МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра Прикладная математика и информатика
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки / специальности)

Разработка программного обеспечения
(направленность (профиль) / специализация)

# ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка функционального веб-приложения для онлайн-магазина автозапчастей»

Обучающийся Ф.В. Серов
(Инициалы Фамилия) (личная подпись)

Руководитель Н.Н. Казаченок
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант М.В Дайнеко
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

## Аннотация

Название выпускной квалификационной работы: «Разработка функционального веб-приложения для интернет-магазина автозапчастей».

Целью исследования является разработка веб-приложения, автоматизирующего процессы поиска и заказа автозапчастей, размещения заказов и взаимодействия с клиентами компании. Объектом исследования являются процессы автоматизации в работе интернет-магазина автозапчастей. Предметом исследования является проектирование и реализация масштабируемого веб-приложения с использованием современных технологических решений.

Выпускная квалификационная работа включает несколько логически связанных глав. В первой главе проводится анализ бизнес-среды и определяются требования к автоматизации с учетом специфики онлайн-торговли автозапчастями. Во второй главе описываются архитектурные решения, структура базы данных и реализация модулей: управление каталогом, управление корзиной покупок, управление заказами и управление отзывами клиентов. В третьей главе рассматриваются результаты тестирования системы, анализируется сокращение ручных операций и частота ошибок. В этой главе дается оценка перспектив дальнейшего развития.

Практическая значимость работы заключается в возможности использования разработанного приложения для оптимизации работы магазинов автозапчастей. Полученные результаты могут быть интересны специалистам, занимающимся разработкой и внедрением информационных систем электронной коммерции.

В заключение следует подчеркнуть, что внедрение разработанного приложения позволило значительно сократить время, необходимое для обработки заказов, и повысить прозрачность взаимодействия с клиентами.

## Abstract

The title of the graduation work is Developing a functional web application for an online auto parts store.

The aim of this graduation work is to develop a web application that automates the processes of searching for and ordering auto parts, placing orders, and interacting with the company's customers. The object of the research is the automation processes in the operation of the online auto parts store. The subject of the study is the design and the implementation of the scalable web application using the modern technological solutions.

The graduation work may be divided into several logically connected parts. The first part carries out the business environment analysis and defines the requirements for automation taking into account the specifics of online auto parts trade. The second part describes the architectural solutions, the database structure and the modules implementation: catalogue management, shopping cart management, order management, and customer feedback management. The third part dwells on the system testing results, analyzes the reduction of the manual operations and the error rate. In this part, the prospects for further development is accessed.

The practical significance of the work consists in the opportunity of using the developed application for optimizing the auto parts stores' operations. The results obtained may be of interest for IT specialists engaged in the development and implementation of e-commerce information systems.

In conclusion, it should be highlighted that the implementation of the developed application has made it possible to significantly reduce the time required for the order processing and to improve the transparency of the interactions with the customers.

**Оглавление**

# Введение

Информационное технологии стали неотъемлемой частью бизнеса, чтобы конкурировать на рынке электронной коммерции, компании создают свои собственные продукты, направленные на автоматизацию бизнес-процессов и удовлетворение клиента.

Современным пользователям необходима реализация множества факторов для выбора продукта. Чтобы обеспечить оперативное взаимодействие с клиентом, настроить эффективное управление товарами и упростить работу менеджеров было принято решение о создании собственной информационной системы.

Главным требованием создания собственного продукта стало использование современных технологий, позволяющих создать эффективное, масштабируемое и удобное функциональное веб-приложение, решающее все задачи. Для этого в создании использовались современные фреймворки React[12] и Express[14], [20], а также надежная база данных PostgreSQL[18], которые соответствуют всем требованиям и позволяют реализовать необходимый продукт.

Таким образом, разработка и внедрение инновационного программного обеспечения для интернет-магазина автозапчастей является не только технически актуальной, но и экономически обоснованной. Эффективное использование информационных систем способствует снижению издержек, повышению оперативности принятия решений и улучшению качества сервиса.

Объект исследования – процессы коммерческой деятельности по продаже автозапчастей в электронной среде, а именно комплексная система организации и автоматизации торговых операций в интернет-магазине автозапчастей.

Предметом работы является разработка функционального веб-приложения для онлайн-магазина автозапчастей. В рамках исследования рассматриваются следующие аспекты:

- многоуровневое архитектурное моделирование (выбор оптимальной структуры клиент-серверных взаимодействий и микросервисов);
- формализация функциональных требований посредством UML-диаграмм вариантов использования, классов и последовательностей;
- разработка и отладка REST API для интеграции с платежными и складскими системами;
- построение логической и физической моделей базы данных с учётом нормализации и индексации для обеспечения высокой производительности.

Целью работы является разработка надёжного, масштабируемого и комфортного в эксплуатации веб-приложения «NVS-CAR», способного обеспечить оперативный поиск автозапчастей, удобную процедуру оформления заказов, а также информационную поддержку пользователей и администраторов.

Задачи работы:

- провести всесторонний анализ рынка автозапчастей, выявить ключевые потребности бизнеса и сформировать набор требований к автоматизированной системе;
- исследовать и сравнить существующие решения на рынке электронной коммерции (Magento, WooCommerce, OpenCart и др.), оценить их архитектурные достоинства и ограничения для обоснованного выбора собственного технологического стека;
- составить полный перечень функциональных и нефункциональных требований к веб-приложению на основе методологии FURPS+ и задокументировать их в UML-диаграммах;
- спроектировать логическую структуру базы данных, определить ключевые сущности и связи, а также оптимизировать модель для обеспечения высокой скорости запросов и сохранности данных;

– реализовать фронтенд-часть на React[23] и бэкенд-часть на Node.js/Express, обеспечив надёжное взаимодействие между ними и внешними сервисами через REST API;
– провести комплексное тестирование (юнит-, интеграционное и нагрузочное) для оценки стабильности, производительности и удобства интерфейса веб-приложения.

Выпускная квалификационная работа состоит из введения, трех основных глав, заключения, списка использованных источников и приложений.

В первой главе рассматривается теоретическая и прикладная база исследования. В частности, проводится анализ современных бизнес-процессов в сфере электронной коммерции автозапчастей, приводится характеристика предметной области, обосновывается актуальность разработки, а также анализируются существующие программные решения.

Вторая глава посвящена проектированию и технической реализации веб-приложения. Здесь подробно раскрываются этапы архитектурного моделирования системы, описывается выбор и обоснование технологического стека, приводится построение логической и физической моделей базы данных, а также формализация бизнес-логики на основе UML-диаграмм.

Третья глава направлена на оценку эффективности функционирования разработанной системы. В данной части приводится описание проведенного тестирования, включая проверку корректности реализации функциональных требований, анализируется производительность работы.

Общий объем работы составляет 69 страниц, материал сопровождается 41 рисунками и схемами, а также 13 таблицами. В процессе подготовки исследования было использовано 23 источника отечественной и зарубежной научной литературы.

# Глава 1 Постановка задачи на разработку программного обеспечения для информационной системы предприятия

## 1.1 Значение программного обеспечения для разработки функционального веб-приложения для онлайн-магазина автозапчастей

В предметной области электронной коммерции, в частности рынка автозапчастей, существует значительная проблема, связанная с недостаточной эффективностью традиционных методов управления товарами, заказами и логистикой. Недостатки таких подходов включают высокие операционные затраты, низкую скорость обработки данных, ошибки при управлении складскими запасами и снижение удовлетворённости пользователей.

Для наглядности была построена диаграмма Ишикавы (причина-следствие) (рисунок 1):

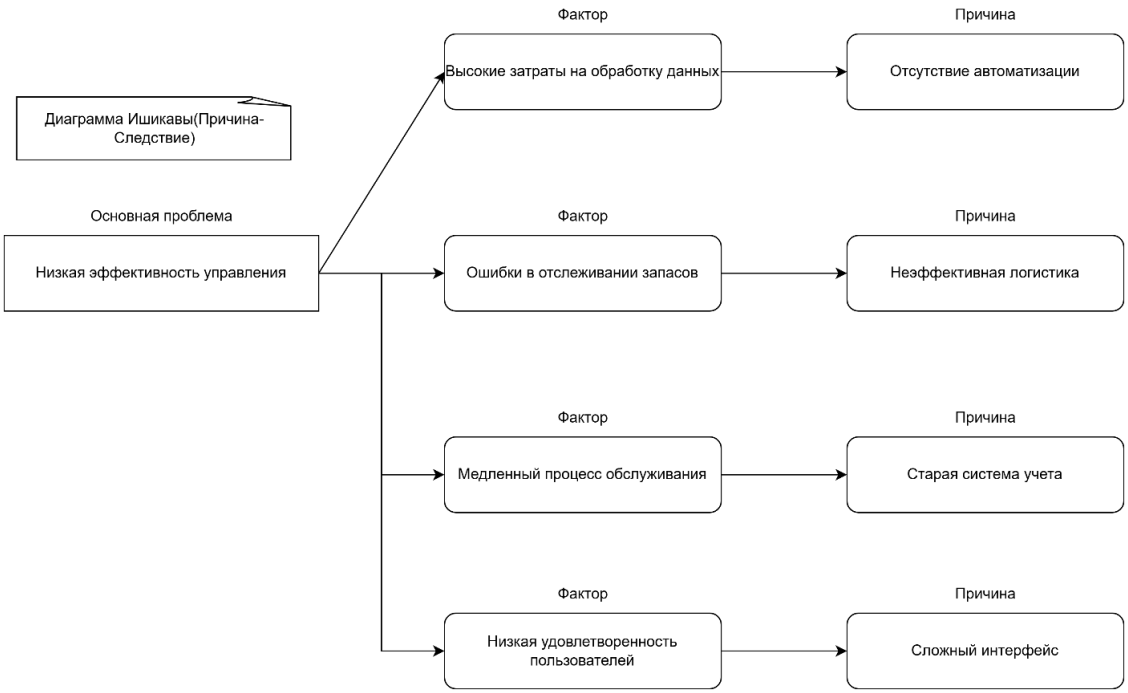


Рисунок 1 – Диаграмма Ишикавы (Причина-Следствие)

Использование специализированного программного обеспечения позволяет устранить указанные ограничения[4], [8]. Внедрение автоматизированной информационной системы, такой как веб-приложение «NVS-CAR», существенно улучшает оперативность обработки заказов, уменьшает вероятность ошибок и обеспечивает высокий уровень клиентского сервиса за счет современных технологий (REST API, WebSocket). Например, внедрение аналогичных систем показало снижение операционных затрат до 30%, ускорение обработки заказов в среднем на 40% и рост удовлетворенности клиентов на 25%.

Ниже представлен график изменения ключевых показателей (рисунок 2):

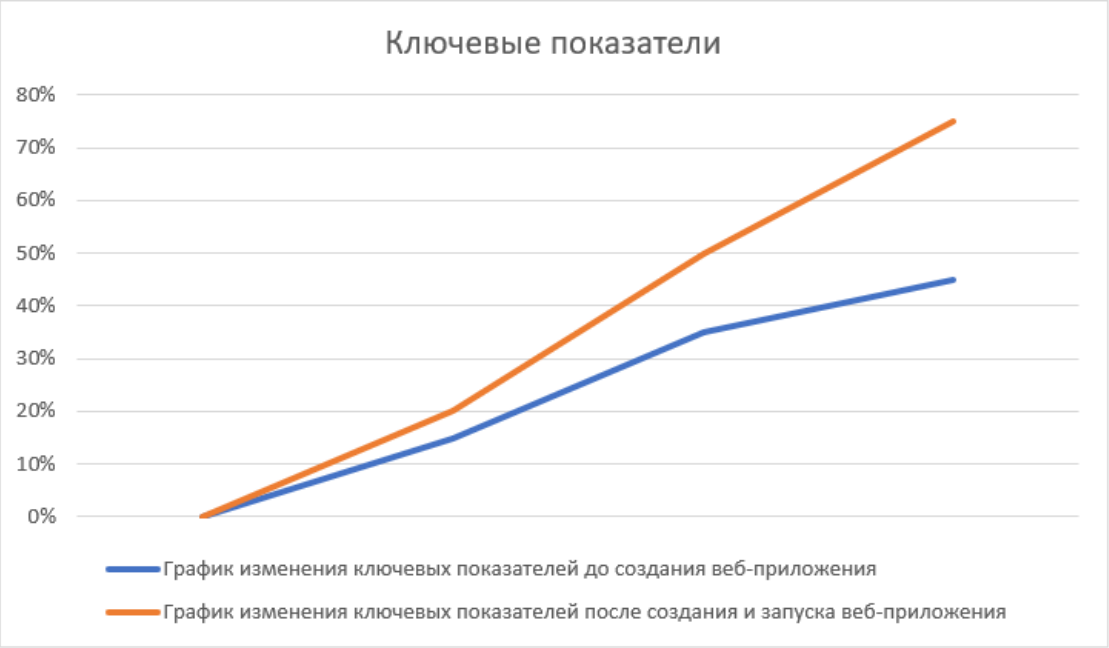


Рисунок 2 – Изменение ключевых показателей

Разработка программного обеспечения занимает центральное место в процессе цифровой трансформации предприятий электронной коммерции. По мере увеличения объёмов данных и количества операций возрастает потребность в эффективных инструментах автоматизации бизнес-процессов.

Разработка специализированных модулей, таких как электронный каталог товаров, корзина заказов, система отзывов и административная панель, обеспечивает удобство и прозрачность взаимодействия как для клиентов, так и для сотрудников. Повышается качество клиентского обслуживания за счёт автоматизации обработки обращений, отслеживания статусов заказов и реализации функций обратной связи. Надежная база данных позволяет хранить огромное количество данных в безопасном месте.

Результаты внедрения программного обеспечения напрямую отражаются на экономических показателях предприятия: сокращаются издержки, уменьшается влияние человеческого фактора, ускоряется обработка информации.

В таблице 1 представлены основные преимущества компаний после разработки и внедрения специализированного веб-приложения для интернет-магазина автозапчастей.

Таблица 1 – Достижения компаний, приобретенные после разработки веб-приложения

| Онлайн-магазин | Ключевые достижения после разработки собственного веб-приложения |
|---|---|
| Exist.ru | Обновлённый веб-интерфейс с расширенным поиском (по VIN, марке, категории) повысил конверсию заказов на 20%<br>Сокращено время обработки заявок благодаря автоматизации дооформления и проверки наличия<br>Развитая программа лояльности в личном кабинете увеличила повторные продажи на 15% |
| AutoDoc (Европа) | Модернизированная платформа с логикой рекомендаций (AI-driven) помогла сократить поиск нужных деталей на 30%<br>Оптимизированный фронтенд (React + SPA) ускорил загрузку страниц, снизив показатель отказов (bounce rate)<br>Интеграция мобильного приложения дала прирост мобильных заказов на 25% |

Краткие пояснения к достижениям:

Exist.ru**:**

– расширенный поиск (по VIN, номеру детали, марке/модели) избавил клиентов от ручного сравнения, тем самым повышая конверсию заказов;
– интеграция автоматических сценариев при оформлении заказов ускорила обработку на стороне склада;
– программа лояльности (личный кабинет с бонусами и скидками) стимулирует клиентов покупать повторно и рекомендовать сервис знакомым.

AutoDoc**:**

– модуль рекомендаций, основанный на данных о покупках, позволяет быстрее находить нужный товар и подсказывает аналогичные детали;
– скорость загрузки (за счёт SPA и оптимизации серверной части) снизила процент отказов: клиенты меньше покидают сайт из-за долгих загрузок;
– мобильное приложение упростило поиск и покупку деталей в дороге, что усилило присутствие бренда среди автомобилистов.

Таким образом, разработка программного обеспечения является неотъемлемой частью современного мира, обеспечивая решение актуальных проблем предприятий в выбранной предметной области. Это не только способствует повышению их конкурентоспособности, но и открывает новые возможности для устойчивого экономического и социального развития.

## 1.2 Основные функции программного обеспечения для информационной системы предприятия

Веб-приложение интернет-магазина автозапчастей «NVS-CAR» обеспечивает решение множества задач, которые включают в себя как поиск нужной детали, так и управление складом, а структура его функций во многом определяется спецификой автосервиса и торговли запчастями. Все функции системы можно условно разделить на три группы:

– обработка данных: сбор информации о наличии деталей, ценах, характеристиках, заказах и отзывах клиентов; её хранение в базе данных и последующий анализ для формирования рекомендаций и прогнозов спроса;
– взаимодействие с пользователями: удобный интерфейс поиска и фильтрации автозапчастей, оформление и отслеживание заказов, система уведомлений об изменениях статуса заказа, адаптивный дизайн под разные устройства;
– автоматизация процессов: приём заказов без прямой работы менеджеров, автоматические уведомления при заказе, занесение данных заказа в базу данных.

Обработка данных в веб-приложении – это полноценный цикл работ, обеспечивающий точность и своевременность всей информационной составляющей бизнеса. Сбор данных производится сразу при поступлении новых товаров на склад (артикул, партия, цена), при каждом оформлении заказа клиентом (модель автомобиля, VIN-номер, количество деталей) и при каждом изменении статуса (оплачен, отгружен, доставлен). Хранение данных организовано в реляционной базе, где таблицы «Товары», «Заказы», «Клиенты» и «Поставщики» связаны между собой через ключевые поля; продумана система резервного копирования и восстановления, чтобы исключить потерю критической информации.

Используя правильный и прозрачный процесс обработки данных, функциональное веб-приложение для компании «NVS-CAR», обеспечивает не только точность учёта и скорость принятия решений, но и возможность гибко масштабироваться по мере роста ассортимента и клиентской базы.

Программное обеспечение (ПО) воплощает в себе совокупность инструментов и алгоритмов, предназначенных для решения конкретных задач в пределах определённой предметной области. Эти задачи могут существенно варьироваться: от элементарной обработки текстов до формирования сложных аналитических моделей.

В секторе электронных продаж автозапчастей всё перечисленное приобретает особую значимость. Правильно выстроенный программный комплекс способен не только оптимизировать работу склада и упростить поиск требуемых деталей, но и повысить удовлетворённость клиентов, одновременно обеспечивая надёжную защиту конфиденциальной информации.

Важнейшей обязанностью ПО остаётся обращение с данными, которое состоит из последовательных этапов:

- сбор – поступление сведений от разнообразных источников, включая партнёров-поставщиков, клиентские заявки и внешние сервисы;
- анализ – фильтрация, классификация, статистическая обработка, а также применение методик интеллектуального анализа (data mining);
- хранение – помещение структурированных или полуструктурированных данных в специально выбранную базу (реляционную либо NoSQL);
- извлечение – предоставление нужной информации в удобном для пользователя формате, будь то вкладка «Список запчастей», отчёт о продажах или рекомендация для конкретного клиента.

Чтобы наглядно представить описанные стадии, построена блок-схема (рисунок 3), отражающая последовательное перемещение информации. К примеру, данные о ценах и наличии деталей поступают от поставщиков, затем проходят валидацию и загружаются в базу, а впоследствии выдаются клиентам в форме веб-страниц поиска или через API.

