**Федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский технический университет им. А.Н. Туполева-КАИ» (КНИТУ-КАИ)**

**ОТДЕЛЕНИЕ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**ДИПЛОМНЫЙ ПРОЕКТ**

**Тема:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Руководитель**

Преподаватель**\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*(должность) (подпись) (И.О. Фамилия)*

**Обучающийся**

4437**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_**

*(группа) (подпись) (И.О. Фамилия)*

**Специальность** 09.02.07 Информационные системы и программирование**\_\_\_**

(шифр и наименование специальности)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Казань 2024 год**

**АННОТАЦИЯ**

Данный документ является пояснительной запиской к выпускной квалификационной работе на тему «Веб-приложение для работы транспортно-логистической компании ООО «КамионЭкспресс»»

Целью дипломного проекта является разработка Веб-приложения транспортно-логистической компании.

Пояснительная записка состоит из введения, заключения, списка литературы, а также трех глав:

* постановки задачи;
* проектирования информационной системы;
* пользовательской документации.

Пояснительная записка содержит в себе 103 страницы, 47 рисунка, 2 приложения, 20 литературных источников.

**Содержание**

[Содержание 1](#_Toc166826232)

[Введение 2](#_Toc166826233)

[Глава 1 3](#_Toc166826234)

[1.1 Формулировка задачи 3](#_Toc166826235)

[1.2 Уровень критичности приложения 4](#_Toc166826236)

[1.3 Надежность функционирования 5](#_Toc166826237)

[1.4 Требования к безопасности 6](#_Toc166826238)

[1.5 Аппаратные ограничения 7](#_Toc166826239)

[1.6 Интерфейсы с другими приложениями 8](#_Toc166826240)

[1.7 Требования к распараллеливанию операций 9](#_Toc166826241)

[1.8 Функции аудита и управления 10](#_Toc166826242)

[1.9 Высокоуровневые требования 11](#_Toc166826243)

[1.10 Требования к языку программирования и коммуникационным протоколам 12](#_Toc166826244)

[Глава 2 13](#_Toc166826245)

[**2.1 Язык программирования** 13](#_Toc166826246)

[**2.2 Среды разработки** 14](#_Toc166826247)

[**2.3 Фреймворки:** 15](#_Toc166826248)

[**2.4 Системы управления базами данных (СУБД):** 15](#_Toc166826249)

[**2.5 Открытые программные библиотеки:** 15](#_Toc166826250)

[**2.6 Обоснование выбора инструментальных средств:** 15](#_Toc166826251)

Введение

Логистическая отрасль — это сложная глобальная сеть, требующая эффективного управления задачами для обеспечения бесперебойной доставки товаров и услуг. Традиционные методы управления могут оказаться несостоятельными в условиях растущей сложности и масштабов логистических операций.

В данном контексте возникает потребность в инновационных решениях, способных оптимизировать управление задачами и повысить эффективность в логистической отрасли. Онлайн-сервисы для улучшения и управления задачами становятся необходимостью, предоставляя инструменты для централизованного планирования, отслеживания прогресса и коммуникации между участниками процесса.

В предложенном исследовании будет разработан комплексный онлайн-сервис, предназначенный для оптимизации управления задачами в логистических операциях. Основываясь на современных технологиях и передовых методах управления проектами, этот сервис позволит участникам процесса эффективно планировать ресурсы, координировать действия и отслеживать прогресс выполнения задач.

Цель исследования — разработать программное решение, которое отвечает потребностям логистической отрасли в управлении задачами, обеспечивая высокий уровень надежности, безопасности и удобства использования. Исследование определит основные требования к сервису, его функциональные возможности и методы реализации с учетом специфики логистических операций и потребностей пользователей.

Это исследование имеет практическое значение для логистической отрасли, поскольку предоставляет новые инструменты и методы управления задачами, способствующие повышению эффективности, снижению затрат и улучшению обслуживания клиентов.

**ГЛАВА 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

Добавить описание

1.1 Формулировка задачи

Логистические проекты представляют собой сложные многосторонние процессы, требующие тщательного планирования, организации и управления. В рамках этих проектов управление задачами играет ключевую роль в обеспечении успешного завершения работ в срок и в соответствии с высокими стандартами качества. Однако, с увеличением масштабов и сложности перевозок, возникают новые вызовы и потребности, которые не всегда удается эффективно решить с помощью традиционных методов управления.

Целью данного проекта является создание онлайн сервиса, который будет способствовать улучшению управления задачами в процессе транспортировки. Основная задача сервиса заключается в предоставлении инструментов и функций, которые позволят всем участникам проекта эффективно планировать, отслеживать и управлять задачами, связанными с процессом транспортировки. При этом особое внимание уделяется оптимизации рабочего процесса, сокращению временных и финансовых затрат, а также повышению качества и прозрачности исполнения задач.

