

Министерство образования и науки Российской Федерации  
**Муромский институт (филиал)**  
федерального государственного бюджетного образовательного учреждения высшего образования  
**«Владимирский государственный университет**  
**имени Александра Григорьевича и Николая Григорьевича Столетовых»**  
(МИ ВлГУ)

Факультет ИТР  
Кафедра ПИн

УТВЕРЖДАЮ  
Зав. кафедрой

Кульков Я.Ю.  
(подпись)  
«  » 2025г

# БАКАЛАВРСКАЯ РАБОТА

Тема Автоматизация процессов доставки грузов через разработку  
информационной системы управления заказами  
МИВУ.09.03.04 - 5.000 ВКР

Руководитель

Кульков Я.Ю.

(фамилия, инициалы)

(подпись)

(дата)

Студент

ПИнз –120

(группа)

Чернышев А.Е.

(фамилия, инициалы)

(подпись)

(дата)

Муром 2025

Данная работа посвящена разработке информационной системы для автоматизации логистических процессов службы доставки. Система реализована как клиент-серверное приложение с клиентской частью на React и серверной частью на Ruby on Rails. Основные функциональные модули включают управление заказами, складской учет с использованием QR-кодов и маршрутизацию доставок. Особое внимание удалено безопасности данных, надежности хранения информации и удобству пользовательского интерфейса. В ходе работы проведены анализ требований, проектирование архитектуры, реализация и тестирование системы. Результатом является готовое к внедрению решение, соответствующее современным стандартам автоматизации логистических операций.

This work presents the development of an information system for automating delivery service logistics processes. The system is implemented as a client-server application with a React-based client and Ruby on Rails server. Core functional modules encompass order management, warehouse operations with QR-code verification, and delivery routing. Particular attention was given to data security, storage reliability, and user interface convenience. The project involved requirements analysis, architectural design, implementation, and system testing. The outcome is a production-ready solution that meets modern standards for logistics operations automation.

## Содержание

Введение .....	5
1 Анализ технического задания.....	7
1.1 Описание предметной области.....	7
1.2 План интервью для определения бизнес-требований .....	7
1.3 Формирование требований .....	8
1.4 Поиск и сравнение аналогов.....	9
1.5 Функциональные возможности.....	13
1.6 Обоснование выбора средств реализации.....	15
2 Проектирование архитектуры системы.....	18
2.1 Диаграмма вариантов использований .....	18
2.2 Диаграмма последовательностей .....	19
2.3 Логическая модель данных.....	20
2.4 Физическая модель данных .....	20
3 Реализация системы .....	22
3.1 Выбор и обоснование алгоритмов .....	22
3.2 Руководство программиста.....	26
3.3 Руководство пользователя .....	39
4 Тестирование системы .....	58
Заключение.....	62
Список литературы.....	63
Приложение А .....	64
Приложение Б .....	69
Приложение В .....	71

Изм.	Лист	№ докум.	Подпись	Дата
Разраб.	Чернышев А.Е.			
Провер.	Кульков Я.Ю.			
Реценз.				
Н. Контр.	Холкина Н.Е.			
Утврд.	Кульков Я.Ю.			

МИВЧ.09.03.04-5.000 ПЗ

Автоматизация процессов доставки грузов через разработку информационной системы управления заказами.  
Пояснительная записка

Лист.	Лист	Листов
	4	71
МИ ВлГУ ПИНз-120		

## **Введение**

Бурное развитие информационных технологий в последние десятилетия привело к необходимости поиска новых решений для автоматизации деятельности различных организаций, предприятий и служб. Рабочие процессы таких организаций часто связаны с обработкой и хранением больших объемов информации. Особенно это актуально для служб, занимающихся учетом данных о клиентах, товарах, доставках и других аспектах бизнеса.

Еще совсем недавно информация подобного рода хранилась в картотеках с использованием бумажных носителей. Этот процесс требовал значительных физических ресурсов, данных заносились вручную на карточки, что не только занимало много времени, но и создавало большие неудобства, повышая вероятность ошибок. Вся информация должна была быть найдена вручную, а сама картотека со временем изнашивалась, что снижало ее надежность.