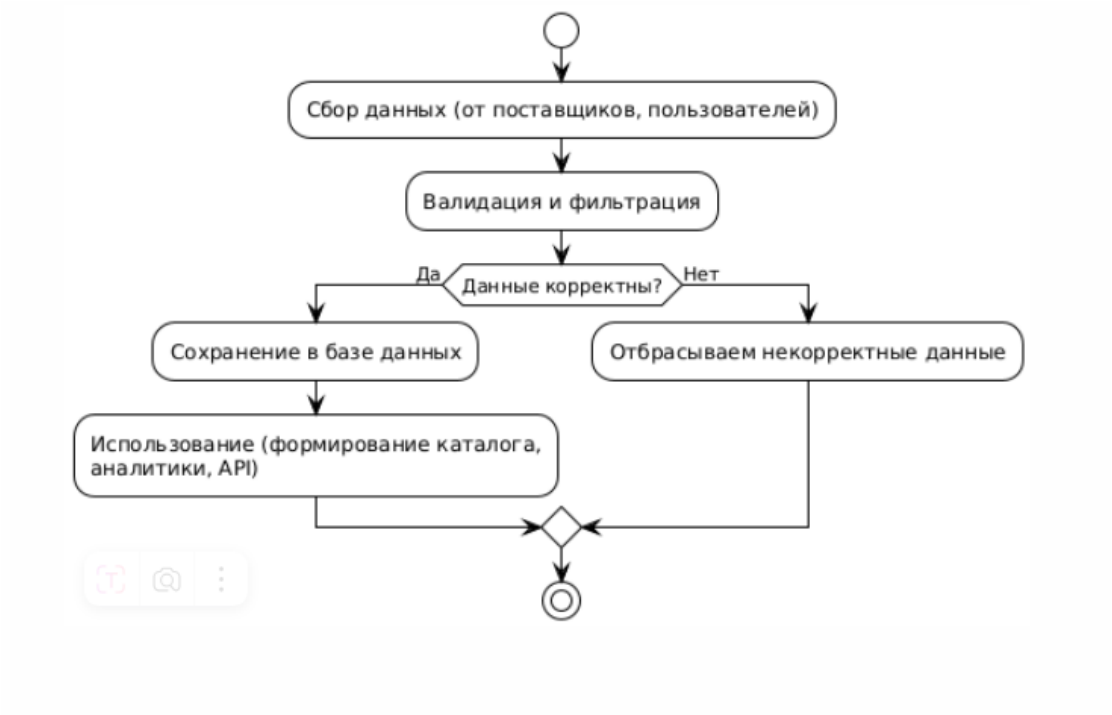


Рисунок 3 – Схема потока данных

Далее можно выделить функцию управления ресурсами: любая организация стремится рационально распределять финансы, рабочую силу и материальные активы. Использование специализированных систем (например, ERP, CRM или WMS) позволяет:

- уменьшить издержки и избежать простоя из-за нехватки запчастей;
- повысить прозрачность бизнес-процессов благодаря централизованным отчётам;
- систематизировать информацию о поставках, заказах и возвратах
- улучшить общее управление ресурсами

Таблица 2 с примерами систем управления, которая наглядно показывает виды специализированных систем, которые повсеместно используются во множестве компаний и обеспечивают улучшение ключевых показателей показана ниже:

Таблица 2 – Системы управления

| Название системы | Основное назначение | Ключевые выгоды |
|---|---|---|
| ERP-система | Комплексное управление ресурсами (финансы, склад, персонал) | Централизованный контроль учётных данных и ускоренное принятие решений |
| CRM-платформа | Взаимодействие с клиентами, анализ их предпочтений | Личный кабинет покупателя, сбор отзывов, настройка программ лояльности |
| WMS (управление складом) | Координация движения товаров по складу | Сокращение времени комплектации заказов, оптимизация складских остатков |

В контексте онлайн-магазинов автозапчастей грамотная интеграция подобных систем позволяет улучшить доступность нужных деталей и одновременно снизить затраты на избыточные позиции.

Веб-приложение для компании «NVS-CAR» предусматривает создание интерфейса, оптимизированного для эффективного выполнения основных пользовательских сценариев: поиска запчастей и оформления заказов. Панель с фильтрами (год выпуска, диапазон цен, производитель) реализована с мгновенной реакцией: при изменении параметров данные обновляются без перезагрузки страницы, что повышает оперативность подбора и снижает когнитивную нагрузку.

Процесс оформления заказа разделён на логические этапы, каждый из которых сопровождается валидацией вводимых данных в реальном времени.

Адаптивность интерфейса достигается за счёт использования гибкой сеточной раскладки и медиазапросов CSS, что гарантирует сохранение функциональности и читабельности на экранах различного размера.

Наглядная иллюстрация взаимодействия пользователя с различными компонентами системы показана ниже (рисунок 4).

При работе с корзиной последовательность действий аналогична: инициируется запрос на добавление товара, что приводит к обновлению

соответствующих записей в базе, после чего пользователю отправляется подтверждение.

Во время оформления заказа фиксируется аналогичная структура: данные проходят через серверную логику, происходит регистрация заказа в базе и возврат идентификатора операции на клиентскую сторону.

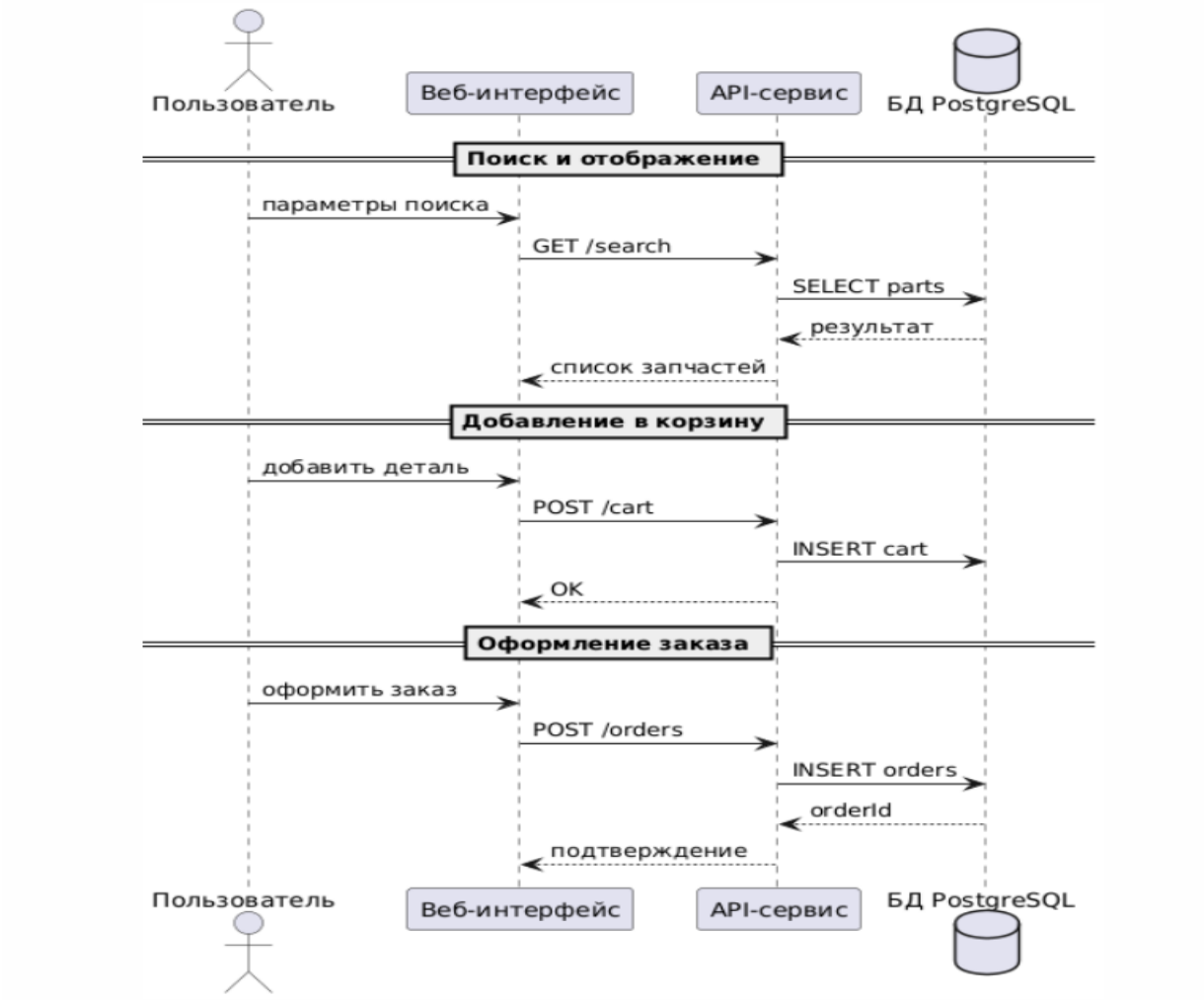


Рисунок 4 – Диаграмма взаимодействия

График оптимизации задач показан ниже (рисунок 5):

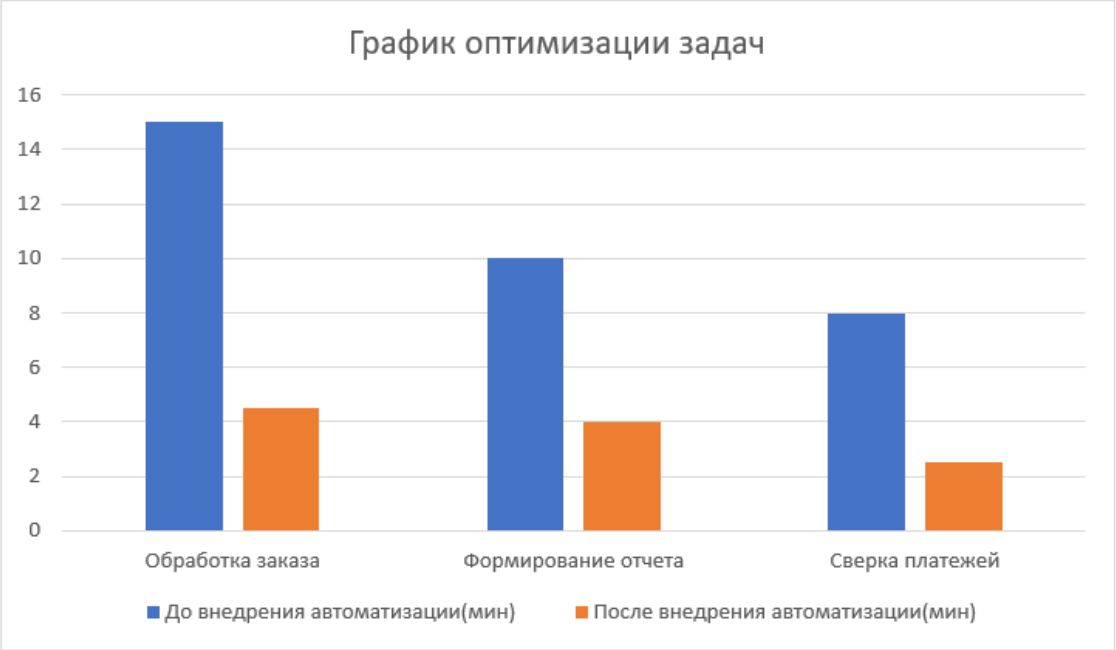


Рисунок 5 – График оптимизации задач

В условиях роста киберугроз любой магазин автозапчастей, взаимодействующий со множеством пользователей, обязан уделять пристальное внимание безопасности:

– аутентификация и авторизация: недопущение доступа посторонних лиц в административную панель;
– шифрование передаваемых данных: использование HTTPS-протокола (TLS) для защиты учётных записей и платёжных реквизитов;
– многоуровневая модель защиты: от межсетевых экранов на физическом уровне до регулярного бэкапа и контроля целостности на прикладном.

Ниже для наглядности представлена модель многослойной безопасности (рисунок 6):

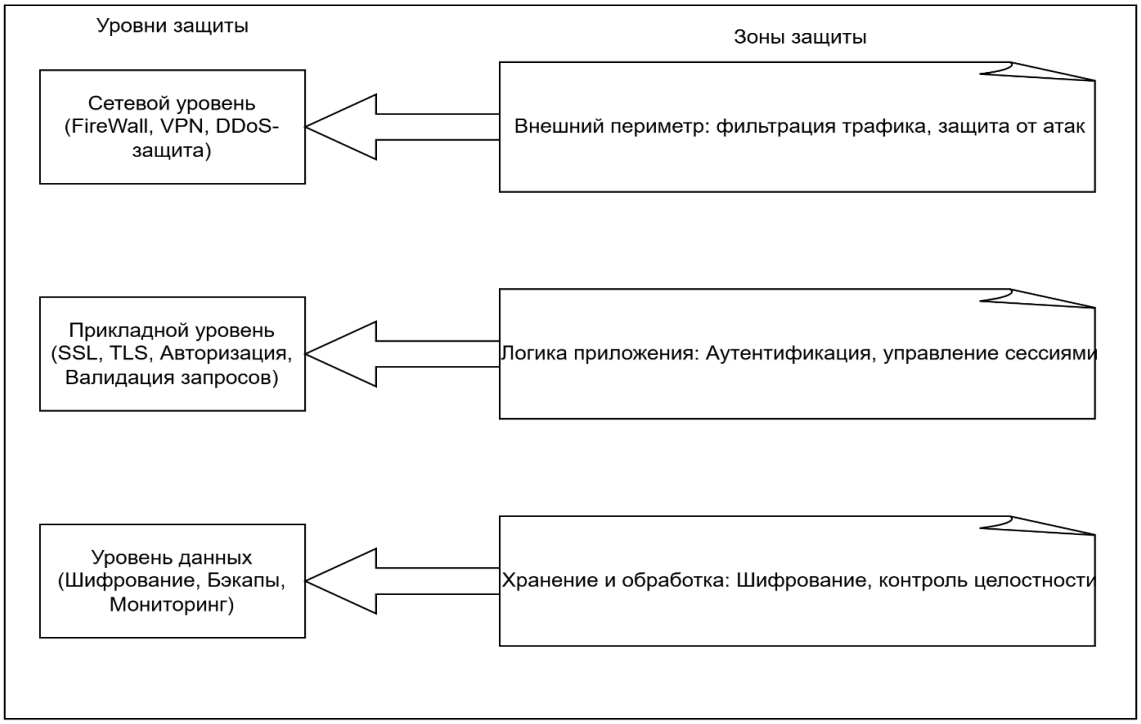


Рисунок 6 – Модель многослойной безопасности.

Пять основных функций программного обеспечения (обработка данных, управление ресурсами, удобный интерфейс, автоматизация рутины и безопасность) формируют «каркас», на котором держится эффективность и надёжность информационной системы. Для интернет-магазина автозапчастей подобная функциональная модель критически важна, поскольку помогает:

- упорядочить процессы складского учёта, логистики и взаимодействия с клиентами;
- повысить оборачиваемость и обороты продаж за счёт удобства и быстродействия веб-приложения;
- защитить персональные данные и финансовую информацию клиентов в условиях растущих сетевых угроз.

Таким образом, понимание и грамотная реализация ключевых функций ПО позволяют достичь конкурентоспособности и развития интернет-магазинов в автомобильной сфере.

## 1.3 Обзор существующих решений и их анализ

В данном разделе представлен сравнительный анализ существующих платформ для организации интернет-магазинов, который даёт возможность оценить достоинства и недостатки готовых решений по сравнению с разработкой собственного программного продукта[6].

Magento является одной из наиболее развитых и функциональных платформ в области электронной коммерции. Она предоставляет широкий спектр возможностей, таких как гибкое управление ассортиментом товаров, мощные средства аналитики и маркетинговые инструменты, поддержку множества языков и валют. Однако платформа требует значительных временных и финансовых ресурсов для её внедрения, настройки и последующего технического сопровождения.

WooCommerce является решением, встроенным в систему WordPress, которая позволяет создавать сайты и настривать их. Преимуществами данного решения является возможность создать быстро и с минимальными навыками создать собственный веб-сайт, однако множество возможностей собственного продукта не получится реализовать, используя WooCommerce.

OpenCart отлично подходит для создания небольших онлайн-магазинов, но главным недостатком платформы является сильная ограниченность в количестве пользователей и при большом количестве трафика, система будет работать нестабильно.

Для систематической оценки различных платформ рассмотрим пять основных критериев:

- функциональность – насколько широко реализованы инструменты управления товарами, заказами, маркетингом;
- стоимость – включает лицензионные платежи, расходы на внедрение, а также техническое сопровождение;
- удобство использования – комфорт для владельцев, контент-менеджеров и конечных пользователей;

– поддержка – помощь сообщества и специалистов, наличие документации, форумов, а также официальные сервисные пакеты;
– безопасность – механизмы защиты данных, обновления для устранения уязвимостей, совместимость с современными протоколами шифрования.

Ниже представлена таблица 3 сравнительного анализа (условная, в баллах от 1 до 5), дополненная с учётом специфики Magento, WooCommerce и OpenCart.

Таблица 3 – Таблица сравнительного анализа компаний

| Критерии | Magento | WooCommerce | Opencart |
|---|---|---|---|
| Функциональность | 5 | 3 | 4 |
| Стоимость | 2 | 5 | 3 |
| Удобство использования | 3 | 4 | 4 |
| Поддержка | 5 | 3 | 3 |
| Безопасность | 4 | 3 | 4 |

Magento демонстрирует максимальную функциональность (5), но сопровождается сравнительно высокими издержками (2) и требует более продвинутых навыков администрирования.

WooCommerce отличается минимальными затратами (5) на внедрение и вполне дружественным интерфейсом для нерегулярных пользователей (4), однако функционал (3) ограничен в сравнении с другими решениями.

OpenCart занимает серединные позиции по стоимости (3) и функционалу (4), при этом предоставляет достаточный комфорт использования (4), хотя при больших объёмах данных может возникать необходимость дополнительных оптимизаций.

Ниже представлен график сравнительного анализа компаний (рисунок 7):

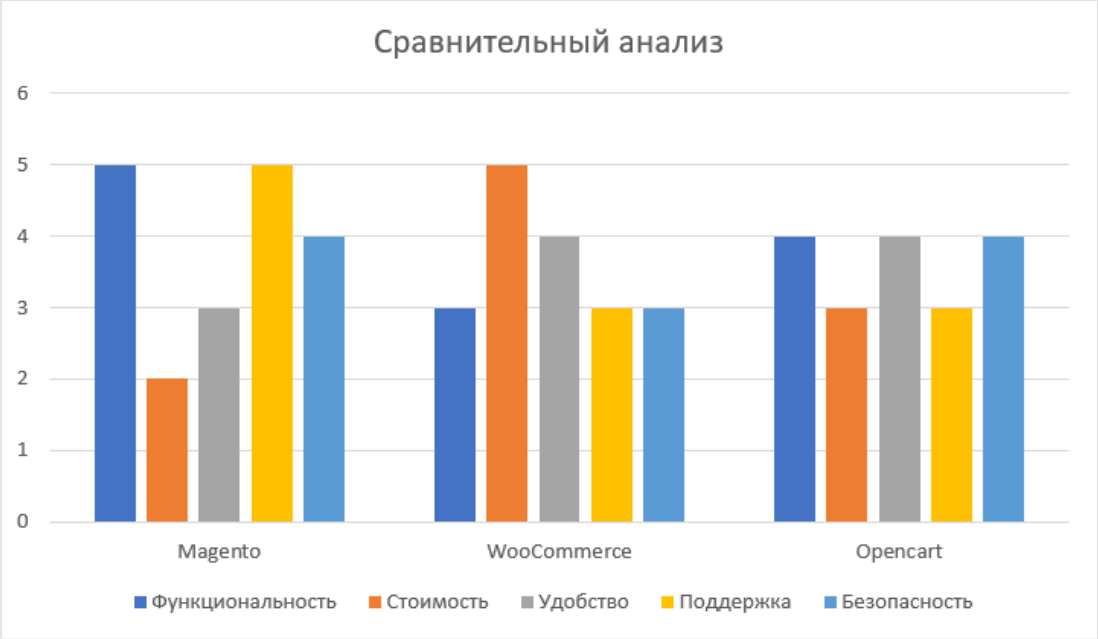


Рисунок 7 – График сравнительного анализа компаний

Анализируя данную диаграмму и сопутствующие характеристики, можно прийти к выводу, что выбор наиболее подходящего инструмента для создания интернет-магазина зависит от множества факторов.