В рамках данного проекта будут рассмотрены следующие аспекты:

Анализ существующих проблем и вызовов в управлении задачами в логистике, выявление основных потребностей и требований пользователей.

Разработка функциональных и технических требований к создаваемому сервису с учетом специфики логистической отрасли и потребностей пользователей.

Проектирование и разработка онлайн сервиса, включая выбор технологических платформ, архитектуры системы, интерфейсов и функциональных возможностей.

Тестирование и оптимизация сервиса с целью обеспечения высокого уровня надежности, производительности и удобства использования.

Обеспечение поддержки и дальнейшего развития сервиса, включая добавление новых функций и адаптацию к изменяющимся потребностям пользователей и рынка.

1.2 Уровень критичности приложения

В контексте управления проектами в логистической отрасли, уровень критичности приложения играет решающую роль в обеспечении успешного завершения транспортировки в срок и в рамках бюджета. Эффективное приложение для управления задачами в процессе логистики должно обладать высоким уровнем критичности, поскольку оно является основным инструментом для планирования, координации и контроля всех аспектов транспортировки. Рассмотрим ключевые аспекты, определяющие уровень критичности данного приложения:

Своевременное выполнение работ: Приложение играет решающую роль в планировании и отслеживании сроков выполнения задач. Неисправность или сбой в работе приложения может привести к нарушению графика работ, что в свою очередь может вызвать серьезные финансовые потери и негативно отразиться на репутации участников проекта.

Прозрачность и коммуникация: Приложение обеспечивает прозрачность в работе и эффективную коммуникацию между всеми участниками проекта, что является ключевым фактором для успешного сотрудничества и достижения общих целей. Недоступность или неисправность приложения может привести к проблемам в обмене информацией и согласовании действий между различными участниками проекта.

Повышенные риски приостановки работ: В случае проблем с приложением возникает риск простоя работ, что может привести к дополнительным затратам на переопределение приоритетов, перенос сроков или даже приостановку процесса транспортировки. Это может привести к серьезным финансовым потерям и задержкам в реализации проекта.

Необходимость оперативного реагирования: в условиях динамичной логистической отрасли важно, чтобы приложение было доступно и функционировало без сбоев в любое время суток. Быстрое реагирование на изменения в планах и ситуации является критическим аспектом успеха, а приложение является основным инструментом для организации эффективной работы.

Финансовые и репутационные риски: Неисправность или сбой в работе приложения может привести к серьезным финансовым потерям для компании и ее партнеров, а также негативно отразиться на репутации как заказчика, так и исполнителя проекта. Это может привести к потере доверия со стороны клиентов и партнеров, а также к юридическим проблемам и судебным искам.

**1.3 Надежность функционирования**

Стабильность и отказоустойчивость: Программный продукт должен обеспечивать стабильную работу в течение всего срока транспортировки, минимизируя риск сбоев и отказов. Это включает в себя разработку и тщательное тестирование программного кода, а также внедрение механизмов автоматического обнаружения и восстановления после возможных сбоев или отказов системы. Например, использование механизмов резервного копирования данных, репликации серверов и балансировки нагрузки для обеспечения непрерывной работы приложения даже в случае отказа отдельных компонентов.

Безопасность данных: Надежность функционирования также предполагает высокий уровень защиты данных от несанкционированного доступа, утечек или повреждений. Приложение должно использовать современные методы шифрования данных, механизмы аутентификации и авторизации пользователей, а также регулярное обновление системы безопасности для предотвращения угроз и атак.

Мониторинг и управление: для обеспечения надежности работы приложения необходимо постоянное мониторинг его состояния и производительности. Это позволяет оперативно выявлять проблемы и сбои в работе, а также принимать меры по их устранению. Для этого могут использоваться системы мониторинга и управления, автоматические уведомления о проблемах, а также возможность удаленного управления и восстановления работы приложения.

Резервное копирование и восстановление: для обеспечения безопасности данных и возможности восстановления после чрезвычайных ситуаций, приложение должно регулярно создавать резервные копии данных и обеспечивать возможность их восстановления. Это позволяет минимизировать потери информации и сокращать время простоя в случае сбоев или инцидентов.

Тестирование и обновления: Надежность функционирования приложения также зависит от качества его тестирования и регулярного обновления. Разработчики должны проводить тщательное функциональное, нагрузочное и безопасностное тестирование приложения перед выпуском его в эксплуатацию, а также регулярно обновлять программное обеспечение для исправления выявленных ошибок и уязвимостей.

* 1. Требования к безопасности

Шифрование данных: Все данные, передаваемые между клиентом и сервером, должны быть зашифрованы с использованием современных алгоритмов шифрования, таких как SSL/TLS, чтобы предотвратить перехват и утечку конфиденциальной информации.

Механизмы аутентификации и авторизации: Приложение должно предоставлять надежные механизмы аутентификации пользователей, такие как пароли, двухфакторная аутентификация или биометрическая идентификация, чтобы гарантировать, что только авторизованные пользователи имеют доступ к системе. Кроме того, должны быть установлены четкие права доступа для контроля разграничения доступа к данным в зависимости от роли пользователя.