Сегодня, в эпоху компьютерных технологий и автоматизации, физические картотеки и бумажные носители постепенно уступают место более современным и удобным решениям. Вместо сложных и неудобных процессов теперь используются мощные компьютерные системы, обеспечивающие быструю и надежную обработку данных. Однако, несмотря на значительный прогресс, по-прежнему существуют проблемы, такие как необходимость быстрого поиска нужной информации, обеспечение надежности хранения данных и соблюдение конфиденциальности.

Для решения этих задач используются специализированные программные продукты, которые часто объединены в крупные информационно-справочные системы. Эти системы предназначены для управления различными массивами данных, включая базы данных, и предоставляют удобный интерфейс для пользователей. Современные информационные системы включают в себя функции для добавления, редактирования, удаления данных, а также для их поиска и простого просмотра. Важным аспектом таких систем является обеспечение надежности хранения информации и предотвращение ее утрат.

Изм	Лист	№ докум	Подпись	Дата
-----	------	---------	---------	------

Реализация этих решений зависит от возможностей используемых технологий, поставленных задач и квалификации разработчика.

Разработка комплексного программного решения для автоматизации логистических процессов службы доставки грузов, включающего:

а) Кроссплатформенную систему:

- 1) реализация на React;
- 2) поддержка четырех ролевых моделей (клиент/доставщик/менеджер/администратор);
- 3) интеграция с QR-код технологиями;

б) Серверную часть:

- 1) RESTful API для централизованного управления данными;
- 2) безопасные HTTP-коммуникации;
- 3) масштабируемая архитектура;

в) Ключевые функциональные модули:

- 1) система управления доставками (маршрутизация, трекинг);
- 2) складской учет с технологией QR-верификации;

г) особое внимание уделяется:

- 1) обеспечению надежности хранения данных;
- 2) реализации удобного пользовательского интерфейса;
- 3) соблюдению требований информационной безопасности;
- 4) оптимизации бизнес-процессов доставки.

Работа представляет собой законченное решение, сочетающее современные подходы к разработке мобильных приложений с эффективной backend-архитектурой, соответствующее актуальным требованиям цифровизации логистических операций.

Изм	Лист	№ докум	Подпись	Дата

## **1 Анализ технического задания**

### **1.1 Описание предметной области**

Предметная область проекта связана с разработкой веб-приложения для автоматизации логистических процессов компании, занимающейся доставкой грузов. Основные задачи системы включают:

- управление заказами (создание, отслеживание, отмена);
- приём и выдача грузов на складах (ПВЗ);
- контроль за доставками и маршрутами;
- управление пользователями (регистрация, авторизация, роли);
- формирование цены и обработка оплат.

Система должна обеспечивать функциональность для различных классов пользователей:

- клиенты: оформление заказов, отслеживание статусов, просмотр истории;
- менеджеры склада: приём и выдача грузов, управление оплатами;
- доставщики: приём грузов в доставку, указание параметров транспорта;
- администраторы: управление ролями, настройка системы, аналитика.

### **1.2 План интервью для определения бизнес-требований**

Для уточнения требований к системе необходимо провести интервью с заинтересованными лицами. Основные вопросы:

- 1) Какие основные задачи должна решать система?
- 2) Кто будет основными пользователями (клиенты, менеджеры, доставщики, администраторы)?
- 3) Какие функции являются приоритетными для каждого класса пользователей?
- 4) Какие данные необходимо учитывать при управлении заказами, складами и доставками?

Изм	Лист	№ докум	Подпись	Дата

- 5) Какие ограничения по времени и ресурсам существуют для выполнения заказов?
- 6) Какие требования к безопасности и авторизации пользователей?
- 7) Какие интеграции с другими системами необходимы (CRM, ERP, платежные системы)?
- 8) Какие отчеты и аналитика должны быть доступны в системе?
- 9) Какие требования к производительности и доступности системы?
- 10) Какие бизнес-цели должны быть достигнуты с помощью системы?