Именно чёткое понимание бизнес-целей, ожидаемых объёмов продаж и специфики внутренней инфраструктуры позволяет определить, насколько оправданно использование каждой платформы или целесообразна разработка собственного, индивидуально настроенного решения.

Выявленные характеристики компаний после анализы проиллюстрированы на диаграмме ниже (рисунок 8).

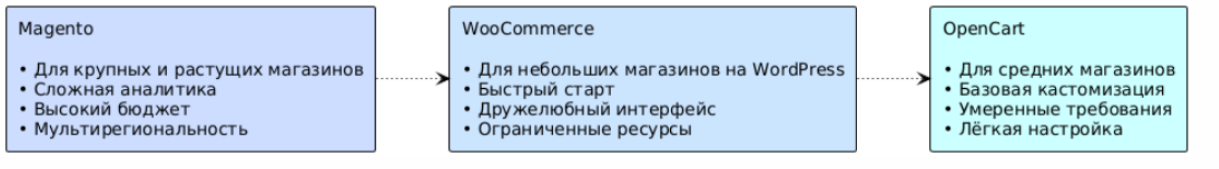


Рисунок 8 – Характеристика компаний

Таким образом, при выборе соответствующей платформы рекомендуется сопоставить собственный бюджет, объёмы будущих продаж, желаемый уровень персонализации и длительность жизненного цикла проекта. Рассмотренные решения позволяют эффективно организовать интернет-торговлю, при условии, что их функциональные особенности будут соответствовать ключевым потребностям и планируемой стратегии развития бизнеса.

Проведённый анализ показывает, что готовые платформы, такие как Magento, WooCommerce и OpenCart, имеют свои сильные стороны, но также и ряд ограничений, связанных с возможностью индивидуальной настройки, производительностью и стоимостью обслуживания. Разработка собственного веб-приложения позволяет наиболее точно учесть уникальные потребности компании «NVS-CAR», обеспечить глубокую интеграцию с существующими внутренними системами и оптимизировать ключевые бизнес-процессы.

Учитывая результаты проведенного анализа, создание собственного функционального веб-приложения для компании «NVS-CAR» является идеальным вариантом, сочетающим в себе как минимизирование расходов, так и реализацию нужного функционала.

### 1.4 Требования к функционалу и интерфейсу приложения

В этой главе подробно были подробно описаны требования к разрабатываемому функциональному веб-приложению. Требования были

созданы по международной классификации FURPS+, что дает возможность рассмотреть все возможности и характеристики реализованной системы.

Функциональными требованиями веб-приложения являются:

– поиск и фильтрация товаров: созданная система должна обеспечивать клиентам удобный и понятный интерфейс поиска товаров с реализацией функционала, позволяющего искать товары по определенным критериям;
– регистрация и аутентификация пользователей: в веб-приложении должны быть реализованы окна с регистрацией и авторизацией пользователей, механизм защиты данных, шифрование. Наглядная иллюстрация процесса регистрации и аутентификации показано на рисунке ниже (рисунок 9):
– Корзина товаров и оформление заказа: в системе должен быть реализован функционал корзины товаров и оформление заказа для удобства клиентов.
– административная панель: для эффективного управления интернет-магазином необходимо реализовать административный модуль.


Рисунок 9 – Процесс регистрации и аутентификации

Основные реализованные функции веб-приложения, которые были выявлены требованиями и необходимы для электронно-коммерческой деятельности компании описаны в таблице 4.

Таблица 4 – Описание функций веб-приложения

| Функция | Описание |
|---|---|
| Поиск и фильтрация товаров | Поиск автозапчастей по параметрам: Марка, модель, категория, ценовой диапазон. |
| Управление корзиной и заказами | Добавление товаров в корзину, изменение количества, удаление позиций и оформление заказа с выбором доставки. |
| Отзывы и поддержка | Возможность оставлять отзывы о товарах и связь с техподдержкой через чат. |
| Административная панель | Управление каталогом, обработка заказов, контроль статусов, отправка уведомлений. |

Для потенциальной поддержки взаимодействия веб-приложения с внешними сервисами используется REST API, что позволит обеспечить интеграцию с платёжными системами, логистическими сервисами и поставщиками автозапчастей. Для мгновенных уведомлений используется технология WebSocket[16]. Представление интеграции веб-приложения с внешними сервисами показано на схеме (рисунок 10).


Рисунок 10 – Интеграция веб-приложения с внешними сервисами

Система уведомлений информирует пользователя о важных событиях (новых заказах, изменении статусов, поступлении отзывов). Уведомления доставляются через WebSocket для мгновенного оповещения (рисунок 11).


Рисунок 11 – Поток уведомлений веб-приложения

Нефункциональные требования функционального веб-приложения(FURPS+) включают в себя:

– Functionality (Функциональность);
– Usability (Удобство использования);
– Reliability (Надежность;
– Performance (Производительность);
– Supportability (Поддерживаемость).

Ниже показан график изменения времени отклика в зависимости от нагрузки веб-приложения (рисунок 12).

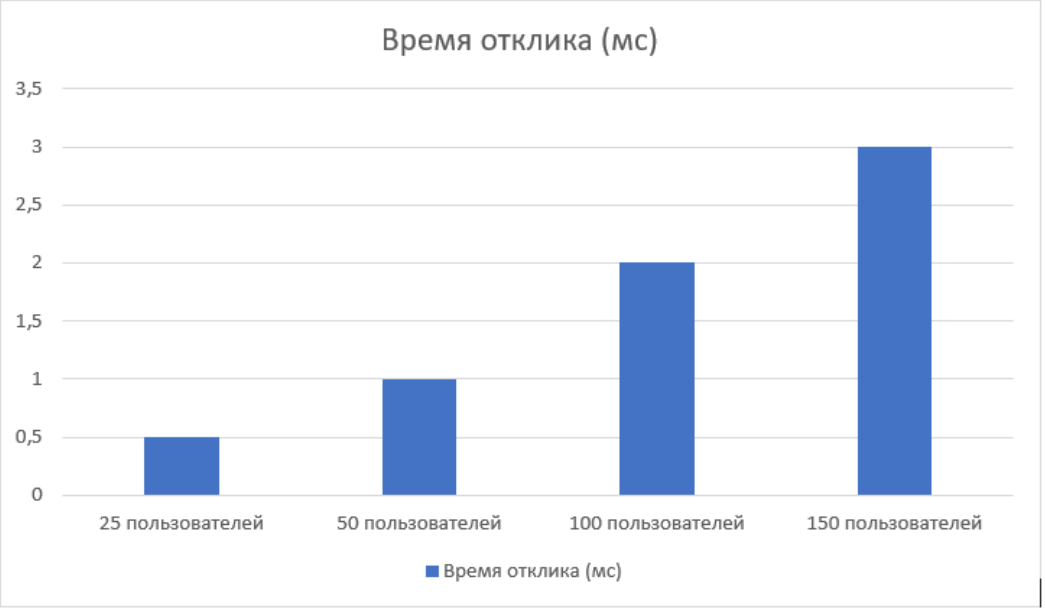


Рисунок 12 – График времени отклика в зависимости от количества пользователей

Далее были описаны дополнительные требования к проекту:

– безопасность: необходимо обеспечить надежную защиту пользовательских данных с применением актуальных методов шифрования и реализовать механизмы защиты от распространённых угроз, таких как SQL-инъекции и другие атаки. Хранение и передача конфиденциальной информации должны быть полностью защищены. Схема безопасности веб-приложения представлена ниже (рисунок 13);
– масштабируемость: архитектура приложения должна предусматривать возможность быстрого подключения новых функциональных модулей, а также обеспечивать возможность горизонтального и вертикального масштабирования, позволяя адаптироваться к изменению уровня нагрузки и динамично развивающимся требованиям рынка.

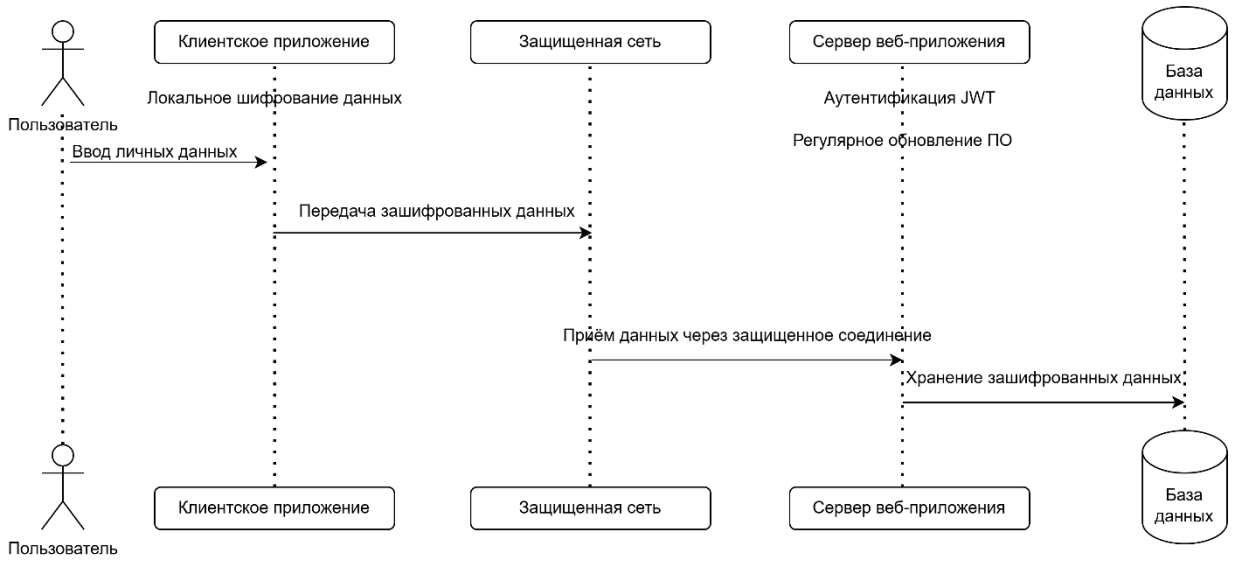


Рисунок 13 – Схема безопасности веб-приложения

Наглядный список всех нефункциональных и дополнительных требований описан в таблице 5. Нефункциональные требования определяют параметры производительности, надежности, безопасности и удобства использования системы. Также учитываются аспекты поддерживаемости, совместимости системы с разными браузерами и платформами. Дополнительные требования включают пожелания по расширяемости, интеграции с внешними сервисами и поддержке мобильных устройств. Также требования затрагивают доставку и аналитику. Это обеспечивает гибкость дальнейшей разработки, возможность быстрого внедрения нового функционала без радикальных изменений существующего кода, а также основу для дальнейшего масштабирования.

Реализация данных требований позволит компании привлечь новых клиентов удобным интерфейсом и функционалом, конкурировать на рынке электронной коммерции за счёт качественного сервиса, повысить надежность и расширяемость веб-приложения.

Таблица 5 – Список дополнительных требований с описанием

| Категория | Описание |
|---|---|
| Functionality (Функциональность) | Приложение должно реализовывать весь спектр типовых бизнес-процессов интернет-магазина автозапчастей – от первичного поиска деталей до сервисного сопровождения после завершения покупки. Все функции должны соответствовать специфике компании и ожиданиям конечных пользователей. |
| Usability (Удобство использования) | Интерфейс должен быть максимально удобным, понятным и простым как для покупателей, так и для администраторов. Реализация адаптивного дизайна позволит корректно отображать приложение на ПК, мобильных устройствах и планшетах. |
| Reliability (Надежность) | Приложение должно обеспечивать высокую устойчивость к сбоям и ошибкам. Для этого предусмотрены механизмы автоматического резервного копирования, процедуры восстановления после аварийных ситуаций и защиты данных от потерь, особенно в периоды пикового спроса. |
| Performance (Производительность) | Приложение должно иметь минимальное время отклика сервера (до 2 секунд) и оптимальную работу с базой данных. Для этого используются технологии кэширования, оптимизации запросов и возможности масштабирования инфраструктуры для обслуживания не менее 100 одновременных пользователей. |
| Supportability (Поддержка) | В приложении должны быть использованы современные технологии, правила написания кода. Также необходимо реализовать техническую документацию, позволяющую поддерживать систему, согласно всем требованиям. |
| Security (Защита) | Конфиденциальность пользователя должна быть в приоритете. Реализованы системы шифрования данных, безопасность их хранения, используются современные технологии для защиты от внешних угроз. |
| Scalability (Расширяемость) | Веб-приложение должно иметь структуру, позволяющую быстро и эффективно расширять возможности системы, интеграцию с внешними системами. |

Представленный комплекс требований станет основой для следующего этапа разработки, на котором будут детально спроектированы модели системы, разработаны технические спецификации и диаграммы, что позволит гарантировать высокое качество и конкурентоспособность готового веб-приложения.

В разработке функционального веб-приложения для компании «NVS-CAR» используется итеративный (Agile) подход, логическое моделирование

выполняется с помощью UML-диаграмм [1,7], которые созданы для визуализации бизнес-процессов, а также структур классов и взаимодействия между ними.

Ниже представлена диаграмма вариантов использования (рисунок 14).

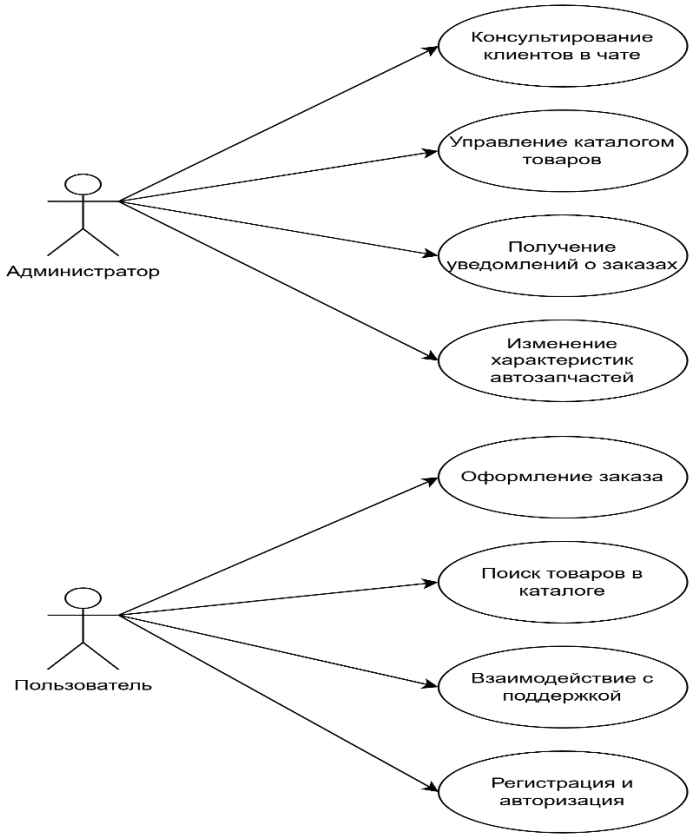


Рисунок 14 – Диаграмма вариантов использования

Представленная UML-диаграмма вариантов использования показывает взаимодействие двух основных типов пользователей. Эта диаграмма демонстрирует, каким образом система разделяет функциональность для административного управления и для взаимодействия с конечным пользователем, что является ключевым для автоматизации бизнес-процессов интернет-магазина автозапчастей.

Проведённое сформированное описание функциональных и нефункциональных требований позволяет получить чёткое представление о задачах, которые должна решать система «NVS-CAR», и критериях её

качества. Функциональные требования определяют набор пользовательских сценариев – от гибкого поиска и фильтрации товаров до полного цикла оформления заказа и работы административной панели. Нефункциональные требования (FURPS+) фокусируются на надёжности, производительности, удобстве использования и масштабируемости, что даёт основу для стабильного функционирования и дальнейшего развития приложения.

Обзор современных решений показал, что существующие платформы лишь частично удовлетворяют отраслевые требования, не всегда обеспечивая должную гибкость настройки, интеграцию с внешними сервисами и оптимизацию под специфические бизнес-сценарии. С учётом выявленных особенностей предметной области были сформулированы ключевые требования к создаваемой системе.

Таким образом, результаты первой главы подтверждают актуальность разработки современного веб-приложения для интернет-магазина автозапчастей и определяют направления дальнейшей работы, связанные с проектированием архитектуры, выбором технологических решений и формализацией требований к будущему программному продукту.

# Глава 2 Процесс разработки программного обеспечения

## 2.1 Проектирование программного обеспечения

Этап архитектурного проектирования веб-приложения «NVS-CAR» представляет собой фазу, в которой задаются масштаб и границы всей системы, а также формируется чёткая карта её внутренних и внешних взаимодействий. В ходе этой работы производится выделение ключевых компонентов, выбирается архитектурный стиль, а также определяется технологический стек. На основании анализа требований к скорости отклика и лёгкости сопровождения выбираются:

- React (ES6+) для реализации гибких, интерактивных интерфейсов с поддержкой компонентного подхода;
- Node.js[21] + Express для создания высокопроизводительного REST-API, обеспечивающего минимальные задержки при обработке запросов[17];
- WebSocket как протокол для двунаправленных каналов оповещений в реальном времени;
- PostgreSQL в роли реляционной СУБД, гарантирующей транзакционную целостность и масштабируемость при росте объёмов оперативных данных.

Создаётся визуализация физического размещения модулей на серверах и контейнерах, а также их логических связей в виде диаграммы компонентов (рисунок 15). Диаграмма развёртывания (рисунок 16) даёт представление о том, какие узлы находятся в зоне доверия, какие соединения должны быть защищены, и где целесообразно организовать балансировку нагрузки.

В совокупности эти артефакты формируют основу для последующей реализации.