Защита от внешних угроз: Приложение должно быть защищено от внешних угроз, таких как атаки типа SQL инъекции, кросс-сайт скриптинга (XSS), кражи сеансов и других распространенных видов кибератак.

Регулярное обновление и мониторинг безопасности: Разработчики должны регулярно обновлять приложение, включая исправление обнаруженных уязвимостей и обновление зависимостей, чтобы минимизировать риски для безопасности. Кроме того, необходимо внедрить систему мониторинга безопасности, которая будет отслеживать аномальную активность и предупреждать о возможных угрозах.

Физическая безопасность серверов и инфраструктуры: важно обеспечить физическую безопасность серверов и инфраструктуры, на которых развернуто приложение, чтобы предотвратить несанкционированный доступ к оборудованию и возможные физические атаки.

Соответствие регулятивным требованиям: Приложение должно соответствовать существующим регулятивным требованиям в области защиты данных, таким как GDPR, HIPAA и другим стандартам безопасности данных, применимым к строительной отрасли.

* 1. Аппаратные ограничения

Характеристики серверов и вычислительного оборудования: Программный продукт должен быть разработан с учетом характеристик серверов, на которых будет развернут, таких как процессорная мощность, объем оперативной памяти, объем хранилища и пропускная способность сети. Это позволяет обеспечить эффективную работу приложения при различных нагрузках и объемах данных.

Требования к операционной системе и окружению: Программный продукт должен быть совместим с определенными операционными системами и окружением, на которых планируется его использовать. Необходимо учитывать особенности различных операционных систем (Windows, Linux, macOS) и поддерживать соответствующие версии и конфигурации.

* 1. Интерфейсы с другими приложениями

Интеграция с системами управления документами: Приложение должно обеспечивать возможность интеграции с системами управления документами, такими как Microsoft Word, Microsoft Excel и другими. Это позволит пользователям удобно хранить, обмениваться и управлять документацией проекта, спецификации и техническая документация.

Интеграция с ERP и CRM системами: Важным требованием является интеграция с системами управления предприятием (ERP) и управления взаимоотношениями с клиентами (CRM), такими как SAP, Oracle, Salesforce и другими. Это позволит автоматизировать процессы управления ресурсами, финансами, закупками и обслуживанием клиентов, обеспечивая единый и централизованный доступ к информации.

Интеграция с системами автоматизации процессов: Приложение должно иметь возможность интеграции с системами автоматизации процессов, такими как Microsoft Power Automate, Zapier и другими. Это позволит создавать автоматизированные рабочие процессы и потоки работы, уменьшая ручной труд и повышая эффективность бизнес-процессов.

1.7 Требования к распараллеливанию операций

Многопоточность и асинхронность: Приложение должно поддерживать многопоточность и асинхронное выполнение операций, что позволит одновременно обрабатывать несколько запросов от пользователей и выполнять длительные операции в фоновом режиме, не блокируя основной поток работы приложения.

Распределенные вычисления: если приложение работает в распределенной среде, например, с использованием облачных вычислений, необходимо обеспечить возможность распределенных вычислений для эффективного использования вычислительных ресурсов и увеличения производительности.

Оптимизация запросов к базе данных: при работе с базой данных необходимо оптимизировать запросы и операции чтения/записи данных, чтобы минимизировать время выполнения запросов и снизить нагрузку на базу данных. Это может включать в себя кэширование данных, предзагрузку данных и оптимизацию структуры базы данных.

Параллельная обработка задач: если приложение обрабатывает большое количество задач или операций, необходимо реализовать механизмы параллельной обработки задач, например, с использованием очередей задач или пула потоков, чтобы обеспечить равномерное распределение нагрузки и ускорить обработку задач.

Управление ресурсами: при распараллеливании операций необходимо учитывать доступные ресурсы, такие как процессорное время, оперативная память и сетевая пропускная способность, и эффективно управлять ими, чтобы избежать перегрузки системы и обеспечить стабильную работу приложения.

* 1. Функции аудита и управления

Отслеживание действий пользователей: Система аудита должна вести журнал действий пользователей, включая вход в систему, просмотр, создание, редактирование и удаление задач, обновление статусов и другие операции. Это позволяет отслеживать активность пользователей, выявлять потенциальные проблемы и конфликты, а также обеспечивать юридическую прозрачность и ответственность.

Хранение и анализ данных аудита: для обеспечения надежности и достоверности данных аудита необходимо предусмотреть систему хранения и анализа журналов действий пользователей. Это позволяет быстро находить и анализировать информацию о прошлых событиях, выявлять тренды и узкие места в процессе управления задачами.

Настройка и контроль доступа к данным аудита: Система управления должна обеспечивать возможность настройки прав доступа к данным аудита, чтобы предотвратить несанкционированный доступ и изменение истории действий пользователей. Это позволяет обеспечить конфиденциальность и целостность данных аудита, а также соответствие требованиям защиты информации.

