодействие пользователя с электронным устройством.

Пользовательский интерфейс состоит из трёх основных частей:

- визуальное оформление, отвечающее за представление информации оператору;

- функциональные возможности системы, включающие набор возможностей для эффективного выполнения профессиональной деятельности;

- техники взаимодействия оператора с системой.

Вход на сайт может быть только в том случае, если введенные имя пользователя и пароль верны.

Интерфейс входа показан на рисунке 19.

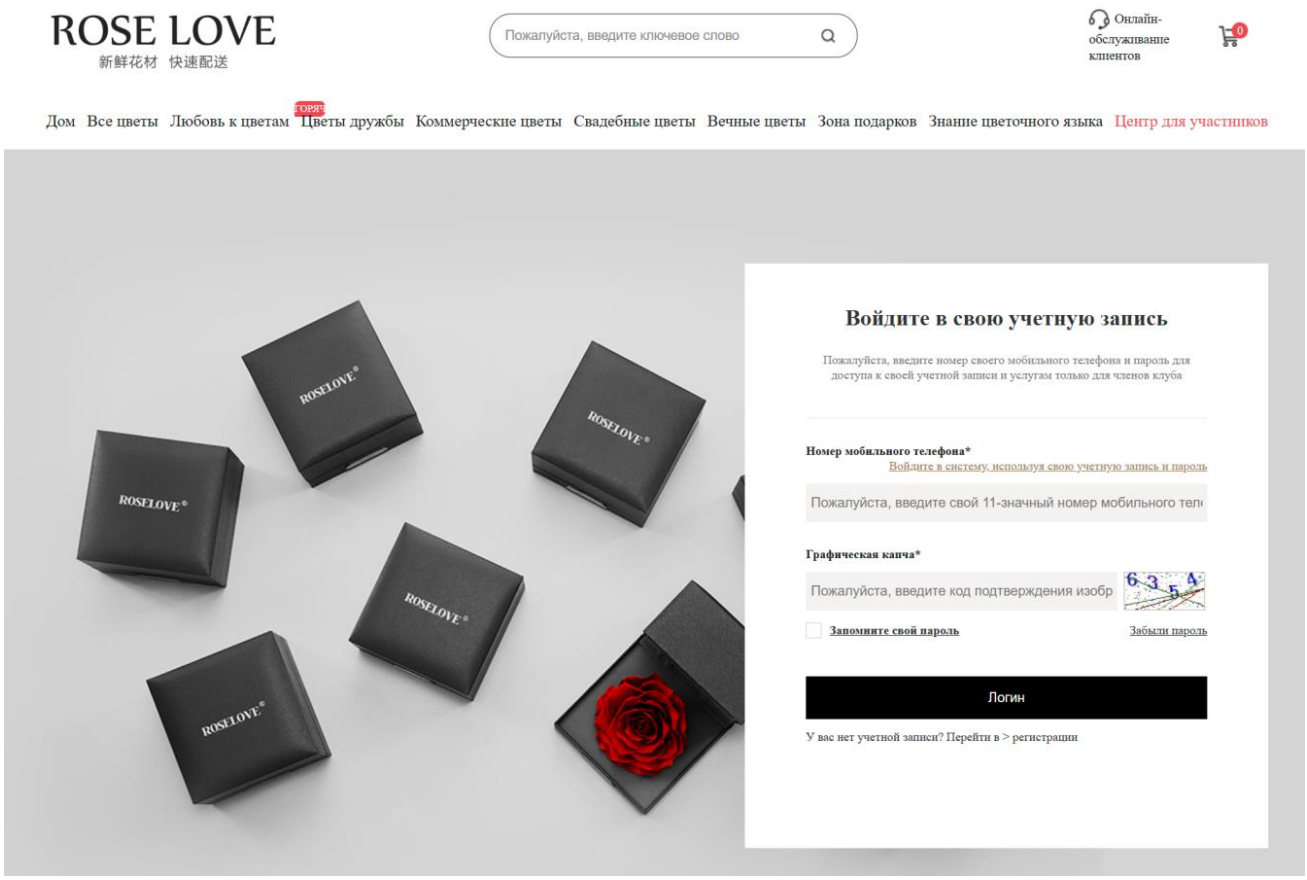

Рисунок 19 – Интерфейс входа на веб-платформу

После входа в систему пользователь нажимает «Мой заказ» на стойке регистрации или входит в систему администратора серверной части, чтобы открыть управление заказами, а затем запрашивает сведения о заказе пользователя в базе данных, или администратор запрашивает заказ пользователя и отображает его в интерфейсе управления серверной частью.

Интерфейс заказа показан на рисунке 20.

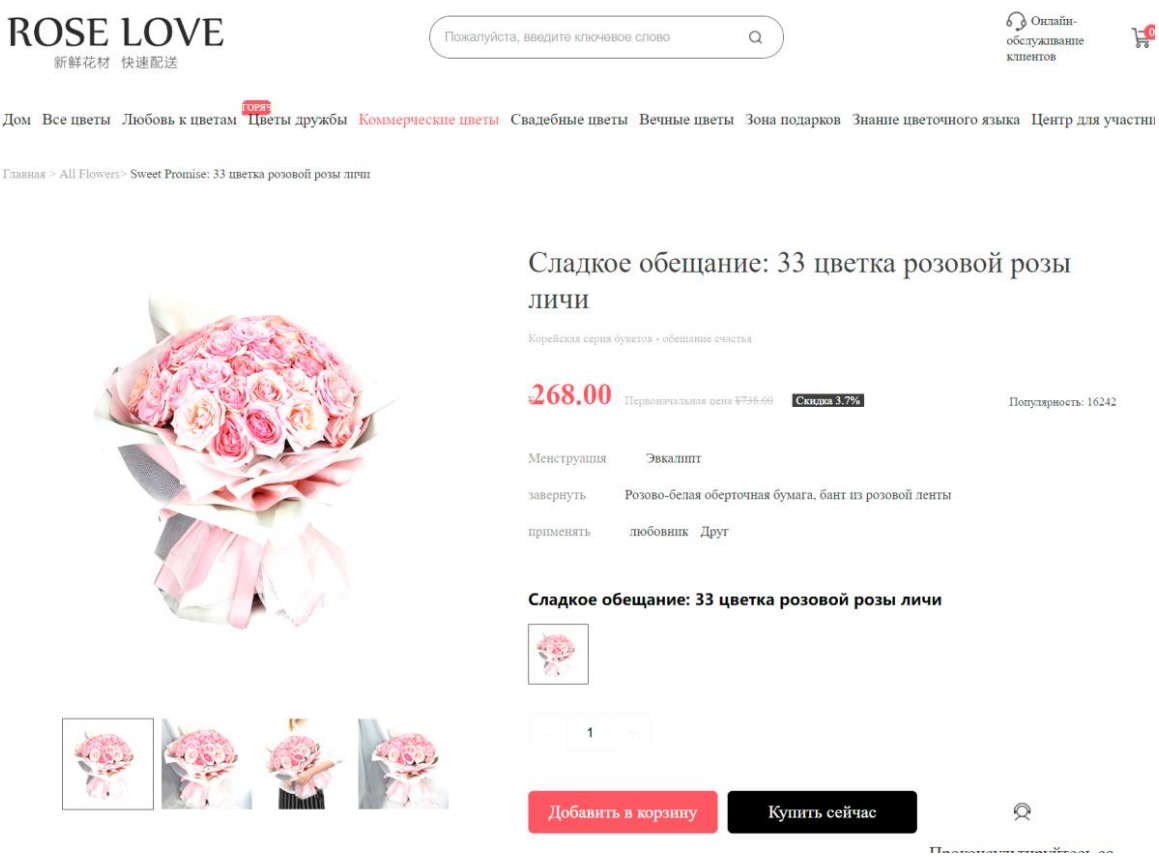

Рисунок 20 – Интерфейс заказа

Блок-схема программы управления заказами является диаграммой активности. Её использование заключается в визуализации логики процесса обработки заказов

Ключевые элементы:

- начало/конец процесса (круглые элементы);
- действия (прямоугольники с закругленными углами);
- условные ветвления (ромбы);
- цикл обработки (repeat while).