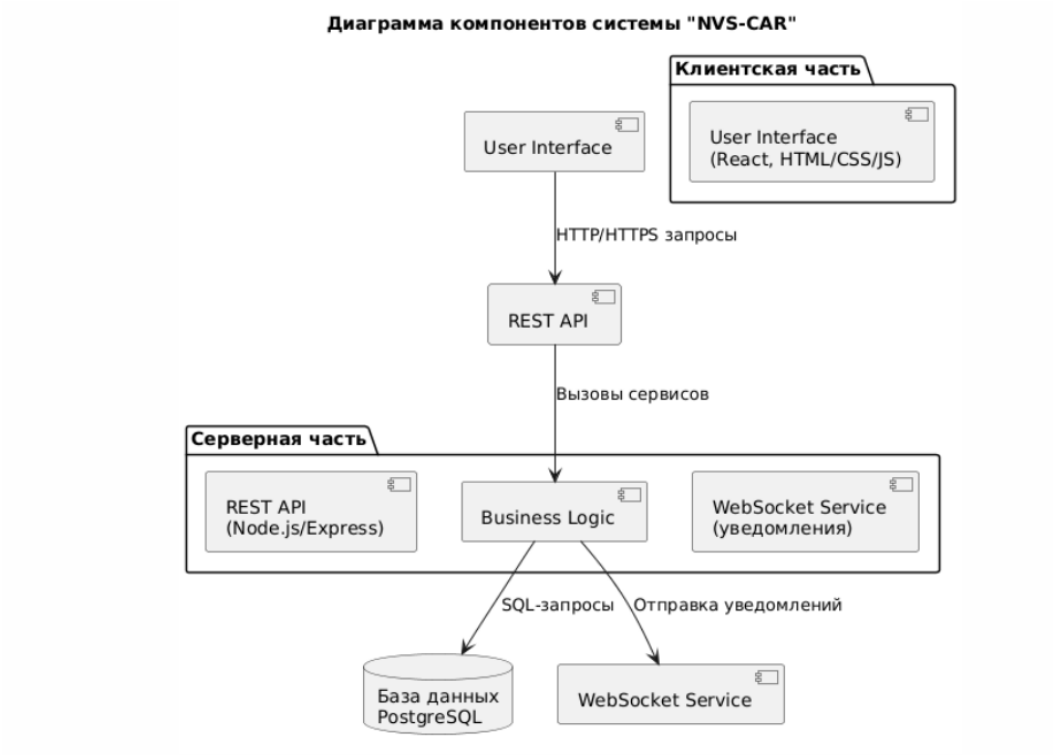


Рисунок 15 – Диаграмма компонентов веб-приложения

Также в рамках данного исследования ключевое значение имеет понимание физического развёртывания компонентов веб-приложения «NVS-CAR» и их взаимодействия на уровне сетевых соединений и сервисных вызовов. На приведённой диаграмме развертывания (рисунок 16) отображены основные узлы системы и каналы обмена данными между ними.

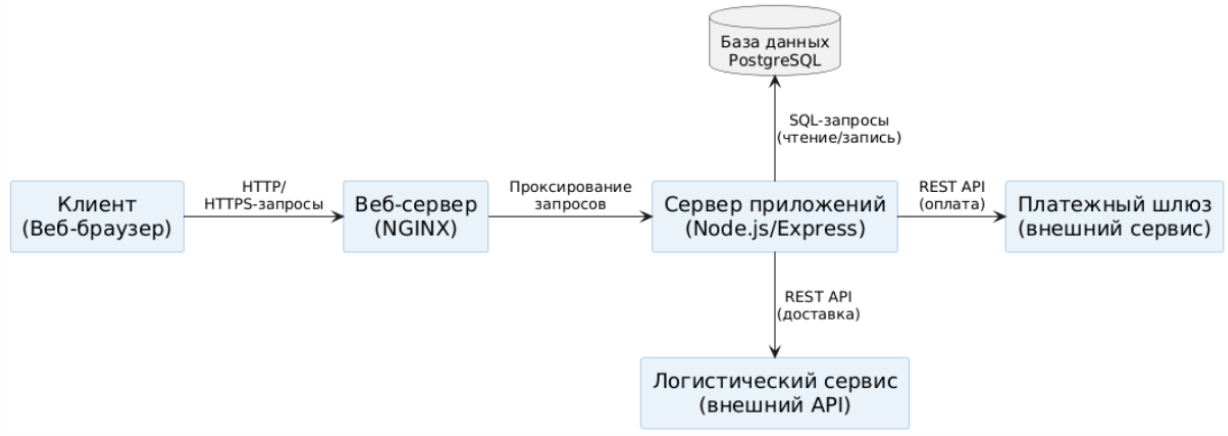


Рисунок 16 – Диаграмма развертывания.

На этапе проектирования пользовательского интерфейса выполняется разработка макетов и интерактивных прототипов, что позволяет проверить корректность структуры экранов, расположение элементов управления и логику навигации до начала программной реализации[8]. Ниже представлена диаграмма пользовательского интерфейса (UI) (рисунок 17) – схема основных экранов с ключевыми элементами управления.

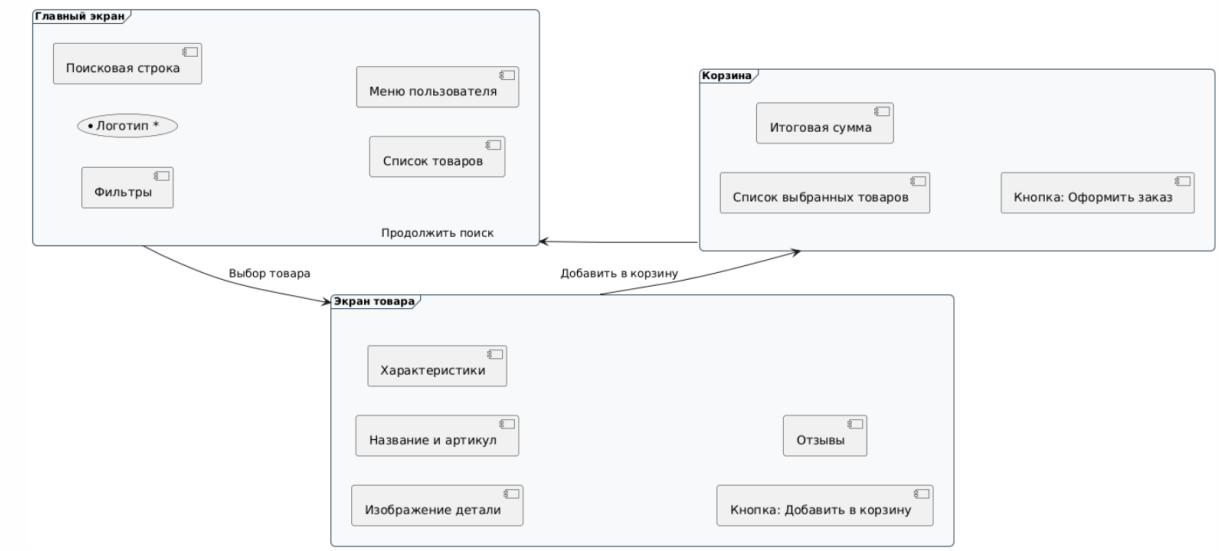


Рисунок 17 – Диаграмма пользовательского интерфейса

На этапе проектирования базы данных формируется логическая и физическая модель хранения информации, обеспечивающая целостность, непротиворечивость и эффективность операций чтения-записи.

Для наглядного представления результатов проектирования используются следующие артефакты:

– схема логической модели (рисунок 18) – отображает основные сущности (например, «Товар», «Заказ», «Пользователь», «Поставщик»), их атрибуты и характер связей (один-ко-многим, многие-ко-многим) с указанием кардинальностей;
– схема физической модели (рисунок 19) – детализация конкретных таблиц с указанием первичных и внешних ключей, индексов и

ограничений, отражающая реализацию в выбранной СУБД (PostgreSQL).

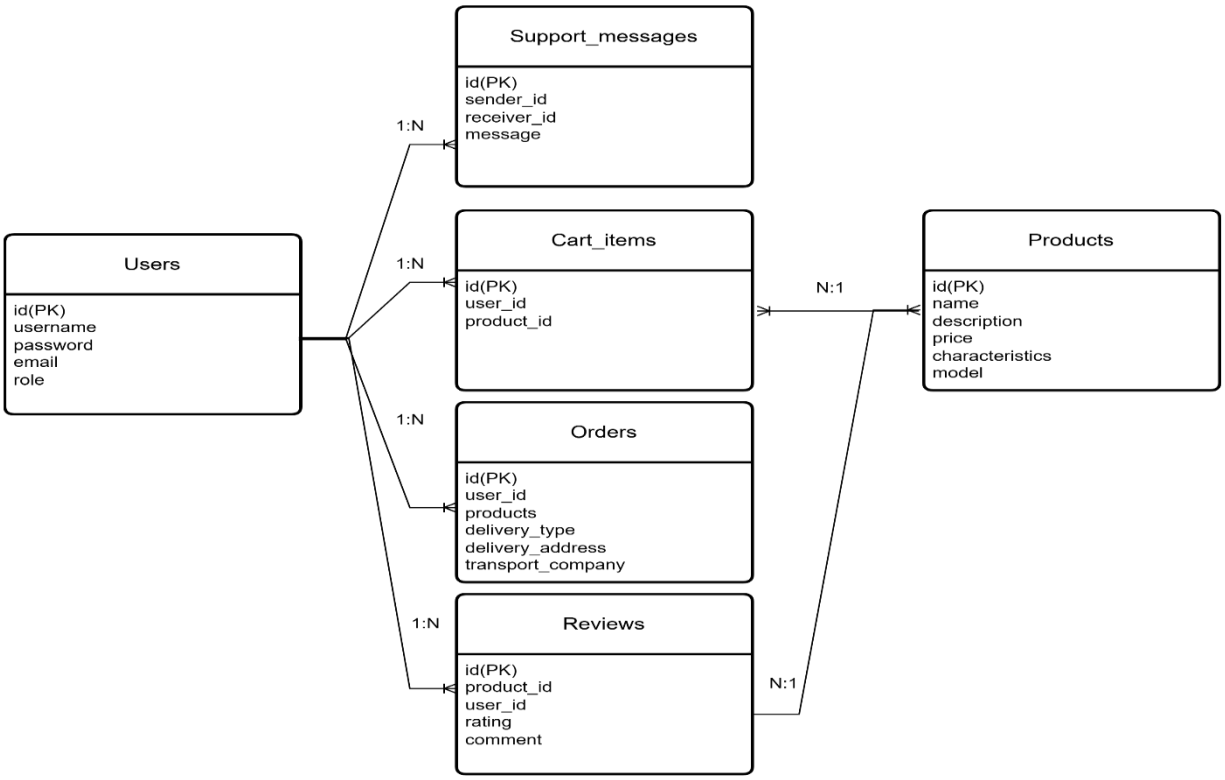


Рисунок 18 – Логическая модель базы данных

Логическая модель определяет основные сущности (таблицы) и связи между ними. Она иллюстрирует, что все данные системы централизованы вокруг пользователей, которые через внешние ключи связаны с заказами, отзывами, избранными товарами (или корзиной) и сообщениями.

Ниже приведено детальное описание физической модели базы данных проекта функционального веб-приложения для компании «NVS-CAR».

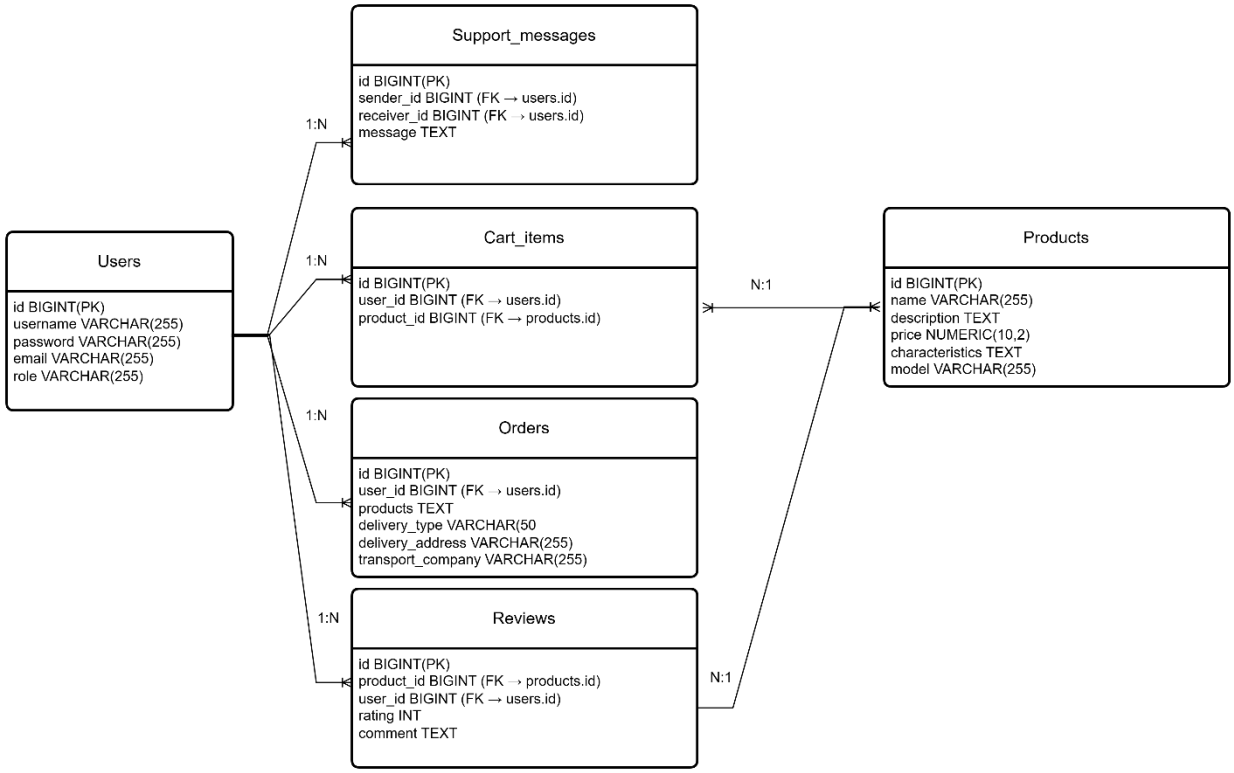


Рисунок 19 – Физическая модель базы данных

Таблица «users» является центральной в системе, так как она хранит основные данные всех пользователей, участвующих в работе интернет-магазина (как клиенты, так и администраторы). Данное поле служит связующим звеном для большинства остальных таблиц, где посредством внешнего ключа (FK) устанавливается отношение с конкретным пользователем.

Таблица «Support_messages» хранит данные личных сообщений или сообщений поддержки между пользователями. Два поля «sender_id» и «receiver_id», являющиеся внешними ключами, ссылаются на таблицу users, что позволяет точно определить отправителя и получателя каждого сообщения.

«Cart_items» отражает связь между пользователями и товарами. Если используется для корзины, здесь можно дополнительно добавить поле «quantity» для указания количества единиц товара. В данной модели каждую

запись связывается с конкретным пользователем (через «user_id») и с конкретным товаром (через «product_id»).

Таблица «orders» содержит все данные, связанные с заказами пользователей. Внешний ключ «user_id» связывает заказ с конкретным пользователем. Поле «products» позволяет хранить подробную информацию о заказанных товарах в виде структурированного текста (например, JSON), что упрощает интеграцию с другими модулями системы.

В таблице «reviews» хранятся отзывы пользователей о товарах. Внешние ключи «product_id» и «user_id» устанавливают связь отзыва с товаром и пользователем соответственно. Это отношение «многие к одному»: один товар может иметь множество отзывов, и один пользователь может оставить несколько отзывов.

Таблица «products» содержит информацию о товарах, доступных в интернет-магазине. Эти данные используются для отображения ассортимента, формирования отзывов, добавления товаров в корзину и других операций.

Связи между таблицами в физической модели связи реализуются посредством внешних ключей.

В результате проделанной работы по проектированию базы данных для проекта функционального веб-приложения для компании «NVS-CAR» были построены логическая и физическая модели базы данных, благодаря которым стали возможны:

- определение ключевых сущностей: логическая модель включает наиболее важные бизнес-сущности интернет-магазина автозапчастей, такие как пользователи, товары, заказы, отзывы, позиции корзины, обращения в службу поддержки и уведомления. Определение этих сущностей обеспечивает полное понимание необходимых данных и их взаимосвязей внутри системы;
- установление семантических связей: все сущности связаны между собой с помощью внешних ключей, что позволяет реализовать типичные отношения «один ко многим» и «многие к одному».

Подобные взаимосвязи точно отражают реальные бизнес-процессы: пользователи оформляют заказы, публикуют отзывы, добавляют товары в корзину, отправляют запросы в техническую поддержку и получают соответствующие уведомления;

Таким образом, разработанная модель базы данных является надежным фундаментом для реализации программного обеспечения управления интернет-магазином «NVS-CAR», позволяя эффективно хранить, обрабатывать и обеспечивать доступ к данным, что критично для автоматизации бизнес-процессов в современных информационных системах.

Следом идет этап проектирования бизнес-логики, в котором формализуются ключевые алгоритмы и процессы.

Чтобы проанализировать поведение пользователя были построены диаграммы активности (рисунок 20), последовательности (рисунок 21), которые представлены ниже.

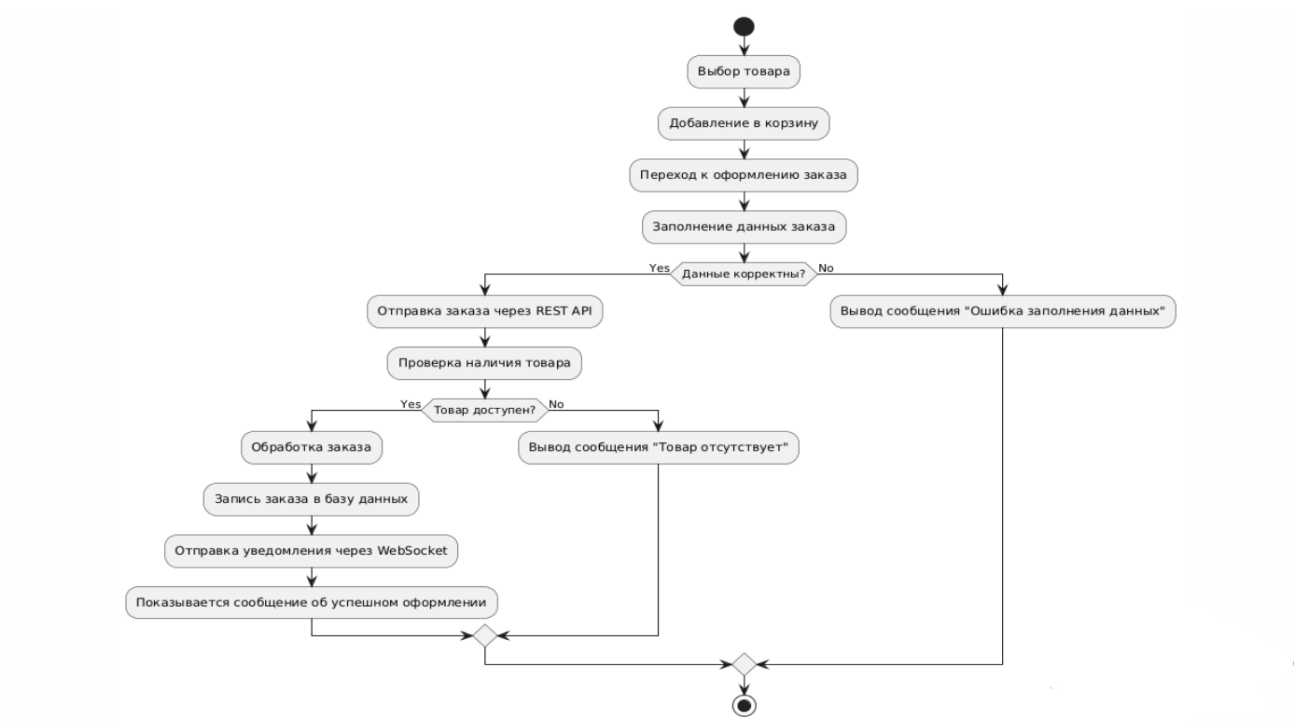


Рисунок 20 – Диаграмма активности

Диаграмма активности отражает процесс взаимодействия пользователя с системой при выполнении таких действий, как авторизация, просмотр и выбор товаров, добавление в корзину и оформление заказа. В ней представлены альтернативные пути развития событий, включая успешную и неуспешную оплату, а также уведомления администратора о новом заказе.