Генерация отчетов и аналитика: Функции управления должны включать в себя возможность генерации различных отчетов и аналитики на основе данных аудита, таких как отчеты о производительности, эффективности использования ресурсов, анализ общей активности пользователей и т. д. Это позволяет оценить текущее состояние проекта, выявить проблемные места и принять информированные решения.

Интеграция с системами управления ресурсами и аналитическими инструментами: Функции аудита и управления должны быть интегрированы с другими системами управления ресурсами и аналитическими инструментами, такими как системы управления предприятием (ERP), системы управления проектами (PMS) и инструменты бизнес-аналитики. Это позволяет обеспечить единый и централизованный доступ к информации и аналитике, повышая эффективность принятия решений и управления проектом.

1.9 Высокоуровневые требования

Интуитивно понятный пользовательский интерфейс: Пользовательский интерфейс приложения должен быть простым, интуитивно понятным и удобным в использовании даже для неопытных пользователей. Это позволяет минимизировать время обучения и повышает удовлетворенность пользователей от работы с приложением.

Масштабируемость и гибкость: Приложение должно быть масштабируемым и гибким, чтобы адаптироваться к различным потребностям и масштабам строительных проектов. Это включает в себя возможность управления большим объемом данных, поддержку различных типов проектов и гибкую настройку функционала под конкретные потребности пользователей.

Доступность через различные устройства и платформы: Приложение должно быть доступным через различные устройства и платформы, включая компьютеры, мобильные устройства и планшеты, а также различные операционные системы (Windows, iOS, Android). Это обеспечивает удобство использования и доступность для пользователей в любое время и в любом месте.

Надежность и высокая производительность: Приложение должно обеспечивать высокий уровень надежности и производительности, чтобы гарантировать стабильную работу даже при высоких нагрузках и большом количестве пользователей. Это включает в себя быстрый отклик интерфейса, минимальное время простоя и отказоустойчивость системы.

Безопасность и конфиденциальность данных: Приложение должно обеспечивать высокий уровень безопасности и конфиденциальности данных, включая шифрование информации, механизмы аутентификации и авторизации пользователей, а также защиту от внешних угроз и атак.

Поддержка современных технологий и стандартов: Приложение должно быть разработано с использованием современных технологий и стандартов разработки программного обеспечения, что обеспечивает его долгосрочную поддержку, масштабируемость и совместимость с другими системами и приложениями.

1.10 Требования к языку программирования и коммуникационным протоколам

Язык программирования: Выбор языка программирования должен учитывать требования к производительности, масштабируемости, безопасности и командной разработке. Для разработки онлайн сервиса для управления задачами в проекте строительства часто используются языки программирования, такие как Python, JavaScript (включая Node.js для серверной части), Java, C# и PHP. Эти языки обладают хорошей поддержкой и обширными библиотеками для разработки веб-приложений, а также обеспечивают высокую производительность и удобство разработки.

Фреймворки и библиотеки: при выборе языка программирования важно также учитывать наличие подходящих фреймворков и библиотек, которые облегчают разработку и обеспечивают необходимый функционал.

Коммуникационные протоколы: Приложение должно поддерживать современные коммуникационные протоколы для обмена данными между клиентом и сервером, а также для взаимодействия с другими приложениями и сервисами. В качестве основных протоколов для веб-приложений обычно используются HTTP/HTTPS для передачи данных по сети и RESTful API для обеспечения взаимодействия между клиентом и сервером. Для обмена сообщениями и асинхронного взаимодействия могут применяться протоколы, такие как WebSocket или MQTT.

Безопасность протоколов: При выборе коммуникационных протоколов необходимо учитывать их безопасность и защищенность от угроз и атак. Например, при использовании HTTP/HTTPS следует обеспечить защищенность передаваемых данных с помощью шифрования (SSL/TLS), а при реализации RESTful API необходимо соблюдать принципы безопасности, такие как аутентификация и авторизация пользователей, защита от инъекций и прочие.

Совместимость и поддержка стандартов: Коммуникационные протоколы должны быть совместимы с существующими стандартами и требованиями, чтобы обеспечить совместимость с другими системами и сервисами. Например, при разработке веб-приложения важно соблюдать стандарты HTTP/HTTPS, а при реализации API следует руководствоваться принципами RESTful или GraphQL.

**ГЛАВА 2 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ**

Добавить краткое описание главы

**2.1 Выбор средств реализации**

## **2.1.1 Язык программирования**

TypeScript: Основной язык программирования для всего стека. TypeScript был выбран из-за его статической типизации, что позволяет избежать многих ошибок на этапе разработки. Статическая типизация помогает обнаруживать ошибки на этапе написания кода, улучшает автозаполнение и навигацию по коду, что значительно ускоряет процесс разработки. TypeScript также хорошо интегрируется с большинством современных фреймворков и библиотек, таких как Vue и NestJS.