Блок-схема программы управления заказами представлена на рисунке 21.

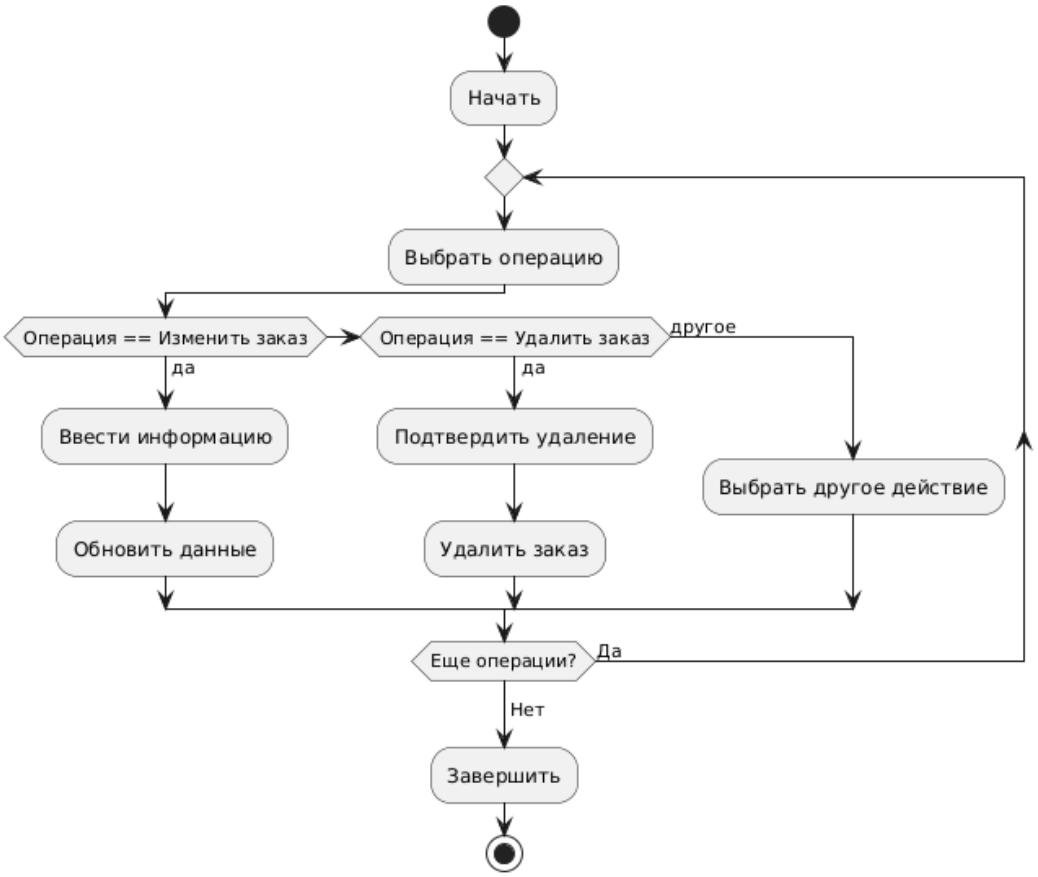

Рисунок 21 – Блок-схема программы управления заказами

Диаграмма компонентов содержит:

- субъект предпринимательской деятельности, основной бизнес-актор (например, ИП или юр.лицо);

- вариантов использования, сценарии работы системы (например, «Оформление заказа»);

- ведущий администратор, управляющий процессами.

Функции интерфейса доступа к данным:

- предоставляет API для чтения данных;

- выполняет CRUD-операции.

База данных:

- хранилище данных (например, PostgreSQL).

Диаграмма компонентов представлена на рисунке 22.

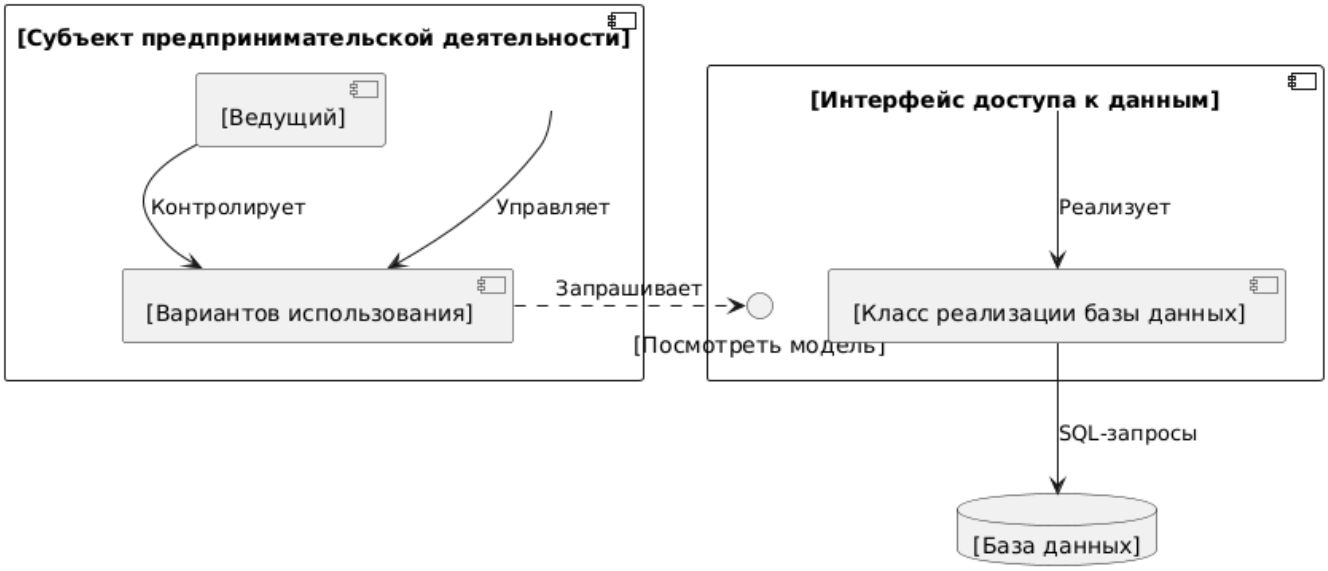

Рисунок 22 – Диаграмма компонентов

Описание элементов:

- старт процесса, инициализация работы с заказом (блок «Начинать»);

- основной цикл обработки, повторяющаяся последовательность выбора и выполнения операций;

- механизм контроля, валидация всех изменений и возможность отмены действия при ошибке.