Диаграмма последовательности иллюстрирует стандартный путь пользователя при оформлении заказа: от авторизации и выбора товара в каталоге, через добавление в корзину и ввод данных доставки, до подтверждения и обработки оплаты.

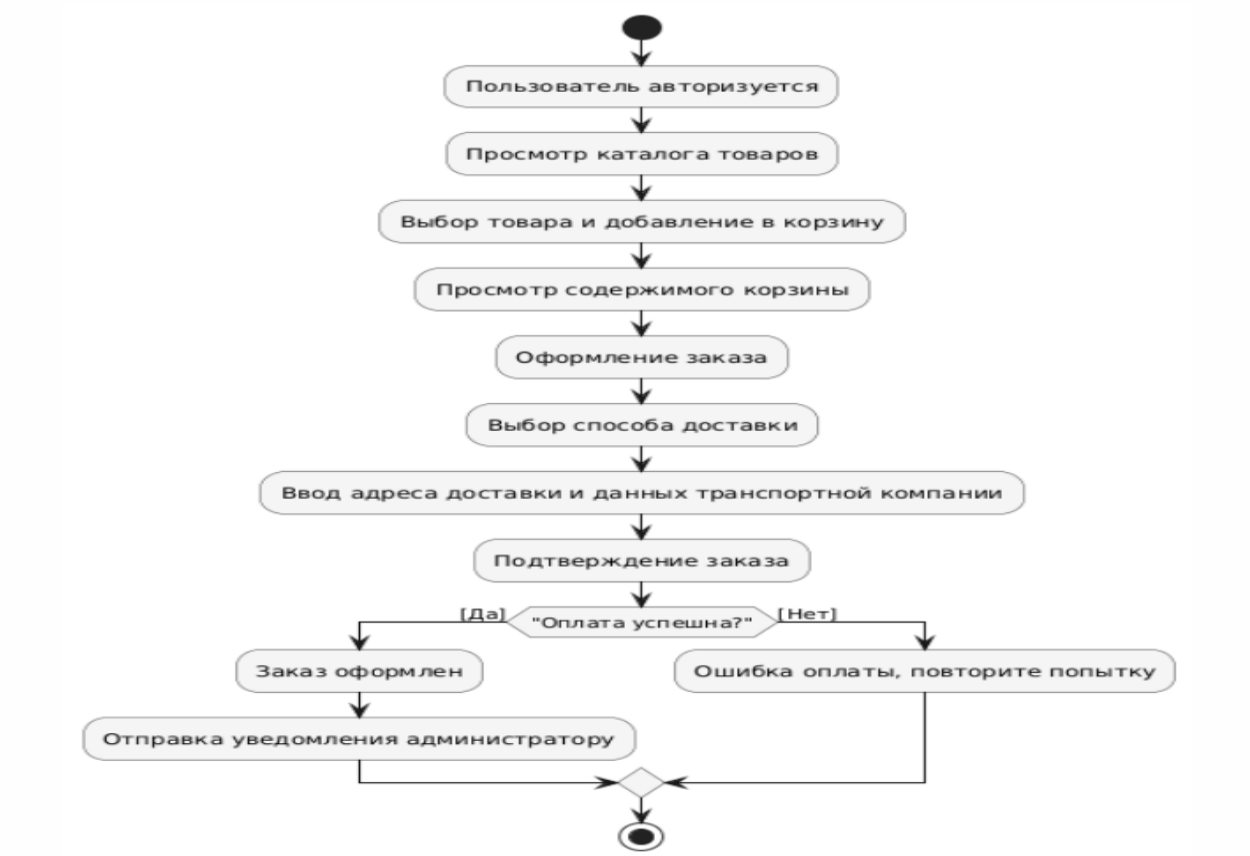


Рисунок 21 – Диаграмма последовательности

На этапе проектирования взаимодействия описываются механизмы обмена данными и протоколы взаимодействия как между внутренними

компонентами системы «NVS-CAR», так и с внешними сервисами (платёжными шлюзами, логистическими API и т. п.).

В рамках этого этапа была построена коммуникационная диаграмма (рисунок 22), в которой проиллюстрированы проверка целостности процесса и взаимодействие пользователя с системой.

Таким образом, коммуникационная диаграмма служит мостом между постановкой задачи и её технической реализацией, обеспечивая прозрачность, полноту и однозначность описания сценария оформления заказа в веб-приложении.

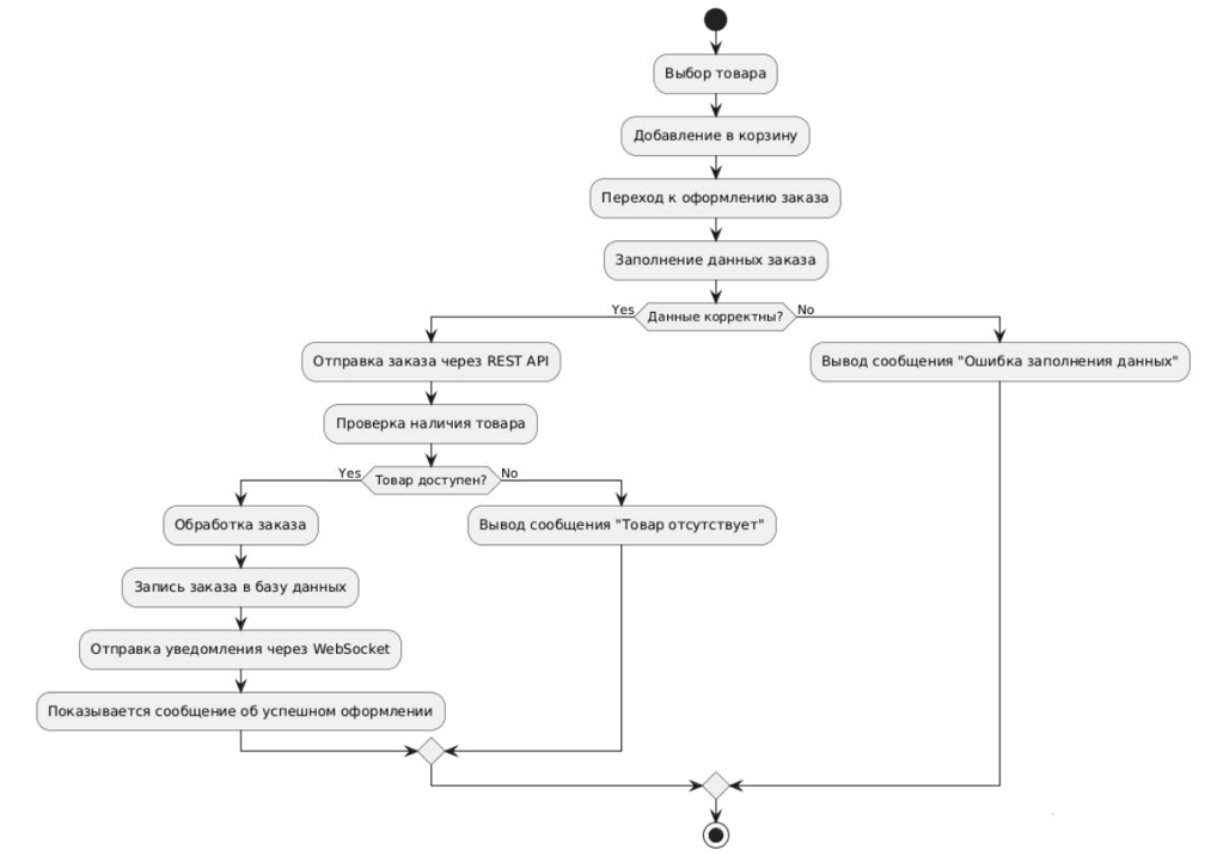


Рисунок 22 – Коммуникационная диаграмма

Также с помощью построенной диаграммы классов (рисунок 23), которая показана ниже мы можем увидеть визуализацию классов и их отношения в системе, включая связи, атрибуты и операции.

Предоставленная UML-диаграмма классов отражает подробную структуру программного обеспечения проекта функционального веб-приложения для компании «NVS-CAR».

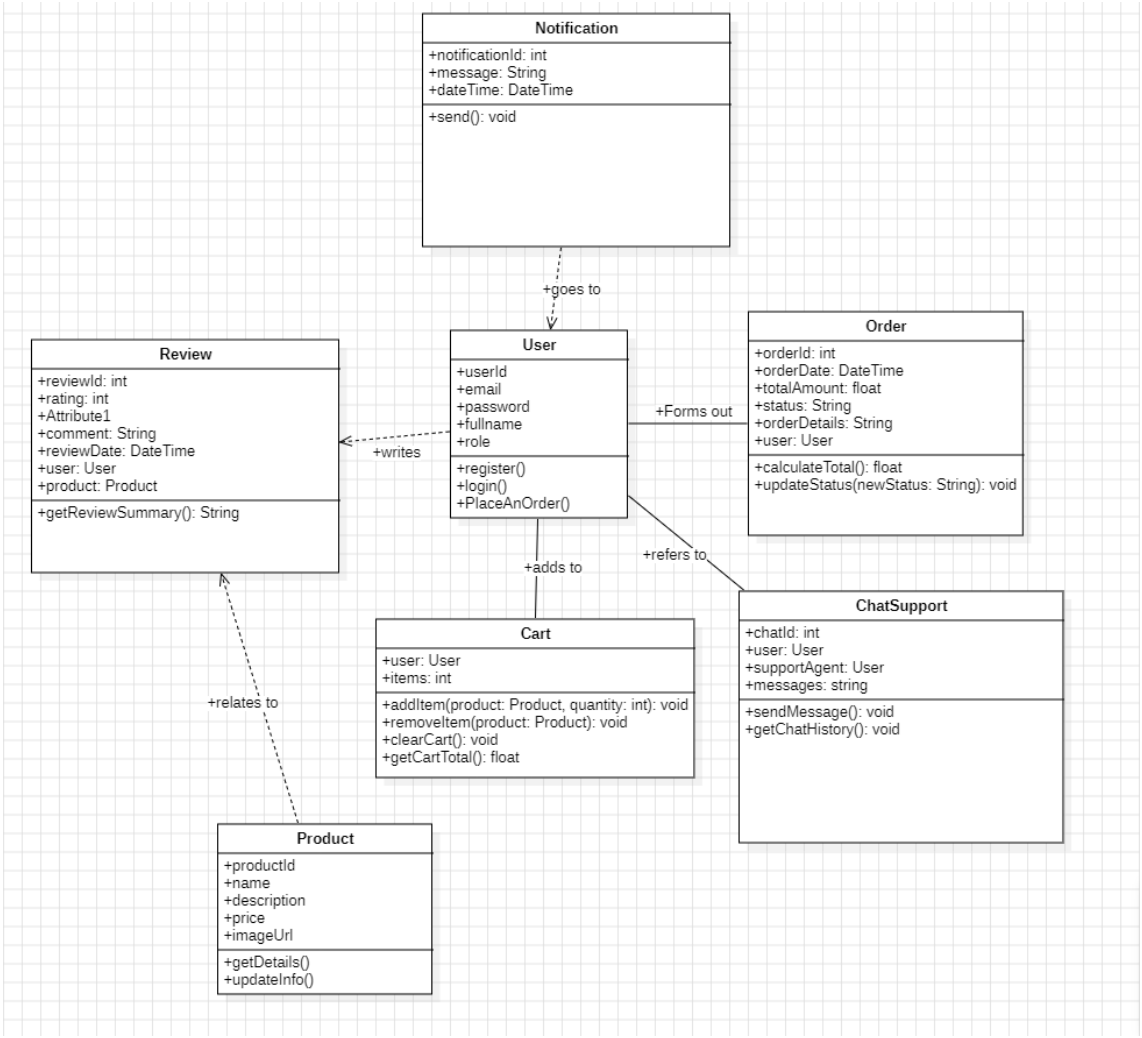


Рисунок 23 – UML-диаграмма классов.

Этап проектирования веб-приложения «NVS-CAR» позволил выработать целостную модель системы, включающую выбор технологического стека, описание пользовательского интерфейса, структуры данных, бизнес-логики и механизмов взаимодействия компонентов.

В совокупности эти проектные артефакты образуют единую логико-структурную основу, необходимую для последовательной реализации, тестирования и сопровождения приложения «NVS-CAR». Это позволяет в

дальнейшем перейти к реализации с минимальными рисками, точно следуя утверждённым спецификациям.

## 2.2 Выбор технологий и инструментов для разработки

При создании веб-приложения интернет-магазина автозапчастей «NVS-CAR» выбран актуальный и эффективный стек технологий, который гарантирует высокую производительность, удобство разработки и масштабируемость. При выборе технологий были учтены такие критерии, как надёжность, активность сообщества разработчиков и простота интеграции с внешними сервисами.

Для создания веб-приложения был выбран язык программирования JavaScript (стандарт ES6+). JavaScript (ES6+) выбран благодаря своей универсальности [2], [6], [13]: он исполняется в браузере и на сервере (Node.js), что упрощает обмен логикой между клиентом и бэкендом. Асинхронная модель с promises и async/await обеспечивает высокую пропускную способность при одновременных запросах к платёжным и логистическим API.

Также был использован фреймворк React. Использование React оправдано высокой скоростью работы, развитым сообществом разработчиков и обширной библиотекой готовых компонентов, что позволяет оперативно создавать интерактивные и масштабируемые пользовательские интерфейсы.

В создании веб-приложения использовались различные инструменты:

- HTML5 и CSS3 – базовые инструменты для разработки и стилизации интерфейсов;
- Redux или Context API – для эффективного управления состоянием приложения
- Webpack, Babel – обеспечивают сборку и преобразование кода, поддерживая современные возможности JavaScript

Использование библиотек компонентов пользовательского интерфейса (например, Material-UI или Bootstrap) ускоряет разработку и обеспечивает адаптивный дизайн[6].

В качестве фреймворка для серверной части приложения был выбран Express, который обусловлен лёгкостью и высокой гибкостью, что позволяет быстро реализовать REST API и обеспечить прозрачную связь между клиентом и сервером. Это особенно подходит для проектов, требующих масштабируемости и простоты расширения.

Базой данных проекта выступала СУБД: PostgreSQL. PostgreSQL[22] выбран в связи с высоким уровнем надёжности, производительностью и широким функционалом для работы со сложными структурированными данными [8], [9]. База данных имеет таблицы, а также внешние ключи для обеспечения целостности и консистентности данных[10], [17].

Наглядное представление взаимодействия приложения с базой данных показана на схеме (рисунок 24).

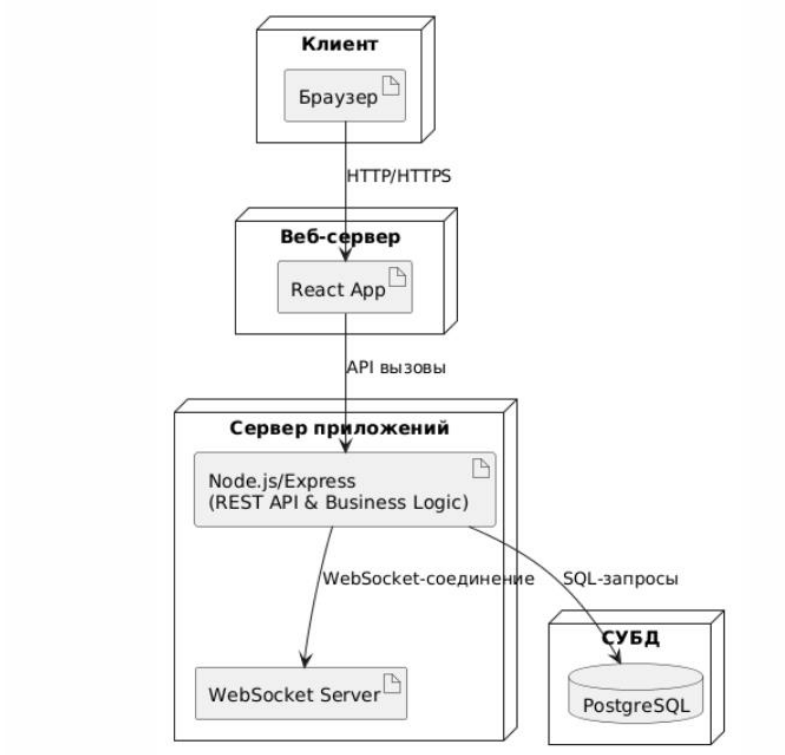


Рисунок 24 – Взаимодействие приложения с базой данных

Для проведения анализа была построена сводная таблица 6, которая отображает оценки технологий. Оценка технологий по ключевым критериям (производительность, поддержка, стоимость).

Таблица 6 – Сводная таблица оценки технологий

| Технология/ Инструмент | Производительность | Поддержка | Стоимость | Примечания |
|---|---|---|---|---|
| React | 4 | 5 | 5 | Высокая масштабируемость |
| Node.js/Express | 5 | 5 | 5 | Отличная производительность и поддержка |
| PostgreSQL | 5 | 4 | 5 | Надёжная СУБД, отлично работает с данными |
| Visual Studio Code | 5 | 5 | 5 | Бесплатный, гибкий редактор кода |

Выбор технологического стека основан на следующих аспектах:

- производительность и масштабируемость;
- гибкость разработки;
- поддержка сообщества и документация;
- современные подходы.

В результате анализа требований и сравнительной оценки технологий был сформирован сбалансированный стек, обеспечивающий «NVS-CAR» необходимые характеристики надёжности, быстродействия и масштабируемости. Выбор этих решений основан на их зрелости, активном сообществе и простоте интеграции с внешними сервисами, что создаёт прочную техническую базу для масштабируемого и устойчивого развития проекта.

## 2.3 Реализация функциональных требований

На этапе реализации функциональных требований выполняется непосредственное воплощение спроектированных алгоритмов и интерфейсов

в коде приложения «NVS-CAR». В рамках данной стадии последовательно решаются следующие задачи:

Разработанное программное обеспечение «NVS-CAR» построено на основе трехуровневой клиент-серверной архитектуры [3]. Такая структура[7] обеспечивает четкое разделение ответственности, упрощает поддержку и масштабирование системы в будущем[4].

Ключевые решения, принятые на этапе проектирования, были последовательно воплощены в программном коде как серверной, так и клиентской части приложения «NVS-CAR». Серверная часть реализована на языке JavaScript с использованием платформы Node.js[5] и фреймворка Express, что обеспечивает эффективную обработку запросов, управление бизнес-логикой, доступ к базе данных и интеграцию с внешними сервисами. Основные модули бэкенда отвечают за обработку пользовательской аутентификации, работу с заказами, каталогом товаров, а также реализацию системы уведомлений и поддержки[15].

Клиентское приложение построено на базе фреймворка React и реализует все основные пользовательские сценарии: навигацию по каталогу, добавление товаров в корзину, оформление заказов, просмотр истории покупок и взаимодействие с интерфейсом поддержки. Использование современных библиотек управления состоянием (Redux или Context API) позволило обеспечить высокую отзывчивость и надёжность интерфейса на всех этапах взаимодействия пользователя с системой[13].