Статическая типизация: TypeScript добавляет статическую типизацию к JavaScript, что позволяет выявлять ошибки на этапе компиляции. Это снижает вероятность ошибок в рантайме и улучшает качество кода.

Поддержка современных возможностей JavaScript: TypeScript поддерживает все современные возможности JavaScript, включая синтаксис ES6/ES7, что позволяет использовать новейшие функции языка, такие как стрелочные функции, асинхронные функции, декораторы и модули.

Улучшенная IDE поддержка: TypeScript обеспечивает отличную поддержку в IDE, таких как WebStorm и Visual Studio Code, предоставляя функции автодополнения, рефакторинга, навигации по коду и отладки. Это значительно ускоряет процесс разработки и облегчает работу с большими проектами.

Совместимость с JavaScript: TypeScript совместим с любым существующим JavaScript-кодом. Это позволяет постепенно внедрять TypeScript в существующие проекты и использовать его совместно с библиотеками и фреймворками, написанными на JavaScript.

Обширная экосистема и сообщество: TypeScript имеет широкую экосистему и активно развивающееся сообщество. Множество популярных библиотек и фреймворков, таких как React, Angular и Vue.js, имеют встроенную поддержку TypeScript или предоставляют официальные типы для интеграции.

Поддержка ООП: TypeScript поддерживает объектно-ориентированное программирование, включая классы, интерфейсы, наследование и модификаторы доступа. Это позволяет создавать масштабируемые и поддерживаемые архитектуры приложений.

Интероперабельность: TypeScript позволяет использовать существующие JavaScript-библиотеки и фреймворки без каких-либо изменений. Также существуют типовые определения для большинства популярных библиотек, что упрощает их интеграцию в TypeScript-проекты.

## **2.1.2 Среды разработки**

WebStorm - это мощная интегрированная среда разработки (IDE) для JavaScript и TypeScript от компании JetBrains. Она была выбрана для данного проекта по нескольким причинам:

Поддержка множества языков и технологий:

WebStorm поддерживает все современные веб-технологии, включая JavaScript, TypeScript, React, Angular, Vue.js, HTML, CSS и многие другие. Это позволяет использовать единую среду разработки для всех частей приложения.

Встроенная поддержка TypeScript:

WebStorm имеет отличную поддержку TypeScript, включая компиляцию, автодополнение, навигацию и рефакторинг. Это делает работу с TypeScript более удобной и эффективной.

Поддержка инструментов сборки и автоматизации:

WebStorm интегрируется с популярными инструментами сборки и автоматизации, такими как Webpack, Gulp, Grunt и npm scripts. Это упрощает настройку и использование процессов сборки и деплоя проекта.

Удобные средства для отладки:

WebStorm предоставляет мощные инструменты для отладки, включая возможность установки точек останова, пошаговую отладку, просмотр переменных и выражений, отладку асинхронного кода и многое другое

Расширяемость и поддержка плагинов:

WebStorm имеет богатую экосистему плагинов, которые позволяют расширять функциональность IDE. Существует множество плагинов для различных фреймворков, библиотек и инструментов, что делает WebStorm гибким и настраиваемым под конкретные нужды проекта.

Интеграция с инструментами тестирования:

WebStorm поддерживает интеграцию с популярными инструментами для тестирования, такими как Jest, Mocha, Karma и другими. Это позволяет легко запускать и отлаживать тесты непосредственно из IDE

Постоянные обновления и поддержка:

WebStorm регулярно обновляется и получает новые функции и улучшения. JetBrains предоставляет отличную поддержку для пользователей, включая документацию, форумы, блоги и техническую поддержку. Это гарантирует, что разработчики всегда будут иметь доступ к актуальной и надежной среде разработки.

**2.1.3 Фреймворки:**

Next.js:

Next.js — это фреймворк для React, который позволяет создавать серверные рендеринговые (SSR) и статические веб-приложения. Он был выбран для данного проекта по следующим причинам:

Серверный рендеринг (SSR):

Next.js поддерживает серверный рендеринг, что позволяет улучшить SEO, так как поисковые системы могут индексировать предварительно отрендеренные страницы. Это также улучшает производительность, так как страницы рендерятся на сервере и отправляются клиенту в готовом виде.

Статическая генерация (SSG):

Next.js поддерживает статическую генерацию, что позволяет создавать статические страницы во время сборки проекта. Это увеличивает производительность и уменьшает нагрузку на сервер, так как статические страницы могут быть кэшированы и быстро доставляться пользователям.

Автоматическая оптимизация:

Next.js автоматически оптимизирует страницы и ресурсы, что улучшает производительность и время загрузки. Это включает оптимизацию изображений, кода и других ресурсов.

Маршрутизация на основе файлов:

Next.js использует файловую систему для маршрутизации, что упрощает создание и управление маршрутами. Каждый файл в папке pages автоматически становится маршрутом приложения.