Блок «Выбрать другое действие» позволяет добавлять новые операции:

- просмотр истории;

- экспорт данных («Экспертиз»);

- печать документов.

Ключевые ветвления:

- изменение заказа;

- удаление заказа;

- двойное подтверждение для критических операций.

Блок-схема управления заказами представлена на рисунке 23.

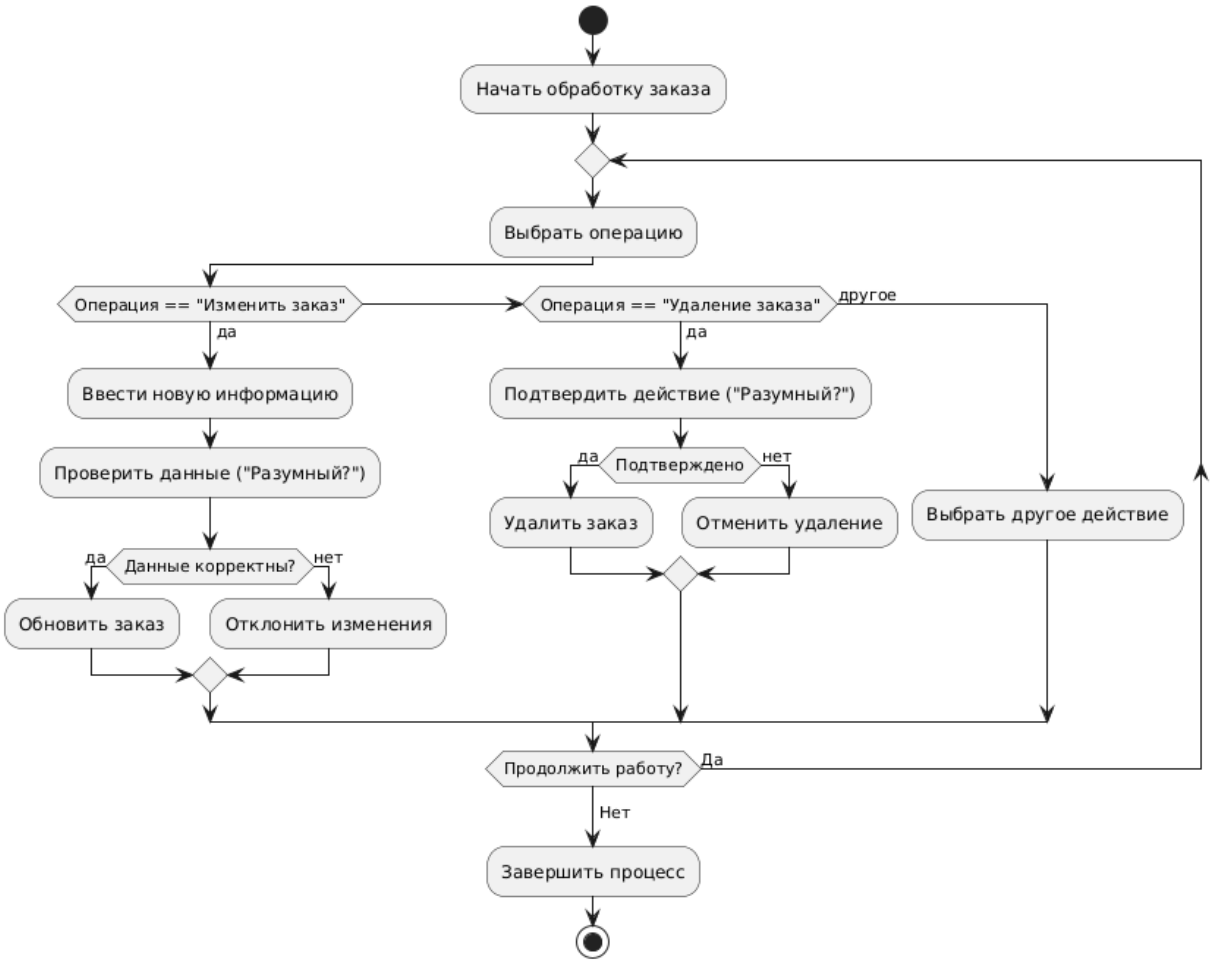

Рисунок 23 – Блок-схема программы управления заказами

Схема интерфейса персональной информации содержит данные о пользователе, введенные им при регистрации. В данном разделе пользователь может их изменить.

Весь перечень данных:

- имя пользователя;

- пароль;

- телефон;

- электронная почта.

Интерфейс с вводом персональной информации представлена на рисунке 24.

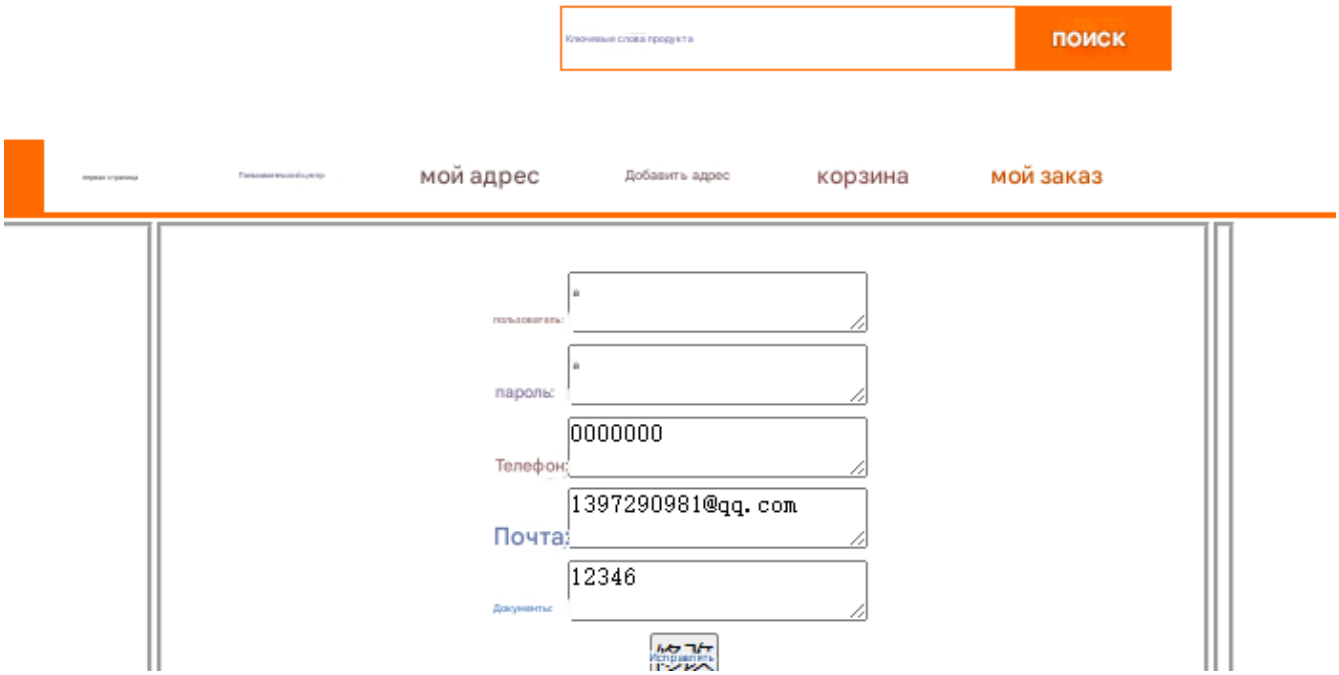

Рисунок 24 – Интерфейс с вводом персональной информации

Ключевые блоки программы обработки персональных данных и их назначение:

инициализация – старт работы с профилем (триггер «Нажмите на профиль»).