Примеры программного кода программы приведены в приложении к данной работе (см. Приложение 1).

Для подтверждения реализованных функциональных возможностей и демонстрации интерфейса разработанного веб-приложения «NVS-CAR» приведены основные иллюстрации, отражающие работу ключевых разделов системы. Скриншоты наглядно отображают структуру и оформление пользовательских страниц, а также реализацию автоматизации бизнес-процессов на практике.

На рисунках представлены ключевые пользовательские интерфейсы разработанного веб-приложения «NVS-CAR», реализующего функциональные требования системы.

На главной странице (рисунок 25) отображается приветственный блок и основные навигационные элементы, позволяющие быстро перейти к просмотру каталога товаров, оформлению заказа или получению информации о компании. Интерфейс выполнен в едином фирменном стиле, что способствует удобству восприятия и облегчает ориентацию пользователя при первом посещении сайта.

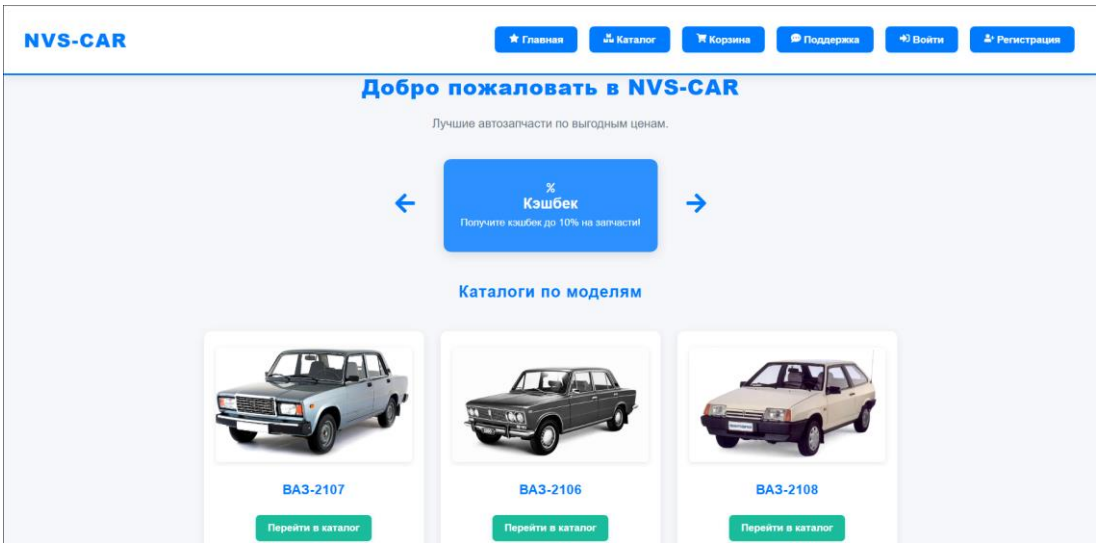


Рисунок 25 – Главная страница веб-приложения

Каталог автозапчастей (рисунок 26) реализован в виде структурированной иерархии товаров с возможностью фильтрации и поиска по ключевым характеристикам: наименование, артикул, категория, применимость к определённым моделям автомобилей. Для каждой позиции предусмотрена отдельная карточка, содержащая описание, фотографию, цену, а также кнопку добавления товара в корзину. Реализация динамической загрузки данных и поддержка мгновенного обновления ассортимента обеспечивают быструю реакцию интерфейса на действия пользователя.

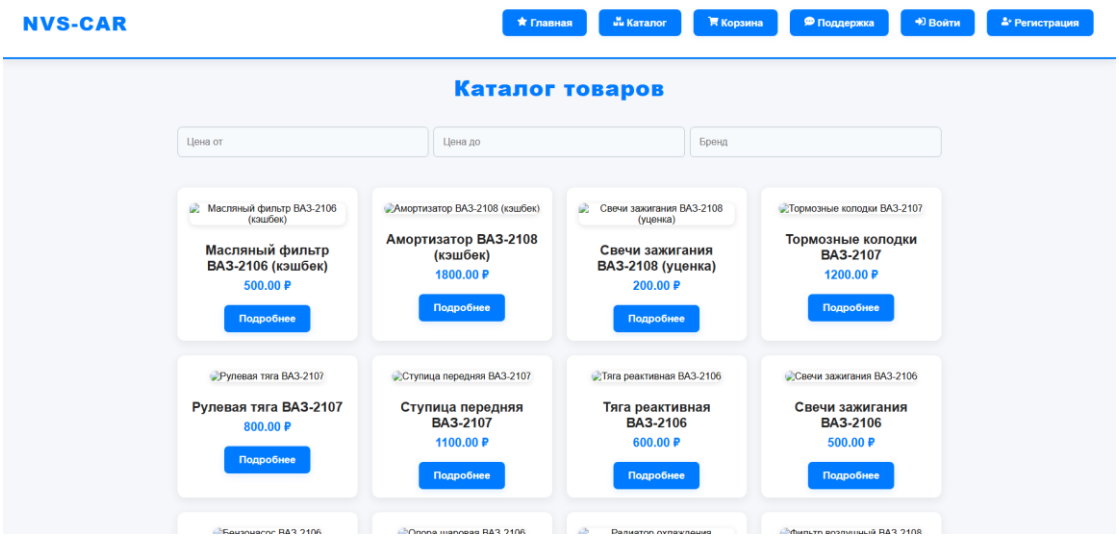


Рисунок 26 – Каталог товаров

На странице остатков товаров на складе (рисунок 27) представлен список доступных позиций с указанием текущего количества на складе. Данный интерфейс предназначен для администратора и позволяет оперативно контролировать наличие товаров, отслеживать динамику запасов и своевременно реагировать на снижение остатков.

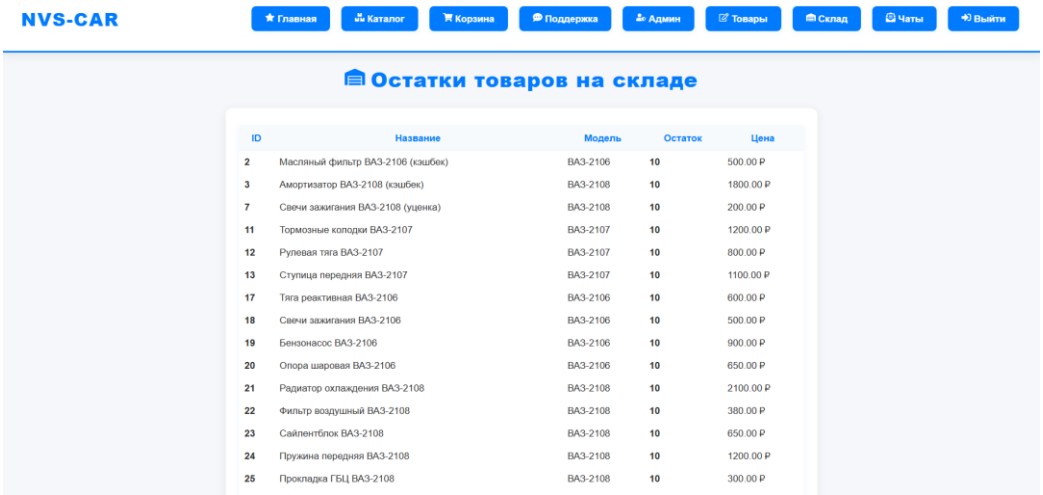


Рисунок 27 – Остатки товаров на складе

В совокупности данные интерфейсы иллюстрируют корректную реализацию основных пользовательских сценариев и подтверждают соответствие приложения заявленным функциональным требованиям.

В результате второго этапа разработки для интернет-магазина автозапчастей «NVS-CAR» сформирована комплексная архитектура программного обеспечения, включающая чёткое разделение клиентской, серверной и базы данных. Обоснован выбор технологического стека, в состав которого вошли React, Node.js/Express и PostgreSQL, что обеспечило гибкость, производительность и масштабируемость системы.

Разработаны макеты интерфейса, а также логическая и физическая модели базы данных, что позволило заранее предусмотреть все ключевые сценарии работы пользователей и администраторов. На этапе реализации воплощены в коде основные бизнес-процессы: обработка заказов, управление каталогом, поддержка пользователей и администрирование склада.

Такой подход позволил не только повысить устойчивость системы к возможным ошибкам, но и упростить последующую доработку при необходимости внедрения новых функций.  Также была проработана система защиты, дана возможность дальнейшего расширения возможностей системы и масштабируемость.

# Глава 3 Оценка эффективности разработанного программного обеспечения

## 3.1 Тестирование и отладка приложения

Оценка эффективности ПО и его тестирования является важнейшей частью разработки. В ходе тестирования использовались как автоматизированные средства, так и ручной анализ, что обеспечило всестороннюю проверку функциональности, производительности и безопасности приложения. Все этапы тестирования показаны в таблице 7.

Таблица 7 – Тестирование веб-приложения

| Этап тестирования | Цель | Метод |
|---|---|---|
| Функциональное тестирование | Проверка корректности бизнес-функций | Ручное тестирование, Postman, ESLint |
| Юнит-тестирование | Проверка отдельных модулей | Автоматизированные тесты (Jest, Mocha) |
| Интеграционное тестирование | Проверка взаимодействия между компонентами | Автоматизированные интеграционные тесты |
| Регрессионное тестирование | Проверка сохранения работоспособности | Автоматизированные тесты (Selenium, Jenkins) |
| Нагрузочное тестирование | Оценка производительности под нагрузкой | Apache JMeter |

В рамках проекта «NVS-CAR» особое внимание было уделено качеству кода. Для контроля стиля, выявления синтаксических и логических ошибок, а также обеспечения единообразия кода был использован ESLint – современный статический анализатор JavaScript-кода.

Список функций приложения с указанием критериев успешности показаны в таблице 8.

Таблица 8 – Список тестируемых функций

| Функция | Описание | Критерии успешности |
|---|---|---|
| Регистрация и авторизация | Создание аккаунта и вход в систему | Успешная регистрация, выдача JWT-токена, отсутствие ошибок |
| Поиск и фильтрация товаров | Поиск автозапчастей по различным параметрам | Корректный вывод результатов, время отклика <2 сек |
| Управление корзиной | Добавление, изменение и удаление товаров | Своевременное обновление данных, отсутствие ошибок |
| Оформление заказа | Создание заказа с проверкой наличия товаров | Заказ успешно создан, уведомление отправлено |
| Отзывы и поддержка | Оставление отзывов, работа чата поддержки | Отзывы сохраняются, чат работает корректно |

Для функционального тестирования веб-приложения был установлен и настроен ESLint[19]. В корневой директории проекта была выполнена установка ESLint как dev-зависимости.

Вывод консоли после запуска ESLint в серверной части показан на рисунке (рисунок 28).

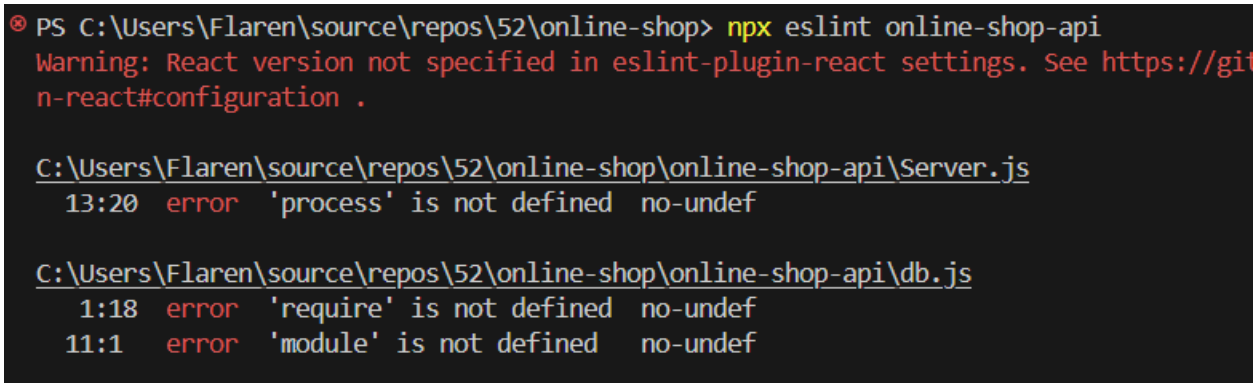


Рисунок 28 – Выявленные ошибки в ходе тестирования серверной части

После того как была запущена команда npx eslint online-shop-api, ESLint выявил следующие ошибки:

- в файле Server.js: ошибка «'process' is not defined» (строка 13:20);

- в файле db.js: ошибки «'require' is not defined» (строка 1:18) и «'module' is not defined» (строка 11:1).

Вывод консоли после запуска ESLint в клиентской части показан на рисунке (рисунок 29).

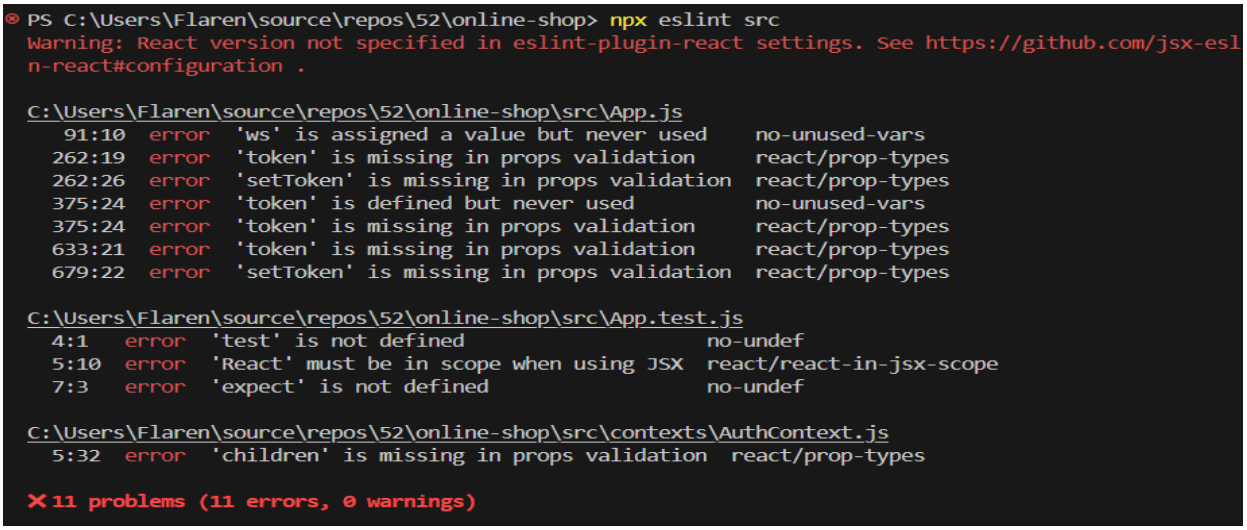


Рисунок 29 – Выявленные ошибки в клиентской части.

После запуска проверки с помощью команды npx eslint src были обнаружены следующие ошибки:

- предупреждение о том, что версия React не указана в настройках ESLint;
- ошибка «ws is assigned a value but never used» в файле App.js, указывающая на неиспользуемую переменную;
- несколько предупреждений о том, что пропсы token и setToken не описаны через PropTypes;
- в тестовом файле App.test.js ESLint ругался на отсутствие определения глобальных переменных Jest (test, expect) и на требование, чтобы React был в зоне видимости при использовании JSX;
- в файле contexts/AuthContext.js обнаружено отсутствие валидации для пропса children.

Далее были проведены анализ и исправление ошибок. Исправления включали в себя:

- проблема с глобальными переменными Node.js: Для устранения ошибок «'process' is not defined», «'require' is not defined» и «'module' is not defined» была добавлена в начало каждого серверного файла (например, Server.js и db.js) директиву/* eslint-env node */, что позволило ESLint распознавать глобальные переменные, присущие Node.js;
- альтернативный вариант: Также были настроены override в файле eslint.config.mjs, чтобы для файлов из директории online-shop-api правило no-undef было отключено.
- указание версии React: в настройках ESLint: это устранило предупреждение о неуказанной версии React;
- неиспользуемая переменная: переменная ws была выявлена как неиспользуемая, после чего удалена из кода программы.

Данное тестирование помогло выявить множество ошибок, которые были оперативно исправлены благодаря следующим мерам:

- добавлению директив /* eslint-env node */ для серверных файлов;
- настройке параметров для работы с React (указание версии, отключение неактуальных правил);
- добавлению комментариев для тестовых файлов и настройки overrides;
- исключению из проверки сгенерированных директорий через .eslintignore, удалось устранить большинство ошибок, обнаруженных ESLint. Это позволило не только повысить качество и читаемость кода, но и обеспечить его соответствие современным стандартам разработки.

Проведённое тестирование показало, что все ключевые функции «NVS-CAR» работают в соответствии с заданными критериями.

Следом было проведено нагрузочное тестирование, целью которого было проверить производительность и стабильность веб-приложения «NVS-CAR» при одновременной работе большого количества пользователей. Особое внимание уделялось следующим функциональным сценариям:

- регистрация и авторизация пользователей;
- просмотр каталога товаров;
- добавление товара в корзину;
- оформление заказа.

Для проведения тестирования был использован Apache JMeter[11]. В тестовом плане были настроены различные элементы программы для корректного тестирования.

Выводы тестирования нагрузочного тестирования показали такие результаты:

- регистрация и авторизация: тесты показали, что при регистрации используется проверка уникальности email. При повторной регистрации с тем же email сервер корректно возвращал ошибку «Пользователь уже зарегистрирован». Для авторизации использовались корректные данные (username и пароль), но возникали проблемы, если данные не соответствовали ранее зарегистрированным;
- нагрузочное тестирование: при моделировании нагрузки до 50–100 одновременных пользователей время отклика оставалось стабильным, а процент ошибок был минимальным.

Иллюстрации проведенного тестирования показаны ниже на рисунках (рисунки 30-34)

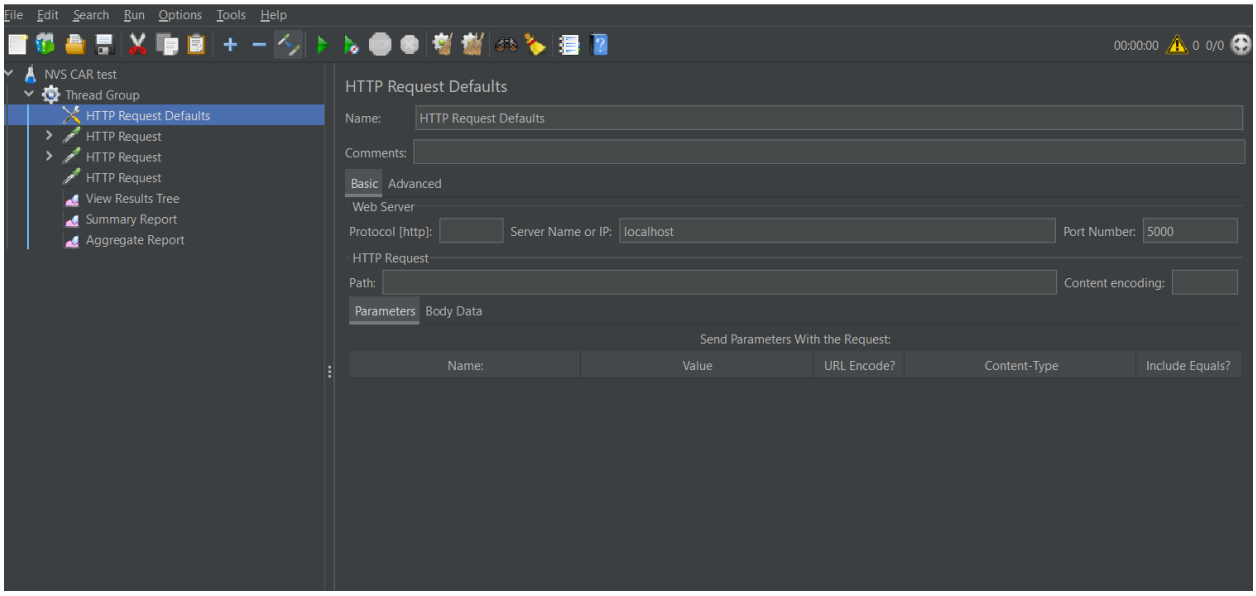


Рисунок 30 – Настройка HTTP Request Defaults

На данном рисунке отображена конфигурация элемента HTTP Request Defaults в JMeter. Здесь задаются общие параметры, которые будут применяться ко всем HTTP-запросам в тестовом плане.