Поддержка TypeScript:

Next.js имеет встроенную поддержку TypeScript, что обеспечивает строгую типизацию и улучшает качество кода. Это делает разработку более надежной и предсказуемой.

Расширяемость:

Next.js предоставляет возможность легко настраивать и расширять конфигурацию с помощью плагинов и пользовательских настроек. Это позволяет адаптировать проект под конкретные требования и использовать дополнительные возможности.

Интеграция с API и микросервисами:

Next.js позволяет легко интегрировать внешние API и микросервисы, что делает его удобным для создания масштабируемых приложений с распределенной архитектурой.

Vue:

Vue — это прогрессивный фреймворк для разработки пользовательских интерфейсов и одностраничных веб-приложений на языке JavaScript. Она была выбрана для данного проекта по следующим причинам:

Компонентный подход:

Vue использует компонентный подход к разработке интерфейсов, что позволяет создавать повторно используемые и изолированные компоненты. Это упрощает разработку, тестирование и поддержку приложения.

Виртуальный DOM:

Vue использует виртуальный DOM для оптимизации производительности. Это позволяет минимизировать количество обновлений реального DOM и повышает скорость работы приложения.

Однонаправленный поток данных:

Vue реализует однонаправленный поток данных, что делает состояние приложения более предсказуемым и упрощает отладку. Данные передаются сверху вниз, что упрощает управление состоянием.

Широкая экосистема:

Vue имеет обширную экосистему библиотек и инструментов, таких как Vuex, Vue Router, Material-UI и многие другие. Это позволяет быстро добавлять новые возможности и интегрировать сторонние решения.

Поддержка TypeScript:

Vue поддерживает TypeScript, что обеспечивает строгую типизацию и улучшает качество кода. Это делает разработку более надежной и предсказуемой.

NestJS:

NestJS — это фреймворк для создания масштабируемых серверных приложений на Node.js, основанный на TypeScript и вдохновленный архитектурными паттернами Angular. Он был выбран для данного проекта по следующим причинам:

Модульная архитектура:

NestJS использует модульную архитектуру, что позволяет разделить приложение на независимые и легко управляемые модули. Это упрощает разработку, тестирование и поддержку приложения.

Инверсия управления (IoC) и внедрение зависимостей (DI):

NestJS предоставляет встроенные механизмы IoC и DI, что позволяет легко управлять зависимостями и улучшает тестируемость кода. Это делает код более модульным и гибким.

Поддержка TypeScript:

NestJS полностью написан на TypeScript и поддерживает все его возможности, включая строгую типизацию, декораторы и современные возможности языка. Это улучшает качество кода и делает его более предсказуемым.

Совместимость с другими фреймворками:

NestJS совместим с популярными фреймворками и библиотеками, такими как Express, Fastify, GraphQL, WebSockets и другими. Это позволяет легко интегрировать дополнительные возможности и использовать существующие решения.

Микросервисная архитектура:

NestJS поддерживает создание микросервисов и распределенных систем, что делает его подходящим для создания масштабируемых и высоконагруженных приложений. Это позволяет строить сложные системы с четко определенными границами и взаимодействием между компонентами.

Управление конфигурацией:

NestJS предоставляет мощные инструменты для управления конфигурацией приложения, что позволяет легко настраивать параметры и адаптировать приложение под различные среды.

Интеграция с базами данных:

NestJS поддерживает интеграцию с различными базами данных через ORM и другие библиотеки, такие как TypeORM, Sequelize и Prisma. Это позволяет легко работать с данными и управлять базами данных.

Sequelize:

Sequelize — это современный ORM (Object-Relational Mapping) для Node.js и TypeScript, который упрощает взаимодействие с базами данных и обеспечивает высокий уровень производительности и безопасности. Он был выбран для данного проекта по следующим причинам:

Автоматическая генерация типов:

Sequelize автоматически генерирует TypeScript типы на основе схемы базы данных, что обеспечивает строгую типизацию и уменьшает количество ошибок. Это делает работу с данными более предсказуемой и безопасной.

Простота использования:

Sequelize предоставляет интуитивно понятный и лаконичный API для выполнения CRUD операций, что упрощает работу с базой данных. Это позволяет быстро и эффективно взаимодействовать с данными.

Миграции базы данных:

Sequelize поддерживает миграции базы данных, что позволяет легко управлять схемой базы данных и вносить изменения. Это обеспечивает консистентность и удобство управления структурой данных.

Поддержка различных баз данных:

Sequelize поддерживает работу с различными базами данных, включая PostgreSQL, MySQL, SQLite и SQL Server. Это позволяет выбрать наиболее подходящую базу данных для проекта и легко переключаться между ними.

Интеграция с GraphQL:

Sequelize легко интегрируется с GraphQL, что позволяет создавать мощные и гибкие API. Это упрощает работу с данными и обеспечивает высокую производительность запросов.