Механизмы контроля:

- двойная проверка для критических операций;
- валидация данных перед записью в БД;
- обязательное логирование удаления;
- изолированная обработка ошибок.

Диаграмма действий обработки персональных данных представлена на рисунке 25.

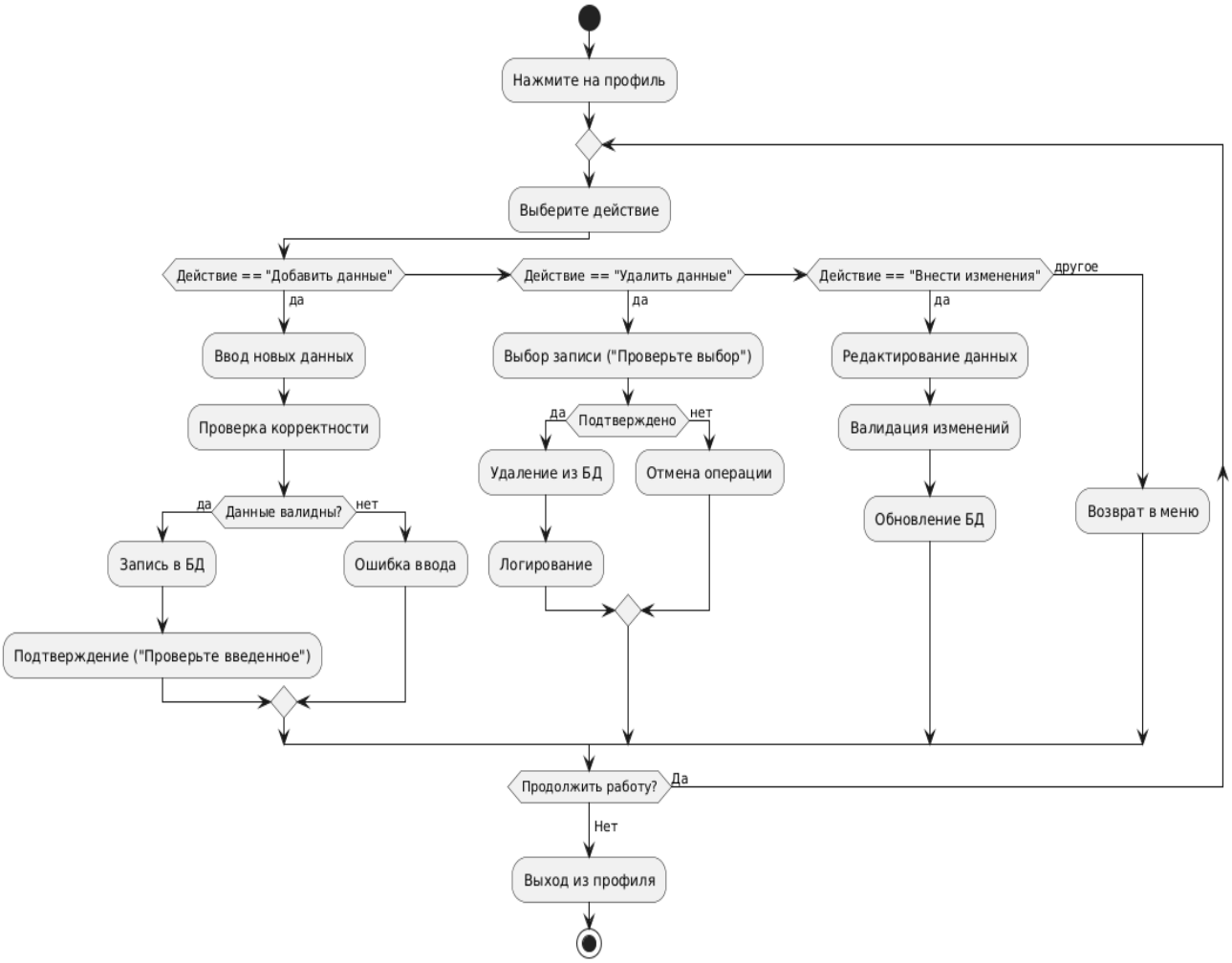

Рисунок 25 – Диаграмма действий обработки персональных данных

Схема управления данными пользователей позволяет администраторам следить за заказами и во время их выполнять. Это способствует продуктивности системы.

Данные представленные в схеме управления данными:

- пользователь;
- имя;
- телефон;
- адрес;
- электронная почта.

Интерфейс управления пользователями представлена на рисунке 26.

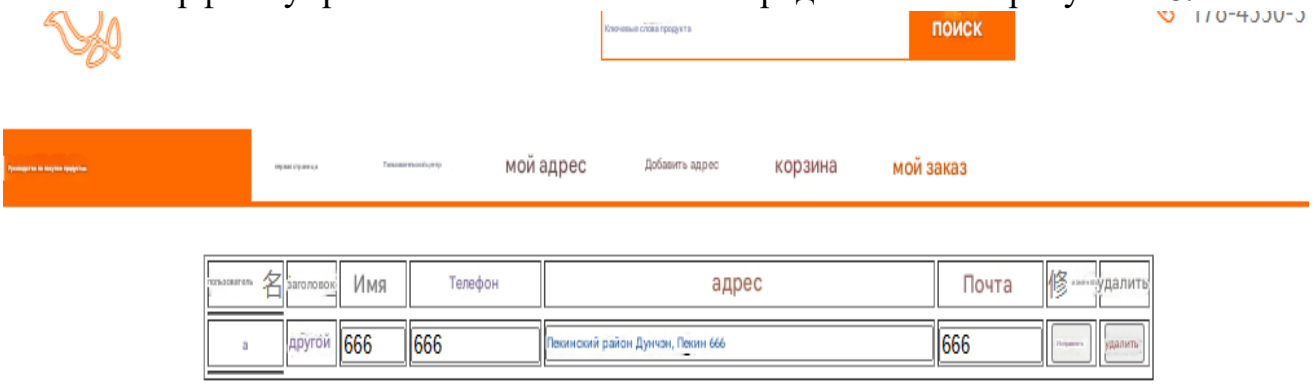

Рисунок 26 – Интерфейс управления пользователями

Схема интерфейса сообщений представляет систему для общения между заказчиком и компанией, выполняющей заказ. Также в этом интерфейсе можно задать интересующие вас вопросы или написать жалобу.

Интерфейс сообщений представлена на рисунке 27.