Эти настройки позволяют не указывать повторяющиеся параметры в каждом отдельном HTTP Request, что упрощает настройку тестового плана и обеспечивает единообразие параметров.

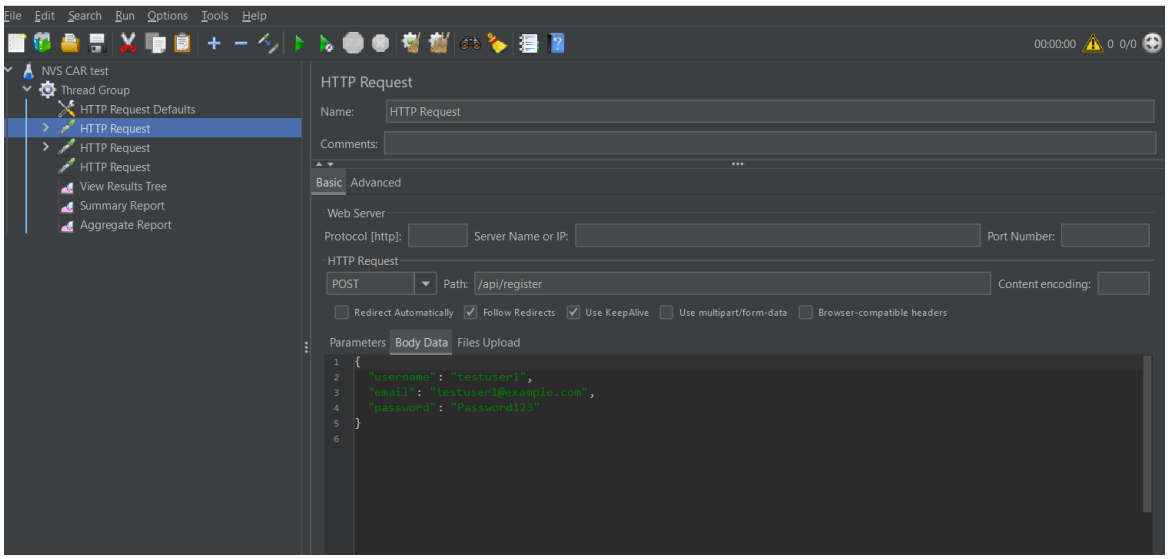


Рисунок 31 – Настройка запросов на регистрацию

На рисунке представлен файл HTTP Request, который был специально настроен под тестирование регистрации пользователя, введены данные, нужный путь к конечной точке регистрации. Благодаря такой настройке, все данные отправятся для проверки регистрации пользователя.

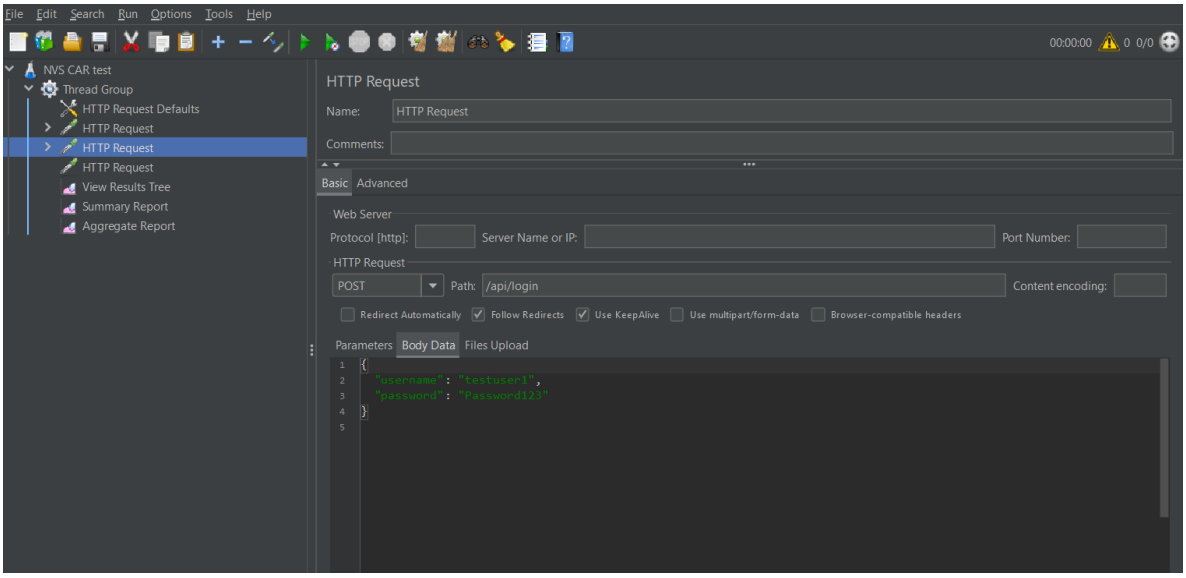


Рисунок 32 – Настройка запросов на авторизацию

Настройки авторизации также были реализованы под корректную доставку данных, использующихся при авторизации пользователя.

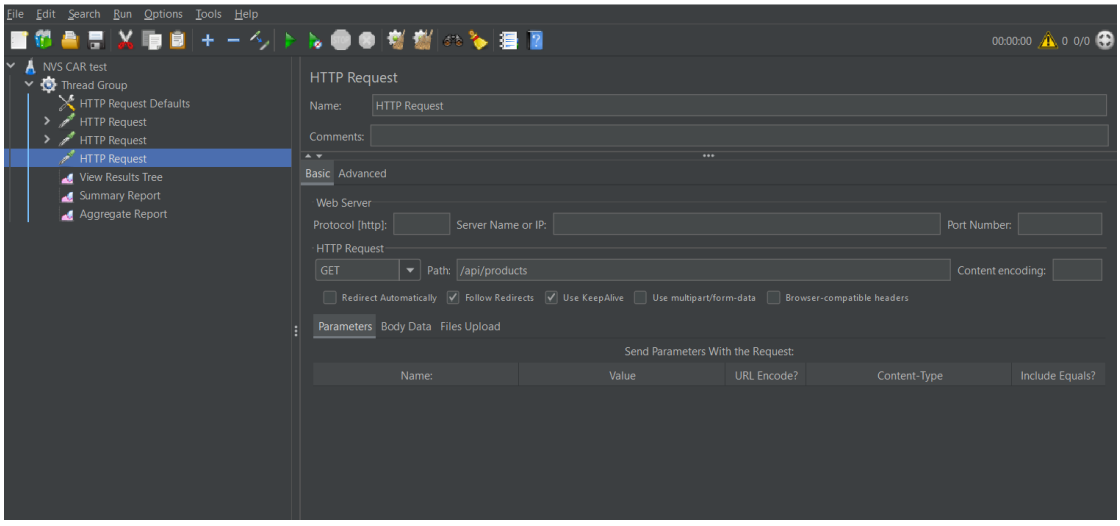


Рисунок 33 – Настройка запросов на просмотр каталога

На данном рисунке показан HTTP Request для получения списка товаров. Эта настройка позволяет получить актуальный каталог товаров, подтверждая, что сервер корректно обрабатывает GET-запросы и возвращает данные.

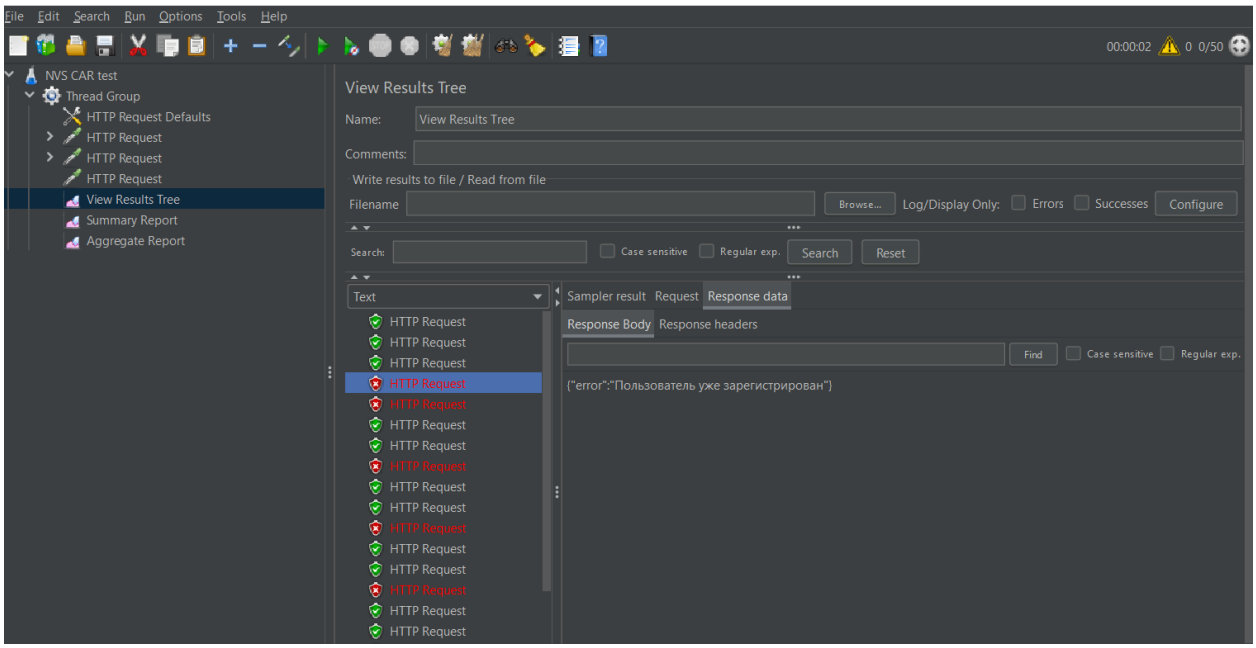


Рисунок 34 – Вывод результатов тестирования в View Results Tree

На этом скриншоте показан вывод результатов тестирования в Listener'e View Results Tree. Здесь видны индивидуальные результаты каждого HTTP-запроса, выполненного в рамках тестового плана. Ошибки, выявленные на рисунке показывают, что регистрация работает успешно и не регистрирует заново пользователя, а сообщает, что он уже зарегистрирован

Таким образом, анализ результатов, полученных в окне View Results Tree, подтверждает корректность реализации тестового плана: функциональные запросы успешно обрабатываются системой, а попытка повторной регистрации корректно вызывает ошибку. Это подтверждает правильность реализации механизма проверки уникальности регистрируемых пользователей. Наглядное представление нагрузочного тестирования проиллюстрировано на рисунке (рисунок 35).


Рисунок 35 – Схема нагрузочного тестирования

Следом за нагрузочным тестирование шли регрессионное и юнит-тестирование. Модульное тестирование (unit-tests) и проверка ключевых блоков регистрации и входа в систему:

- создание нового пользователя (POST /api/register → статус 201);
- подтверждение шифрования пароля;
- успешная и неуспешная авторизация (POST /api/login → статусы 200/401).

Ниже представлена схема unit-тестирования регистрации и входа (рисунок 36).

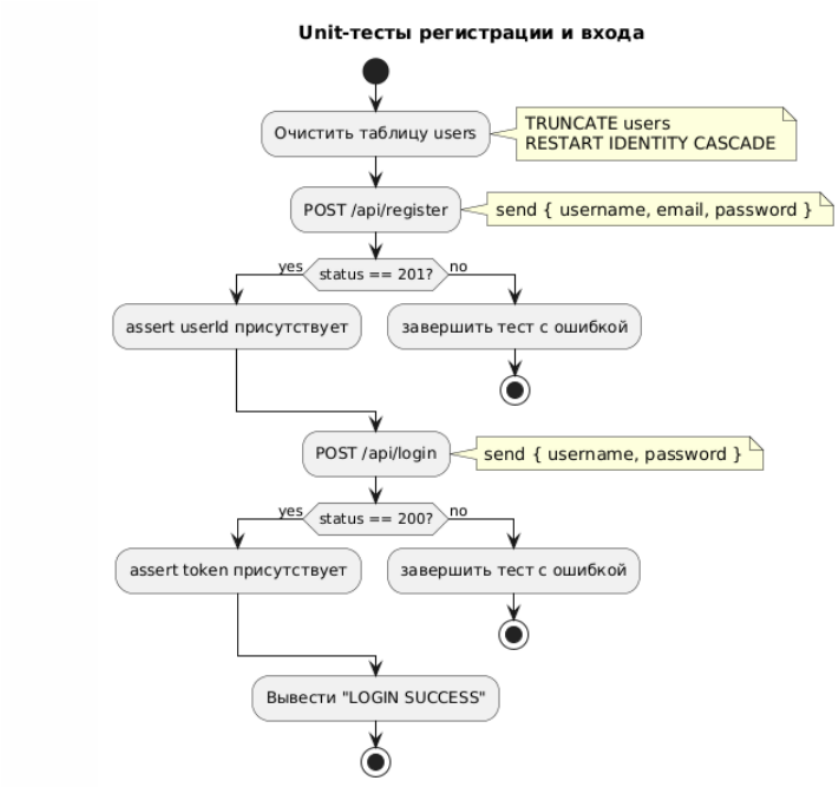


Рисунок 36 – Unit-тестирование

В интеграционно-сквозном тестировании (Регрессионное тестирование) была проведена симуляция полного пользовательского сценария от роли администратора до оформления покупки:

- регистрация и вход администратора;
- публикация товара администраторами (POST /api/products → статус 201);
- регистрация и вход обычного юзера;
- добавление товара в корзину (POST /api/cart → статус 200);
- оформление заказа (POST /api/orders → статус 201).

Ниже представлена схема регрессионного тестирования (рисунок 37).


Рисунок 37 – Схема регрессионного тестирования

В результате получено:

- Unit-тесты прошли без сбоев, проверены основные операции регистрации и авторизации;
- E2E-тест отработал успешно: товар создаётся, добавляется в корзину и покупается, все HTTP-статусы совпали с ожидаемыми.

В итоге тестирования проведён комплекс тестов, охватывающий как отдельные модули, так и полный пользовательский путь. Использованный стек инструментов (supertest + assert + Node.js) оказался лёгким для внедрения и масштабирования при добавлении новых сценариев.

Далее было проведено интеграционное тестирование, в современных веб-системах надёжность серверной логики определяется не только корректностью отдельных модулей, но и их взаимодействием. Интеграционное тестирование позволяет проверить целостность бизнес-процессов на примере реальных сценариев: регистрации, авторизации, управления каталогом и оформления заказов. Для тестирования были использованы Jest и Supertest.

Схема интеграционного тестирования (рисунок 38) иллюстрирует весь процесс. Визуально показана последовательность этапов – от постановки цели, выбора инструментов до настройки среды и непосредственного выполнения тестов.

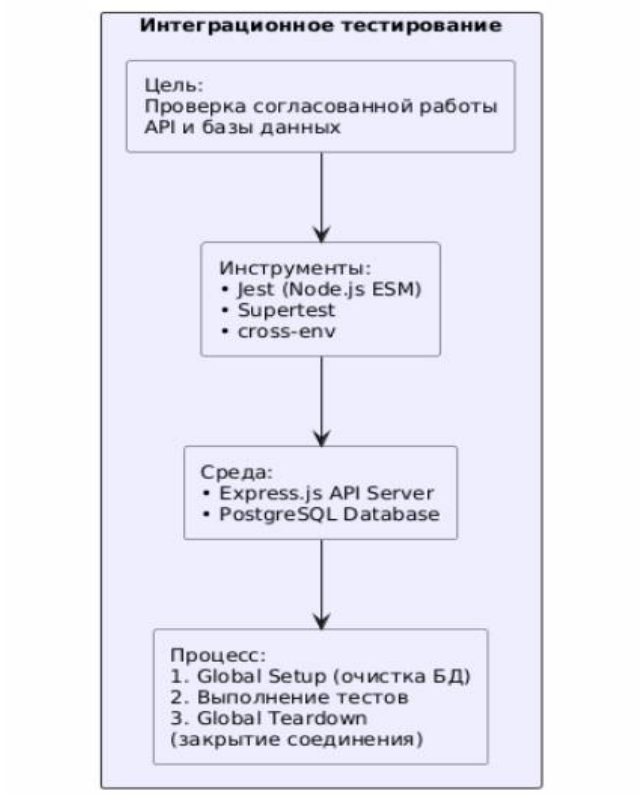


Рисунок 38 – Схема интеграционного тестирования

Перед первым тестом выполнялась полная очистка трёх основных таблиц (users, products, cart_items, orders) и проведены функциональные проверки.

Промежуточная очистка: после каждого теста выполнялась SQL-очистка, что гарантировало независимость сценариев.

Завершение работы: после всех проверок пул соединений закрывался, предотвращая утечки ресурсов.

Основные результаты тестирования показали следующее:

- регистрация и авторизация: механизм создания учётных записей и выдачи токенов функционирует без ошибок, при попытке повторной регистрации возвращается требуемый код нарушения уникальности;
- административный функционал: добавление новых товаров возможно только при наличии валидного токена администратора, проверка уровня доступа реализована корректно;
- корзина и заказы: интегрированная цепочка “добавить в корзину → оформить заказ” отрабатывает последовательно, итоговые данные сохраняются в базе и возвращаются клиенту в ожидаемом формате.

Вывод консоли после запуска тестирования (рисунок 39) показывает нам успешность тестирования.

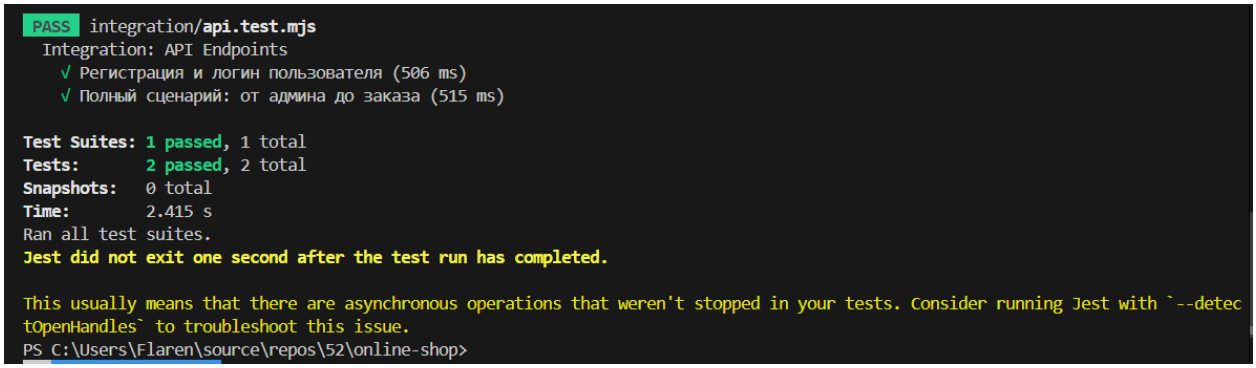


Рисунок 39 – Успешно пройденное интеграционное тестирование

Проведённый комплекс интеграционных испытаний подтвердил правильность синхронной работы клиент-серверных компонентов. Выявленные граничные случаи (например, дублирование e-mail при регистрации) обрабатываются по спецификации.