Высокая производительность:

Sequelize обеспечивает высокую производительность благодаря оптимизированному коду и продуманной архитектуре. Это позволяет быстро выполнять сложные запросы и обрабатывать большие объемы данных.

**2.1.4 Системы управления базами данных (СУБД):**

PostgreSQL: Реляционная СУБД, выбранная за её надежность, масштабируемость и поддержку ACID-транзакций. PostgreSQL поддерживает сложные запросы и предоставляет множество инструментов для работы с данными, что делает её идеальной для использования в веб-приложениях.

## **2.1.5 Открытые программные библиотеки:**

**Sequelize:** ORM для TypeScript и Node.js, который обеспечивает удобную работу с базой данных и поддерживает PostgreSQL. Sequelize позволяет легко определять схемы данных, выполнять запросы и миграции, обеспечивая высокий уровень абстракции и производительности.

## **2.1.6 Обоснование выбора инструментальных средств:**

Выбор указанных инструментов и технологий был основан на следующих критериях:

Поддержка TypeScript: Все выбранные технологии и инструменты имеют отличную поддержку TypeScript, что позволяет использовать его преимущества на всех уровнях стека. Это обеспечивает единообразие в кодовой базе и улучшает совместимость между различными компонентами системы.

Производительность: Next.js и React обеспечивают высокую производительность фронтенд-приложения, благодаря оптимизированному рендерингу и возможности серверного рендеринга. NestJS и PostgreSQL обеспечивают производительность и надежность серверной части, благодаря своей архитектуре и поддержке высоких нагрузок.

Сообщество и документация: У всех выбранных инструментов есть активное сообщество и хорошая документация, что облегчает процесс разработки и решения возникающих проблем. Это позволяет разработчикам быстро находить ответы на вопросы и использовать лучшие практики.

Масштабируемость: Выбранные технологии позволяют легко масштабировать приложение как на уровне фронтенда, так и на уровне бекенда. Next.js поддерживает статическую генерацию страниц и серверный рендеринг, что упрощает масштабирование веб-приложений. NestJS и PostgreSQL обеспечивают поддержку горизонтального и вертикального масштабирования серверных приложений.

2.2 Выявление требований к программному обеспечению

Заказчик системы – ООО ТЛК "КАМИОНЭКСПРЕСС".

Целью данной разработки является создание функционирующих модулей карты, калькулятора, заявки, новостей, клиента, сотрудников, администратора.

Разрабатываемое приложение в дальнейшем способствует уменьшению работы логиста и менеджера по продажам. Использование разрабатываемых модулей клиента и сотрудника позволит создавать, редактировать и отслеживать заявки на грузоперевозку. В последствии облегчить процесс взаимодействия клиента с сотрудниками.

В области деятельности субъекта реализуются следующие бизнеспроцессы:

— заказ и планирование;

— подбор и управление транспортом;

— коммуникация и отслеживание грузов.

Необходимость автоматизировать процесс взаимодействия клиента с сотрудниками. Благодаря автоматизации сотрудник в разы сокращает время на обслуживание клиента.

Разрабатываемая программа система должна представлять собой веб-сайт.

Существующих аналогов реализуемого программного решения нет. Укажите преимущества и недостатки аналогов.

Для выбранной темы были выявлены следующие пользовательские требования:

Таблица 1 – Выявленные пользовательские требования

|  |  |
| --- | --- |
| № | Пользовательские требования |
| 1 | Возможность отправлять заявки |
| 2 | Возможность посчитать стоимость грузоперевозки |
| 3 | Возможность отслеживать машины с грузом |
| 4 | Возможность администратора на добавлять новости |
| 5 | Возможность отслеживать выполнение заявки |
| 6 | Возможность администрировать заявку |
| 7 | Возможность редактирования калькулятора |

Для разрабатываемой системы были выявлены следующие акторы, взаимодействующие с ней:

1) Клиент;

2) Сотрудник;

3) Администратор.

Описание акторов модулей программной системы учета успеваемости представлено в таблице 2.

Таблица 2 – Выявление акторов

|  |  |
| --- | --- |
| Акторы | Краткое описание |
| Клиент | Физическое или юридическое лицо |
| Сотрудник | Менеджер по продажам, который обсуживает клиентов. Логист, который выстраивает выгодные маршруты и сообщает их продажнику. |
| Администратор | Особый пользователь, который имеет расширенные права для выполнения специальных задач – администрирования и, в зависимости от метода, некоторого редактирования контента |

Выявленные варианты использования для ранее указанных акторов представлены в таблице 3.