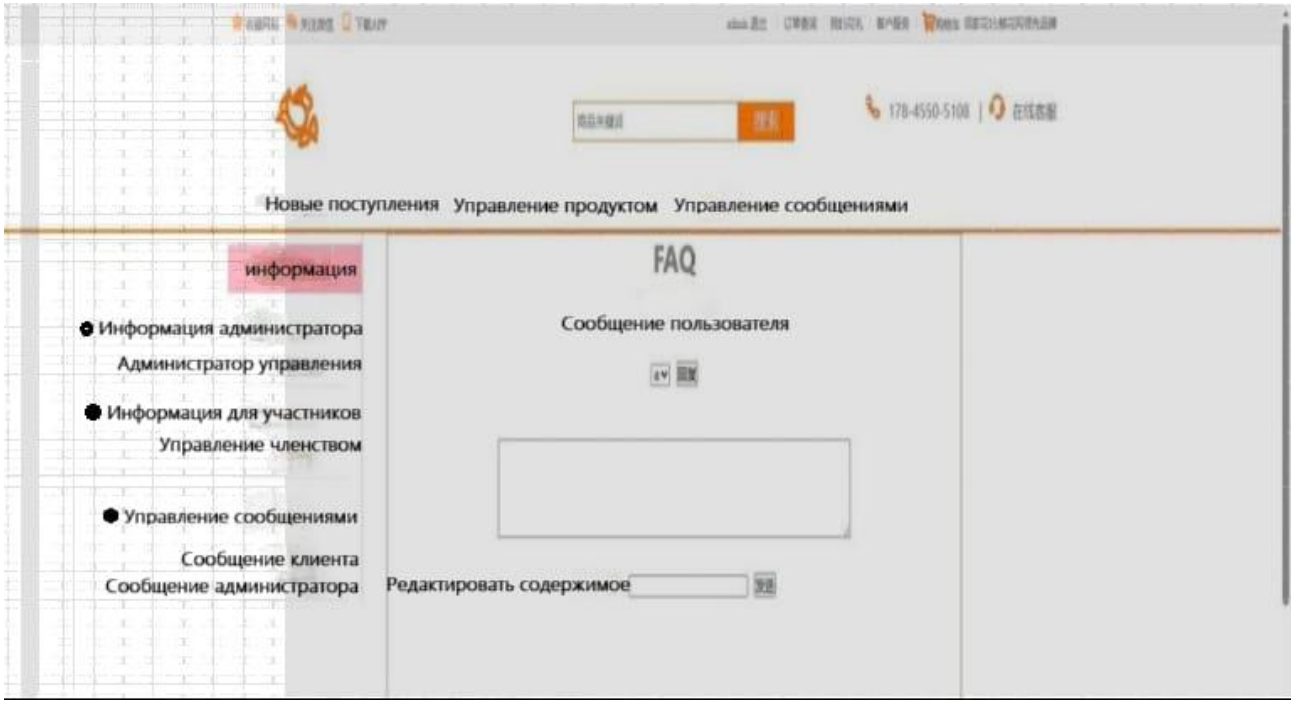

Рисунок 27 – Интерфейс сообщений

При большом количестве заказов в схеме интерфейса управления пользователями появляется несколько пользователей, что позволяет удобно просматривать количество заказчиков.

Интерфейс управления пользователями представлена на рисунке 28.

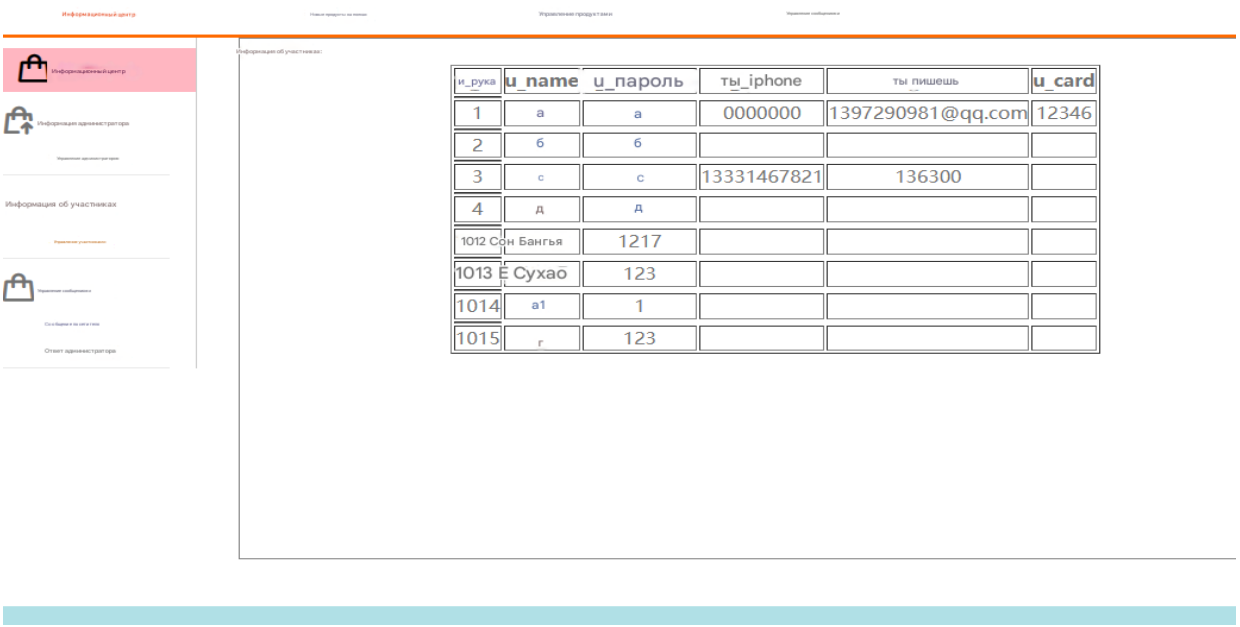

Рисунок 28 – Интерфейс управления пользователями

Блок-схема управления продукцией представляет собой алгоритм управления товарами с тремя основными операциями CRUD (Create, Read, Update, Delete), реализованный как:

- линейный процесс с явными точками принятия решений;
- разделённая логика (partition) для группировки операций;
- ветвление через if-then-else условия.

Пошаговое описание логики:

- проверка прав доступа перед выполнением операций;
- выбор операции (ключевое ветвление);
- три фундаментальные операции управления продуктом;
- подтверждение действия («Это действительно так?»);
- двойная проверка перед физическим удалением;
- альтернативный путь отмены;
- обязательная валидация изменений;
- полное заполнение атрибутов товара;
- контрольная точка подтверждения;
- обработка отмены операции.

Блок-схема программы функции управления продукцией представлена на рисунке 29.