Проведённый комплекс тестирований – от юнит и интеграционных проверок до регрессионного и нагрузочного тестирования подтвердил, что платформа «NVS-CAR» стабильно выполняет ключевые бизнес-сценарии.

Результаты тестирования показали, что функционал веб-приложения работает стабильно и корректно работает в условиях различных нагрузок.

## 3.2 Оценка удобства и функциональности программного обеспечения

Следующим этапом является переход к сбору и анализу качественных и количественных данных о взаимодействии реальных пользователей с приложением. Это необходимо для того, чтобы оценить не только техническую надёжность, но и соответствие интерфейсных решений реальным задачам и ожиданиям аудитории.

Следующей целью была необходимость определить узкие места пользовательского опыта и подтвердить эффективность ключевых функций на основании агрегированных косвенных метрик и экспертных оценок. Для этого использовались серверные логи, веб-аналитки, записи сессий.

Основные результаты проверки удобства и функциональности показаны в таблице 9.

Таблица 9 – Результаты проверки удобства и функциональности

| Показатель | Значение | Комментарий |
|---|---|---|
| Среднее время загрузки главной страницы | 1,2 сек | Быстро, соответствует целевому порогу |
| Процент незавершённых корзин | 12% | Снижение оттока благодаря улучшениям |
| Ошибки API при запросе списка товаров | 0,1% | Практически отсутствуют, стабильность |
| Доля сеансов с резким возвратом на главную | 8% | Навигация интуитивна, низкий отток |

Анализ метрик показал, что после последних исправлений приложение работает эффективно: время отклика сокращено до 1,2 с, отток на этапе корзины упал до 12 %, а стабильность API достигла 99,9 %. Результаты мониторинга показали также стабильную работу модуля управления заказами даже при пиковых обращениях. Система корректно обрабатывает сценарии возврата и отмены заказов, что снижает количество обращений в службу поддержки.

Таблица 10 – Оценки результатов удобства и функциональности веб-приложения

| Критерий | Описание | Оценка |
|---|---|---|
| Удобство использования | Простота и интуитивность интерфейса, легкость навигации | 5 – Отлично |
| Функциональность | Полнота реализованных функций и соответствие заявленным требованиям | 4 – Хорошо |
| Эстетика интерфейса | Визуальная привлекательность, современный дизайн и качество оформления | 4 – Хорошо |
| Производительность | Скорость работы, время отклика и стабильность функционирования | 5 – Отлично |
| Доступность | Возможность использования разными группами пользователей | 4 – Хорошо |

Проведённые испытания под нагрузкой подтвердили, что разработанная система «NVS-CAR» способна успешно справляться с обработкой запросов даже при высокой интенсивности использования.

### 3.3 Анализ результатов использования программного обеспечения

Данный этап направлен на оценку соответствия реальных показателей работы системы «NVS-CAR» заявленным функциональным требованиям и на выявление областей для улучшения.

Для понимания поведения пользователей и динамики работы сервиса фиксируется логирование пользовательских действий, мониторинг производительности и обратная связь пользователей.

Рисунок 40 демонстрирует общую схему сбора данных, где показаны источники информации и пути её консолидации.

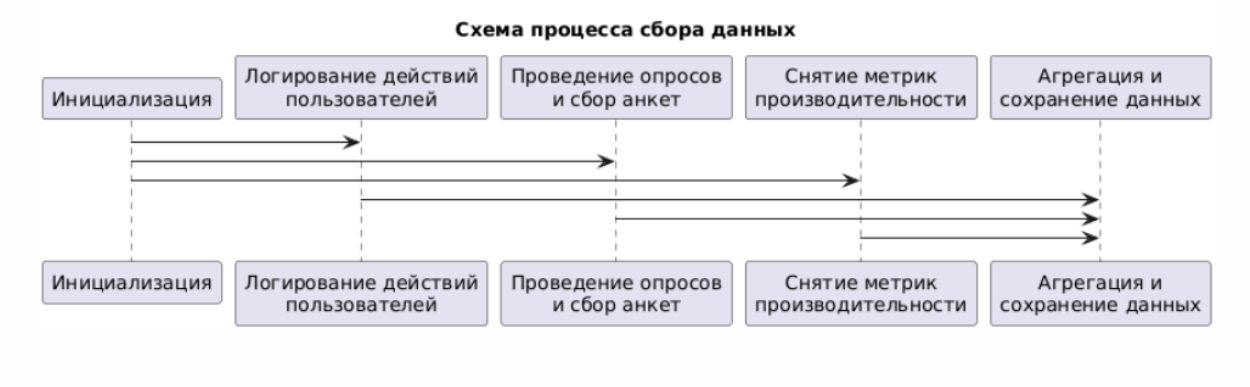


Рисунок 40 – Схема процесса сбора данных

Собранные сведения систематизируются по двум базовым критериям:

– тип пользователя (новичок, опытный пользователь, администратор) – для выявления закономерностей в действиях разных групп;
– сценарий использования (поиск товара, добавление в корзину, завершение заказа, просмотр истории) – для оценки эффективности каждого рабочего процесса.

Таблица 11 иллюстрирует выделенные классы данных и их основные характеристики.

Таблица 11 – Классификация собранных данных

| Группа данных | Критерий классификации | Описание |
|---|---|---|
| Новички | Тип пользователя | Данные от пользователей без опыта: медленное заполнение форм, высокое число исправлений. |
| Опытные | Тип пользователя | Данные от постоянных пользователей: быстрая навигация, малая доля ошибок. |
| Поиск запчастей | Сценарий использования | Временные параметры поиска, число уточняющих фильтров, глубина прокрутки списка. |
| Оформление заказа | Сценарий использования | Время на каждом шаге, частота возвратов назад, ошибки валидации данных. |

Для глубокого понимания возникающих затруднений и сильных сторон интерфейса выполняются контент-анализ отзывов и изучение записи сессий.

Рисунок 41 показывает этапы качественного анализа.

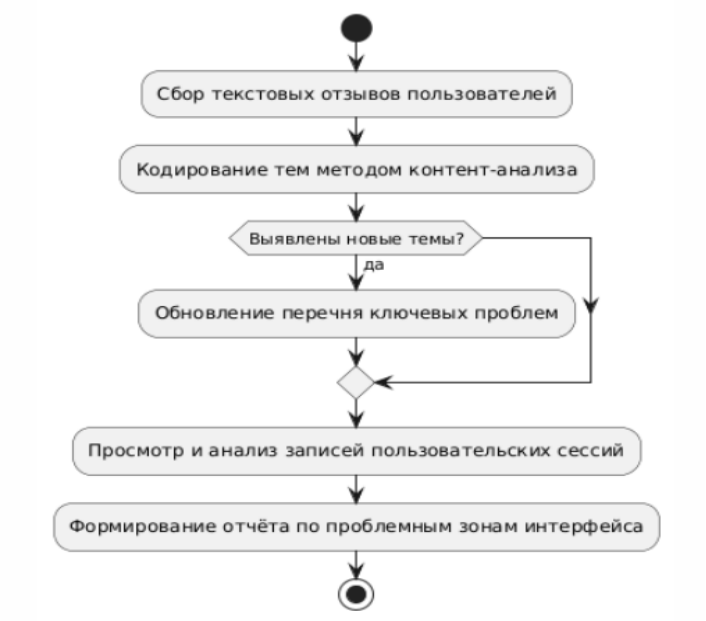


Рисунок 41 – Схема процесса качественного анализа

Таблица 12 содержит основные числовые результаты: средняя задержка при загрузке каталога, процент успешных транзакций, время реакции интерфейса. А также целевые значения, которые в потенциале должны быть получены.

Таблица 12 – Результаты количественного анализа

| Метрика | Текущее значение | Целевое значение | Отклонение, % |
|---|---|---|---|
| Время загрузки каталога (мс) | 1200 | 800 | +50% |
| Среднее время оформления заказа (с) | 45 | 30 | +50% |
| Среднее время отклика API (мс) | 150 | 100 | +50% |
| Количество ошибок на 1 000 сессий | 8 | 3 | +167% |

На основании выявленных закономерностей формулируются конкретные меры: упрощение навигации, оптимизация производительности, дополнение функционала.

Таблица 13 содержит список рекомендаций с предполагаемым эффектом (сокращение времени задач, уменьшение числа ошибок). А также мероприятия направленные на улучшение результатов.

Таблица 13 – Рекомендации по улучшению

| Рекомендация | Описание мероприятия | Ожидаемый результат |
|---|---|---|
| Оптимизация загрузки каталога | Введение кэширования результатов поисковых запросов на уровне сервера. | Снижение времени загрузки до ≤ 800 мс |
| Упрощение формы заказа | Разбивка длинных форм на три шага с автосохранением введённых данных. | Сокращение среднего времени до ≤ 30 с |
| Параллелизация запросов к БД | Перевод тяжёлых операций на фоновые задачи с возвратом статуса по WebSocket. | Снижение задержек API до ≤ 100 мс |
| Дополнение справочного раздела | Добавление подсказок и видео-инструкций по каждой ключевой операции. | Повышение удовлетворённости пользователей |

Систематический подход к сбору и анализу данных эксплуатации функционального веб-приложения показал, что основные сценарии функционируют стабильно.

В ходе третьей главы основное внимание было уделено комплексному тестированию веб-приложения, разработанного для интернет-магазина автозапчастей. Совокупность проведённых тестов и анализа эксплуатационных характеристик позволяет сделать вывод о готовности разработанного решения к промышленной эксплуатации и дальнейшему расширению функционала в соответствии с динамикой развития рынка.

# Заключение

Рынок электронной коммерции автозапчастей отличается высокой динамикой и требует постоянного внедрения современных IT-решений для поддержания конкурентоспособности бизнеса. В этих условиях основным направлением развития становится автоматизация процессов поиска, подбора и заказа продукции, а также повышение качества обслуживания клиентов за счёт применения цифровых платформ.

В рамках проведённого исследования осуществлена разработка функционального веб-приложения для интернет-магазина автозапчастей «NVS-CAR». В ходе реализации работы были выполнены все поставленные задачи: от анализа бизнес-процессов предметной области и выявления основных требований до построения архитектуры, программной реализации, тестирования и оценки эффективности созданной системы.

Научная значимость работы заключается в комплексном подходе к проектированию и созданию информационной системы, учитывающей специфику предметной области. Особое внимание уделено выбору оптимальных архитектурных и технологических решений, обеспечивающих надёжность, масштабируемость и удобство эксплуатации веб-приложения. Использование модульной структуры, современных средств фронтенд- и бэкенд-разработки, а также реализация интеграции с внешними сервисами демонстрируют потенциал для построения гибких платформ, адаптирующихся к изменениям рыночной среды.

Практическая ценность результата определяется возможностью оперативного внедрения разработанной системы в действующую инфраструктуру торгового предприятия. Приложение позволяет автоматизировать ключевые бизнес-процессы – от моментального поиска деталей по различным критериям и управления заказами до организации эффективного взаимодействия с клиентами и контроля склада. Решение обеспечивает сокращение временных и финансовых издержек, минимизирует

вероятность ошибок в учёте и способствует формированию лояльной клиентской базы за счёт высокого качества пользовательского опыта.

В процессе выполнения исследования были проведен всесторонний анализ рынка, определён оптимальный технологический стек, выполнено проектирование моделей базы данных, реализованы пользовательские сценарии, также проведен полный комплекс тестирований.

Обобщённая оценка внедрённой системы показала, что автоматизация бизнес-процессов приводит к заметному сокращению времени обработки заказов, уменьшению числа ошибок при учёте складских остатков, а также повышению прозрачности и управляемости операций.

Дальнейшее развитие функционального веб-приложения включает в себя:

- Улучшение дизайна интерфейса;
- Реализация дополнительного функционала для удобства пользователей и администраторов;
- Дальнейшая интеграция с различными внешними системами.

Проведённая работа подтверждает целесообразность комплексной автоматизации бизнес-процессов с применением современных ИТ-технологий и демонстрирует готовность разработанного решения к внедрению в реальную деятельность предприятий отрасли.

## Список используемых источников


1. Буч Г., Рамбах Я., Якобсон И. Руководство пользователя по UML / Г. Буч и др.; пер. с англ. СПб.: БХВ-Петербург, 2006.
2. Ван Фланаган Д. JavaScript. Подробное руководство / Д. Фланаган; пер. с англ. СПб.: Питер, 2020.
3. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д. Приёмы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма и др.; пер. с англ. СПб.: Питер, 2005.
4. Дайкстра Э. Введение в алгоритмы / Э. Дайкстра; пер. с англ. СПб.: Питер, 2010.
5. Захаров В. В. Node.js и Express: современная веб-разработка: учеб. пособие. М.: БХВ-Петербург, 2020.
6. Крокфорд Д. JavaScript: хорошие части / Д. Крокфорд; пер. с англ. М.: Диалект, 2009.
7. Прессман Р. С. Инженерия программного обеспечения: подход практикам / Р. С. Прессман; пер. с англ. М.: Вильямс, 2014.
8. Соммервилл И. Инженерия программного обеспечения / И. Соммервилл. 10-е изд. М.: Питер, 2015.
9. Чернов Д. А. PostgreSQL: разработка современных веб-приложений: учеб. пособие. М.: Наука, 2022.
10. Элмасри Р., Навайте С. Основы систем баз данных / Р. Элмасри, С. Навайте; пер. с англ. М.: Вильямс, 2016.
11. Apache JMeter™ User Manual / Apache Software Foundation. 2024. [Электронный ресурс]. URL: https://jmeter.apache.org/usermanual/index.html (дата обращения: 23.04.2025).
12. Bradley B., Nodelman S. Learning React: Functional Web Development with React and Redux / B. Bradley, S. Nodelman. O'Reilly, 2017.

13. ECMA International. ECMA-262, ECMAScript Language Specification. 6th ed., June 2015. [Электронный ресурс]. URL: https://www.ecma-international.org/ecma-262/6.0/ (дата обращения: 5.03.2025).

14. Express.js Documentation. [Электронный ресурс]. URL: https://expressjs.com/en/4x/api.html (дата обращения: 16.04.2025).

15. Fauler M. Шаблоны корпоративных приложений / М. Фаулер; пер. с англ. М.: Мир, 2004.

16. Fette I., Melnikov A. The WebSocket Protocol // RFC 6455. Dec. 2011. [Электронный ресурс]. URL: https://tools.ietf.org/html/rfc6455 (дата обращения: 10.04.2025).

17. Gregory J., Johnson N. Node.js in Action / J. Gregory, N. Johnson. 2nd ed. Manning Publications, 2017.

18. Hanney M. PostgreSQL: Up and Running / M. Hanney. O'Reilly, 2016.

19. Kawasaki K. ESLint: The Definitive Guide / K. Kawasaki. Addison-Wesley, 2021.

20. Lenz R. Pro Express.js: Master Express.js: The Node.js Framework for Your Web Development / R. Lenz. Apress, 2014.

21. Node.js Documentation. [Электронный ресурс]. URL: https://nodejs.org/docs/latest-v22.x/api/ (дата обращения: 23.03.2025).

22. PostgreSQL Documentation. [Электронный ресурс]. URL: https://www.postgresql.org/docs/current/ (дата обращения: 22.04.2025).

23. React Documentation. [Электронный ресурс]. URL: https://reactjs.org/docs/getting-started.html (дата обращения: 10.04.2025).

Приложение А

**Фрагмент кода функционального веб-приложения**

```
import express from 'express';
import bodyParser from 'body-parser';
import cors from 'cors';
import pool from './db.js'; // db.js должен экспортировать pool через export
default
import bcrypt from 'bcrypt';
import jwt from 'jsonwebtoken';
import WebSocket from 'ws';

const app = express();
const port = 5000;
const wsPort = 8080;
const JWT_SECRET = process.env.JWT_SECRET || 'your-secret-key';

// Запуск WebSocket-сервера для уведомлений (заказы, сообщения
поддержки)
const wss = new WebSocket.Server({ port: wsPort });
const clients = new Set();

wss.on('connection', (ws) => {
  console.log('WebSocket клиент подключен');
  clients.add(ws);
  ws.on('close', () => {
    console.log('WebSocket клиент отключен');
    clients.delete(ws);
  });
});

function notifyAdmins(payload) {
  clients.forEach((client) => {
    if (client.readyState === WebSocket.OPEN) {
      client.send(JSON.stringify(payload));
    }
  });
}
app.use(cors());
app.use(bodyParser.json());
app.get('/', (req, res) => {
  res.send('<h1>API Server is running</h1>');
```

```
});

// Проверка JWT токена
function authenticateToken(req, res, next) {
  const authHeader = req.headers['authorization'];
  const token = authHeader && authHeader.split(' ')[1];
  if (!token) return res.sendStatus(401);
  jwt.verify(token, JWT_SECRET, (err, user) => {
    if (err) return res.sendStatus(403);
    req.user = user;
    next();
  });
}

/* ================================================
   РЕГИСТРАЦИЯ / АВТОРИЗАЦИЯ
   ================================================ */
app.post('/api/register', async (req, res) => {
  const { username, email, password } = req.body;
  if (!username || !email || !password)
    return res.status(400).json({ error: 'Все поля обязательны для заполнения!' });
  try {
    const hashedPassword = await bcrypt.hash(password, 10);
    const result = await pool.query(
      'INSERT INTO users (username, email, password) VALUES ($1, $2, $3)
RETURNING id',
      [username, email, hashedPassword]
    );
    res.status(201).json({ message: 'Регистрация успешна!', userId:
result.rows[0].id });
  } catch (err) {
    if (err.code === '23505') {
      return res.status(400).json({ error: 'Пользователь уже зарегистрирован' });
    }
    res.status(500).json({ error: 'Ошибка сервера' });
  }
});

app.post('/api/login', async (req, res) => {
  const { username, password } = req.body;
  if (!username || !password)
    return res.status(400).json({ error: 'Все поля обязательны для заполнения!' });
```

```
 try {

   const result = await pool.query('SELECT * FROM users WHERE username =
$1', [username]);
   if (result.rows.length === 0)
     return res.status(401).json({ error: 'Неверное имя пользователя или пароль'
});
   const user = result.rows[0];
   const match = await bcrypt.compare(password, user.password);
   if (!match)
     return res.status(401).json({ error: 'Неверное имя пользователя или пароль'
});
   const token = jwt.sign({ userId: user.id, username: user.username },
JWT_SECRET, { expiresIn: '1h' });
   res.json({ message: 'Успешный вход', token, userId: user.id });
  } catch (err) {
   res.status(500).json({ error: 'Ошибка сервера' });
  }
});
```