Таблица 3 – Выявленные варианты использования

|  |  |  |
| --- | --- | --- |
| Основной актор | Наименование | Формулировка |
| Клиент | Авторизоваться | Клиент при помощи логина и пароля может зайти в приложение под собственной учетной записью |

Продолжение таблицы 3

|  |  |  |
| --- | --- | --- |
| Основной актор | Наименование | Формулировка |
| Сотрудник | Авторизоваться | Сотрудник при помощи логина и пароля может зайти в приложение под собственной учетной записью |
| Администратор | Авторизоваться | Администратор при помощи логина и пароля может зайти в приложение под собственной учетной записью |
| Клиент | Просмотреть личные данные | Клиент может просматривать свои личные данные в соответствующем разделе личного кабинета |
| Клиент | Создание заявки | Клиент может отправить заявку на грузоперевозку |
| Сотрудник | Обработка заявки | Сотрудник обрабатывает заявку клиента. |

Продолжение таблицы 3

|  |  |  |
| --- | --- | --- |
| Основной актор | Наименование | Формулировка |
| Клиент | Калькулятор | Возможность посчитать стоимость грузоперевозки в разделе главной страницы |
| Клиент | Новости | Клиент может посмотреть актуальную информацию компании на главной странице |
| Администратор | Создать уникального пользователя-сотрудника | Администратор обладает возможностью создания аккаунта сотрудника для дальнейшего внедрения в систему |
| Администратор | Калькулятор | Редактирование коэффициентов прайса и редактирование доступных городов. |
| Администратор | Новости | Редактирование, добавление и удаление новостей |
| Администратор | Документы | Добавление и удаление документов компании из соответствующего раздела |
| Клиент | Регистрация | Клиент может зарегистрироваться в специальном разделе или во время оформления заявки |

Исходя из выявленных вариантов использования для акторов на рисунке 1 была представлена диаграмма вариантов использования.

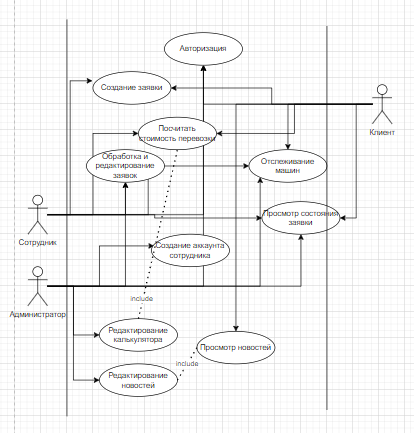


Рисунок 1 – Диаграмма вариантов использования

В таблице 4 показан полный шаблон варианта использования «Оставить заявку».

Таблица 4 - Описание варианта использования «Оставить заявку»

|  |  |
| --- | --- |
| Идентификатор и название: | Оставить заявку |
| Автор: Рустем | Дата создание 30.09.2023 |
| Основное действующее лицо: Клиент | Дополнительное действующее лицо: - |
| Описание: | Клиент в модуле главной страницы может посчитать стоимость и отправить заявку на реализацию доставки. |
| Триггер: | Клиент вводит габариты груза, тип и пункт доставки, чтобы оставить заявку. |
| Предварительные условия: | PRE-1. Клиент знает город доставки.  PRE-2. Клиент знает габариты и тип груза.  PRE-3. База данных по оценкам студентов доступна.  PRE-4. Открыт список доступных городов. |
| Выходные условия: | POST-1. Запрос сохраняется в системе.  POST-2. Сотрудник получает уведомления о созданной заявки. |
| Нормальное направление развития использования: | 1.0 Оставить заявку  1) Открытие главной страницы  2) Поиск и выбор точку отправки  3) Поиск и выбор точку доставки  4) Ввести габаритов груза  5) Узнать стоимость перевозки  6) Оставить контактные данные |
| Альтернативное направление развития варианта использования: | 1.1 Оставить заявку  1) Открытие главной страницы  2) Поиск и выбор отправной точки  3) Если указанной точки нет, оставить пожелание |
| Исключения: | 1.0.E1 Выставление некорректных данных  1) При вводе некорректного количества объема груза «Было введено некорректное значение»;  2) Система предлагает пользователю ввести иные габариты. |
| Частота использования: | Примерно 2 раза в неделю постоянным пользователем. |

Рассмотрим пользовательскую историю на основе технологии BDD для процедуры “Изменения коэффициента прайса калькулятора”.

**История:** ИЗМЕНЕНИЯ КОЭФФИЦИЕНТА ПРАЙСА КАЛЬКУЛЯТОРА

**Как** АДМИНИСТРАТОР

**Я хочу** иметь информацию о текущих коэффициентах

**Чтобы** я мог в будущем изменять коэффициенты прайса на определенные маршруты.

**Сценарий 1:** Указан существующий маршрут.

*Дано* выбран маршрут, для которого формируются данные.

*И* открыто окно для редактирования.

*Когда* администратор изменит данные и нажимает кнопку “Сохранить”.

*Тогда* данные сохраняются.

*И* закрывается окно редактирования.

**Сценарий 2:** Указан несуществующий маршрут.

*Дано* открывается окно для создания маршрута.

*Когда* администратор введет данные и нажимает кнопку “Сохранить”.

*Тогда* создается новый маршрут.

*И* закрывается окно создания маршрута.