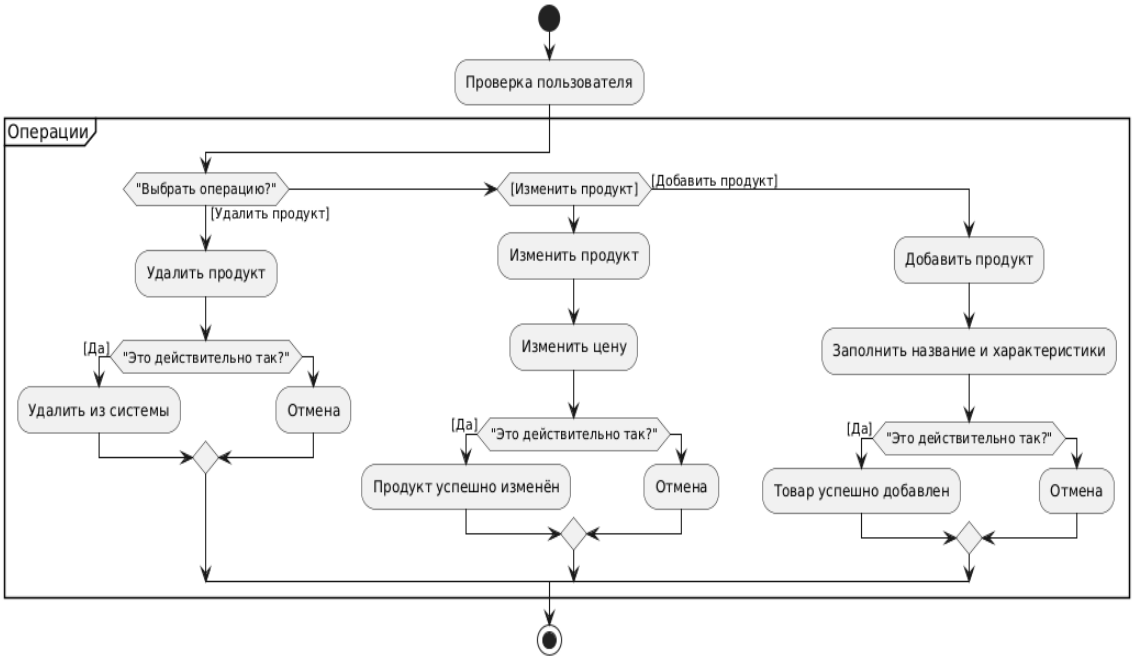

Рисунок 29 – Блок-схема программы функции управления продукцией

Интерфейс управления продуктами позволяет анализировать остатки товара и удобно считать сумму из нескольких букетов. Схема интерфейса управления продуктом представлена на рисунке 30.

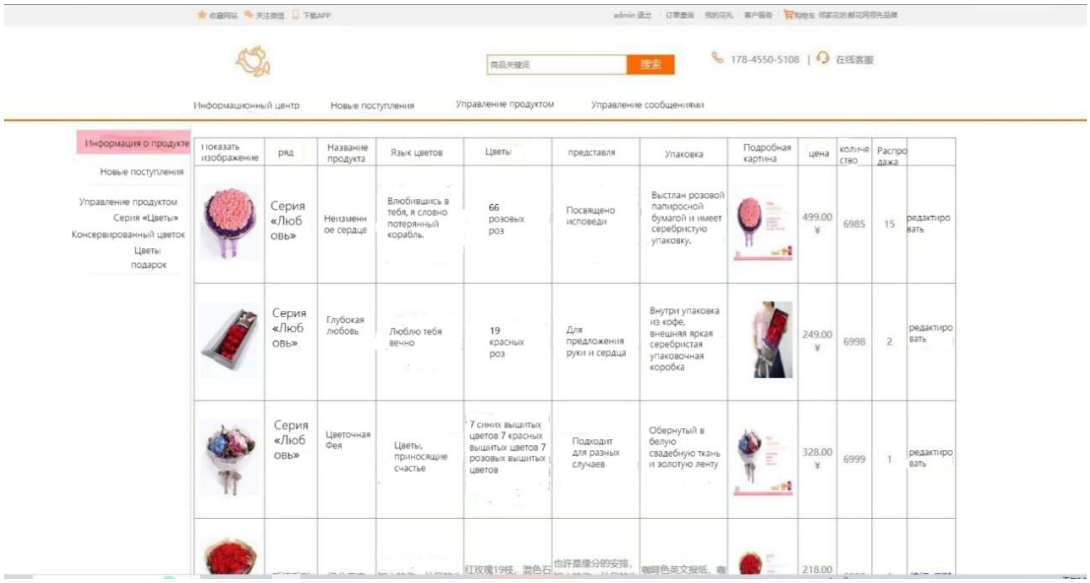

Рисунок 30 – Интерфейс управления продуктами

Пользовательский интерфейс системы Neighbor Flowers обладает проработанной структурой и базовой функциональностью, но требует доработок в следующих направлениях:

- удобство, более четкая обратная связь и навигация;

- безопасность, внедрение дополнительных проверок;

- производительность, оптимизация загрузки данных.

3.2.2 Тестирование пользовательского интерфейса

Тестирование веб-сайтов является основой для создания веб-сайтов с компьютерным программным обеспечением. Его можно использовать не только для проверки дизайна, но и для моделирования реальной среды клиента, чтобы проблемы, возникающие во время тестирования, могли быть обнаружены и решены своевременно, чтобы обеспечить надежность и наличие веб-сайта программного обеспечения.

Существует два основных метода тестирования «черного ящика», а именно «тестирование на прохождение» и «тестирование на провал». При проведении «проходного тестирования» фактически подтверждаются возможности программного обеспечения, но не тестируются его возможности. Тестировщики программного обеспечения используют только самые простые и интуитивно понятные тестовые примеры. При разработке и выполнении тестовых случаев всегда сначала проводите прохождение тестов. Прежде чем проводить разрушающее тестирование, проверьте, можно ли реализовать базовую функциональность программного обеспечения. Убедившись, что программное обеспечение работает правильно, вы можете использовать различные средства для поиска дефектов путем взлома программного обеспечения. Тестовые случаи, разработанные и выполненные исключительно для взлома программного обеспечения, называются тестами на отказ или тестами, вызывающими ошибки.

Тестирование белого ящика, также известное как структурная проверка, направлено на проверку возможных дефектов и недостатков в процессе написания программ. Тестирование «белого ящика» в основном включает в себя два различных метода тестирования: один анализ исходного кода, который описывает процесс работы программы путем наблюдения за исходным кодом; другой тестирование логического покрытия, которое проверяет каждую часть информации в исходном коде.

Модуль входа в систему в основном предназначен для пользователей и администраторов. После отработки нескольких тестовых примеров ожидаемые результаты тестирования, полученные после тестирования, показаны в Таблице 15.

Таблица 15 – Таблица тестовых сценариев модуля входа пользователя

№	Действие	Входные данные	Ожидаемый результат	Фактический результат
1	Ввод идентификатора без пароля	Логин: 789, Пароль: пусто	Ошибка входа	Соответствует
2	Ввод пароля без логина	Логин: пусто, Пароль: 789	Ошибка входа	Соответствует
3	Корректные данные	Логин: 789, Пароль: 789	Успешная авторизация	Соответствует

После многих практик тестирования результаты теста такие же, как и ожидалось, поэтому тест модуля входа в систему соответствует ожиданиям.

Модуль оценки в основном управляет опубликованными сообщениями. Тестовые примеры показаны в Таблице 16.

Таблица 16 –Тестовые примеры модуля оценки

Действие	Входные данные	Ожидаемый результат	Фактический результат
Отправка текста	«Тест»	Успех	Соответствует
Пустое сообщение	Пусто	Ошибка	Соответствует
Загрузка GIF-изображения	image.gif	Успех	Соответствует
Ввод смайлика	«:)»	Успех	Соответствует
Ввод спецсимвола	«☀»	Успех	Соответствует

Было введено несколько различных тестовых случаев, и результаты теста соответствовали ожиданиям. Таким образом, результаты теста модуля сообщений соответствовали ожиданиям.

Модуль личной информации в основном управляет вводом личной информации. Примеры испытаний показаны в Таблице 17.

Таблица 17 – Тестовые примеры модуля личной информации

Действие	Входные данные	Ожидаемый результат	Фактический результат
Пропуск логина	Пароль: 789456, Телефон: 789, Email: 789@qq.com	Ошибка	Соответствует
Пропуск пароля	Логин: 789, Телефон: 78789, Email: 787@qq.com	Ошибка	Соответствует
Пропуск телефона	Логин: 278, Пароль: 789, Email: 789@qq.com	Успех	Соответствует

При вводе разных данных документ не может быть введен. Остальные опции обязательны. Результаты теста такие же, как ожидалось. Тест модуля личной информации пройден.

Модуль изменения заказа в основном отвечает за управление добавлением, удалением и изменением продуктов.

Тестовые примеры показаны в Таблице 18.

Таблица 18 – Тестовые примеры модуля изменения заказа

Действие	Входные данные	Ожидаемый результат	Фактический результат
Удаление товара	Нажатие кнопки «Удалить»	Успех	Соответствует
Изменение цены на 50	Цена: 50	Успех	Соответствует
Попытка ввода отрицательной цены	Цена: -1	Ошибка	Соответствует

Модуль корзины корректно обрабатывает все базовые операции. Рекомендуется добавить тесты для:

- комбинаций товаров;

- промокодов;

- ограниченных предложений.

Тестирование модуля корзины представлено в таблице 19.

Таблица 19 – Тестирование модуля корзины

Действие	Входные данные	Ожидаемый результат	Фактический результат
Добавление 1 товара	ID товара: 123, Количество: 1	Корзина содержит 1 позицию	Соответствует
Добавление 10+ товаров	ID товара: 456, Количество: 12	Предупреждение о максимуме	Соответствует
Удаление товара	ID товара: 123	Корзина пуста	Соответствует
Обновление количества	ID товара: 456, Новое количество: 3	Итоговая сумма пересчитана	Соответствует

Платежная система корректно обрабатывает основные сценарии. Для улучшения рекомендуется:

- тестирование 3D Secure;

- проверка альтернативных платежных методов;

- тесты на время обработки транзакции.

Тестирование платежного модуля представлено в таблице 20.

Таблица 20 – Тестирование платежного модуля.

Действие	Входные данные	Ожидаемый результат	Фактический результат
Оплата картой	Номер: 4242 4242 4242 4242	Успешная оплата	Соответствует
Недостаточно средств	Номер: 4000 0000 0000 9995	Отказ в оплате	Соответствует
Неверный CVV	CVV: 999	Ошибка валидации	Соответствует
Просроченная карта	Месяц/год: 01/2020	Ошибка "Карта просрочена"	Соответствует

Обнаружена проблема с комбинированными фильтрами. Рекомендуется:

- исправить логику пересечения фильтров;

- добавить тесты для сложных запросов;

- проверить производительность при 5+ фильтрах.

Тестирование фильтрации товаров представлено в таблице 21.

Таблица 21 – Тестирование фильтрации товаров

Критерий фильтрации	Входные параметры	Ожидаемый результат	Фактический результат
По цене (мин-макс)	От 500 до 1000 ₽	Только товары в диапазоне	Соответствует
По типу цветов	"Розы"	Только розы	Соответствует
По назначению	"Для свадьбы"	Соответствующие композиции	Соответствует
Комбинированный	Цена 500-1000 + "Розы"	Розы в указанном ценовом диапазоне	Частично соответствует

Основные функции работают, но требуется доработка:

- адаптация меню для планшетов;

- оптимизация для разрешений 320-480 px;

- тестирование жестов (свайпы).

Тестирование мобильной версии представлено в таблице 22.

Таблица 22 – Тестирование мобильной версии

Устройство	Действие	Ожидаемый результат	Фактический результат
iPhone 13	Просмотр каталога	Корректное отображение	Соответствует
Samsung Galaxy S22	Оформление заказа	Адаптивный интерфейс	Соответствует
iPad Air	Редактирование профиля	Доступны все элементы	Проблемы с меню
Xiaomi Redmi Note 10	Фильтрация товаров	Работоспособность	Соответствует

Критические проблемы при высокой нагрузке. Рекомендации:

- оптимизировать запросы к БД;

- внедрить кэширование;

- настроить балансировку нагрузки.

Тестирование производительности представлено в таблице 23.

Таблица 23 – Тестирование производительности

Сценарий	Нагрузка	Ожидаемый результат	Фактический результат
Просмотр каталога	1000 RPS	Время отклика <1с	0.8 с
Оформление заказа	500 параллельных	Без ошибок	3 % ошибок
Поиск по сайту	200 RPS	Время отклика <2 с	2.4 с

Общие рекомендации:

- внедрить Selenium для регрессионного тестирования;

- настроить сбор метрик производительности;

- добавить тесты для восстановления пароля, истории заказов, системы лояльности;

- сохранять все тест-кейсы в TestRail или аналогичной системе.

Проведенное тестирование пользовательского интерфейса (UI) веб-платформы Neighbor Flowers позволило оценить удобство взаимодействия, функциональность и надежность системы. В ходе тестирования были проверены ключевые модули, включая управление продуктами, заказами, персональными данными и авторизацию.

Положительные аспекты:

- интерфейс логичен и прост в использовании. Пользователи легко ориентируются в разделах (личный кабинет, корзина, заказы);

- все CRUD-операции (добавление, редактирование, удаление) работают корректно;

- критические операции (удаление, изменение данных) требуют подтверждения, что снижает риск случайных ошибок;

- интерфейс корректно отображается на разных устройствах (ПК, планшеты, мобильные телефоны).

Выявленные проблемы:

- некоторые формы (например, добавление товара) не проверяют ввод на корректность (например, отрицательная цена);

- нет уведомлений об успешном выполнении операций (например, «Товар добавлен в корзину»);

- при редактировании профиля отсутствует подтверждение по SMS/email для критических изменений;

- некоторые страницы (например, список заказов) загружаются дольше ожидаемого.

Доработка UI/UX:

- добавить подсказки и валидацию для полей ввода (например, запрет ввода отрицательных цен);

- внедрить уведомления об успешных/неудачных операциях (тосты, pop-up);

- добавить хлебные крошки и кнопку «Назад» в ключевых разделах;

Повышение безопасности:

- ввести двухфакторную аутентификацию (2FA) для критических действий (удаление аккаунта, изменение email);

- реализовать маскирование данных (например, скрытие части номера телефона в профиле);

Оптимизация производительности:

- уменьшить время загрузки через кеширование и ленивую загрузку изображений;

- оптимизировать запросы к API для списков (например, пагинация заказов).

Дополнительное тестирование:

- провести юзабилити-тестирование с реальными пользователями для оценки удобства;

- реализовать нагрузочное тестирование для проверки стабильности при высокой посещаемости.

ЗАКЛЮЧЕНИЕ

В ходе проведения исследования изучены современные технологии разработки интернет-магазинов, реализовано создание веб-платформы для продажи цветов с использованием ASP.Net и Vue.js, обеспечивающее разнообразие выбора товаров.

Созданная веб-платформа охватывает ключевые аспекты взаимодействия с клиентами и управления бизнесом: регистрацию и авторизацию пользователей, каталог товаров с фильтрацией, корзину покупок, оформление и обработку заказов, а также административные функции по управлению товарами, пользователями, заказами и статистикой продаж.

В процессе разработки были применены методы объектно-ориентированного проектирования, разработаны структурные модули, реализована клиент-серверная архитектура, а также проведено тестирование, подтвердившее корректность функционирования платформы.

По результатам работы опубликовано 2 статьи в научном журнале «Флагман наук» (№11, №12, 2024 г).

Разработанный программный продукт продемонстрировал эффективность современных веб-технологий в решении прикладных задач электронной коммерции.

Созданная система охватывает ключевые аспекты взаимодействия с клиентами и управления бизнесом: регистрацию и авторизацию пользователей, каталог товаров с фильтрацией, корзину покупок, оформление и обработку заказов, а также административные функции по управлению товарами, пользователями, заказами и статистикой продаж. В процессе разработки были применены методы объектно-ориентированного проектирования, разработаны структурные модули, реализована клиент-серверная архитектура, а также проведено тестирование, подтвердившее корректность функционирования платформы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Бабичев, С. А. Разработка интернет-приложений: Учебное пособие / С. А. Бабичев. – М.: КноРус, 2022. – 312 с.

2 Бройдо, В. Л. Инженерия программного обеспечения: Учебное пособие / В. Л. Бройдо, Ю. А. Кузнецов. – СПб.: Питер, 2019. – 336 с.

3 Ван Лэй. Принципы проектирования интерфейсов во Vue.js // Интерфейсы и взаимодействие. – 2022. – №6.

4 Ван Хао. Использование JWT-токенов для аутентификации пользователей в ASP.Net // Информационная безопасность. – 2023. – №10.

5 Гао Тин. Повышение производительности веб-приложений на ASP.Net // Высоконагруженные системы. – 2021. – №5.

6 Глушков, И. П. Основы проектирования информационных систем: Учебник для вузов / И. П. Глушков, А. С. Васильев. – М.: Академия, 2019. – 412 с.

7 Го Синь. Обработка заказов и управление корзиной покупок в e-commerce системе // Электронная торговля. – 2021. – №9.

8 Евсеев, Г. П. Информационные технологии: проектирование и разработка приложений / Г. П. Евсеев. – М.: Юрайт, 2021. – 450 с.

9 Захаров, В. П. Проектирование человеко-машинных интерфейсов / В. П. Захаров. – СПб.: БХВ-Петербург, 2022. – 276 с.

10 Зиновьев, Д. В. Прототипирование интерфейсов и UX-дизайн: Учебник / Д. В. Зиновьев. – М.: ДМК Пресс, 2021. – 288 с.

11 Коряковцев, Е. А. Технологии проектирования веб-приложений: Учебное пособие / Е. А. Коряковцев. – М.: Форум, 2020. – 280 с.

12 Кузнецов, Д. И. Системы управления базами данных: проектирование и реализация / Д. И. Кузнецов. – М.: Форум, 2020. – 280 с.

13 Кузнецов, С. И. Технологии разработки программного обеспечения: Учебное пособие / С. И. Кузнецов. – М.: Юрайт, 2022. – 384 с.

14 Лапшин, С. А. Архитектура программных систем: Учебное пособие / С. А. Лапшин. – СПб.: Питер, 2020. – 352 с.

15 Ли Мэй. Архитектура клиент-сервер для онлайн-магазина на ASP.Net // Программная инженерия. – 2023. – №5.

16 Ли Тин. Примеры реализации административных панелей с использованием Vue.js и REST API // Управление проектами. – 2022. – №10.

17 Ли Хун. Разработка системы новостей и отзывов на платформе ASP.Net // Автоматизация и ПО. – 2022. – №6.

18 Литвинов, Ю. А. Методология и практика программной инженерии / Ю. А. Литвинов. – М.: Юрайт, 2021. – 360 с.

19 Ло Жуй. Реализация модулей авторизации и регистрации на ASP.Net // Информационная безопасность систем. – 2022. – №10.

20 Лю Вэй. Безопасность данных в ASP.Net приложениях: методы защиты и аутентификации // Информационная безопасность. – 2023. – №6.

21 Лю Мин. Разработка мультистраничных интерфейсов с помощью Vue Router // Разработка веб-приложений. – 2023. – №2.

22 Микулин, А. Н. Программирование на платформе .NET: Учебник / А. Н. Микулин. – СПб.: БХВ-Петербург, 2019. – 384 с.

23 Павлов, С. А. Разработка веб-приложений: Учебное пособие / С. А. Павлов. – М.: КноРус, 2020. – 328 с.

24 Панкратов, В. Н. Информационные системы в экономике и управлении: проектирование и внедрение / В. Н. Панкратов. – М.: КноРус, 2021. – 296 с.

25 Сапожников, В. А. Разработка и тестирование программного обеспечения: Учебное пособие / В. А. Сапожников. – М.: Инфра-М, 2021. – 304 с.

26 Сендов, В. Я. Проектирование программного обеспечения: Учебное пособие / В. Я. Сендов. – М.: Форум, 2021. – 336 с.

27 Симонов, Е. Б. Web-программирование: технологии, протоколы, интерфейсы / Е. Б. Симонов. – М.: ДМК Пресс, 2019. – 298 с.

28 Соловьев, В. Д. Основы программной инженерии: Учебник / В. Д. Соловьев. – М.: Юрайт, 2021. – 468 с.

29 Титов, В. А. Проектирование клиент-серверных приложений: Учебник / В. А. Титов. – СПб.: БХВ-Петербург, 2018. – 296 с.

30 Цзинь Хуа. Разработка платформы онлайн-продаж с использованием ASP.Net и Vue.js // Вестник цифровых решений. – 2023. – №3.

31 Чернышев, Ю. Н. Разработка приложений на ASP.NET Core / Ю. Н. Чернышев. – СПб.: Питер, 2021. – 336 с.

32 Чжан Вэй. Использование фреймворка Vue.js при создании пользовательских интерфейсов // Информационные технологии. – 2021. – №9.

33 Чжан Хао. Интеграция Vue.js с ASP.Net Web API для построения интерактивных интерфейсов // Программная интеграция. – 2023. – №4.

34 Чэнь Линь. Организация логирования и отслеживания ошибок в ASP.Net Core // Надежность информационных систем. – 2022. – №3.

35 Чэнь Хэ. Создание адаптивных веб-интерфейсов с использованием Vue.js // Современные веб-технологии. – 2021. – №12.

36 Шалаев, И. А. Основы веб-программирования на JavaScript и Vue.js / И. А. Шалаев. – М.: ДМК Пресс, 2021. – 310 с.

37 Шевчук, А. В. Проектирование и реализация программных систем: Учебник / А. В. Шевчук. – М.: Академия, 2020. – 392 с.

38 Яковлев, Н. В. Современные веб-технологии: ASP.NET и JavaScript: Учебное пособие / Н. В. Яковлев. – СПб.: Питер, 2020. – 288 с.

39 Ян Лэй. Оптимизация загрузки данных в SPA-приложениях на Vue.js // Электронные вычисления. – 2021. – №6.

40 Ян Чжун. Реализация системы учёта заказов в ASP.Net Web API // Программирование и автоматизация. – 2022. – №8.

41 Ян Шуан. Использование компонентов Element UI во Vue.js приложениях // Технологии фронтенд-разработки. – 2021. – №10.

42 Амурский государственный университет [Электронный ресурс]. – Режим доступа: https://amursu.ru/. – 03.03.2025.

43 CodinGame [Электронный ресурс]. – Режим доступа: https://codingame.com/. – 15.04.2025.

44 JavaRush [Электронный ресурс]. – Режим доступа: https://javarush.ru/. – 22.02.2025.