### 1.3 Формирование требований

При проектировании системы автоматизации логистических процессов ключевое значение имеет четкое определение функциональных и нефункциональных характеристик, которые обеспечат эффективную работу всех участников цепочки доставки. Исходя из анализа бизнес-процессов компании и потребностей различных категорий пользователей, система должна реализовывать комплексный подход к управлению заказами, складами и доставками, обеспечивая при этом высокий уровень безопасности, удобства взаимодействия и масштабируемости. Основные требования к системе можно структурировать следующим образом:

#### 1.3.1 Функциональные требования

##### а) Управление пользователями и доступом:

- 1) многоуровневая система регистрации с дифференциацией по типам пользователей (клиенты, курьеры, менеджеры складов, администраторы);
- 2) механизм назначения и динамического изменения ролей с детализированными правами доступа;
- 3) возможность делегирования полномочий между сотрудниками;

##### б) Работа с заказами:

Изм	Лист	№ докум	Подпись	Дата

- 1) полноценный жизненный цикл обработки заказов от создания до финального исполнения;
  - 2) интеллектуальная система отмены заказов с учетом временных ограничений и бизнес-правил;
  - 3) архив выполненных заказов;
- в) Складская логистика:
- 1) учет операций приема-передачи грузов;
  - 2) интеграция с системами маркировки (QR-коды);
- г) Организация доставки:
- 1) учет транспортных характеристик;
  - 2) мониторинг выполнения доставок в реальном времени.

### **1.3.2 Нефункциональные требования**

- а) Безопасность и надежность:
- 1) ролевая модель доступа с детализированными правами;
- б) Пользовательский опыт:
- 1) адаптивный интерфейс для разных устройств;
  - 2) персонализированные рабочие пространства;
- в) Интеграционные возможности:
- 1) API для подключения внешних сервисов;
  - 2) поддержка стандартных протоколов обмена данными;
- г) Масштабируемость:
- 1) возможность наращивания функционала;
  - 2) поддержка роста количества пользователей;
  - 3) горизонтальное масштабирование системы.

### **1.4 Поиск и сравнение аналогов**

Для проведения всестороннего анализа рыночных решений и выявления оптимальных подходов к реализации системы рекомендуется выполнить

Изм	Лист	№ докум	Подпись	Дата

детальное исследование существующих платформ, специализирующихся на логистике и управлении доставками. В рамках сравнительного анализа целесообразно рассмотреть следующие популярные отраслевые решения:

а) 4Logist.com - Комплексная платформа для управления логистикой, предлагающая:

- 1) интегрированные инструменты управления складскими операциями;
- 2) систему маршрутизации и планирования доставок;
- 3) мобильные приложения для курьеров и клиентов;
- 4) аналитические инструменты для оптимизации логистических процессов;

д) Битрикс24 (логистический модуль) - Корпоративное решение, включающее:

- 5) CRM-систему с интеграцией логистических функций;
- 6) инструменты управления транспортными средствами;
- 7) систему учета грузоперевозок;
- 8) возможности автоматизации документооборота;

е) TMS (Transportation Management System) системы:

- 9) Oracle Transportation Management;

ж) Специализированные SaaS-решения:

- 10) LogistiX.

При проведении анализа следует учитывать:

- соответствие функциональных возможностей требованиям бизнеса;
- гибкость и масштабируемость архитектуры;
- уровень интеграции с существующей ИТ-инфраструктурой;
- стоимость владения и масштабирования;
- опыт внедрения в аналогичных бизнес-моделях;
- качество технической поддержки и экосистемы разработчиков.

Такой комплексный подход к анализу позволит не только выбрать оптимальные технологические решения, но и избежать типичных ошибок при проектировании системы.

Изм	Лист	№ докум	Подпись	Дата
-----	------	---------	---------	------

### 1.4.1 4Logist.com

Специализированная логистическая платформа для среднего и крупного бизнеса. Предлагает комплексное решение для управления цепочками поставок с акцентом на маршрутизацию и контроль доставок. Отличается глубокой отраслевой специализацией и развитыми аналитическими инструментами. Пример рабочего окна платформы представлен на рисунке 1.

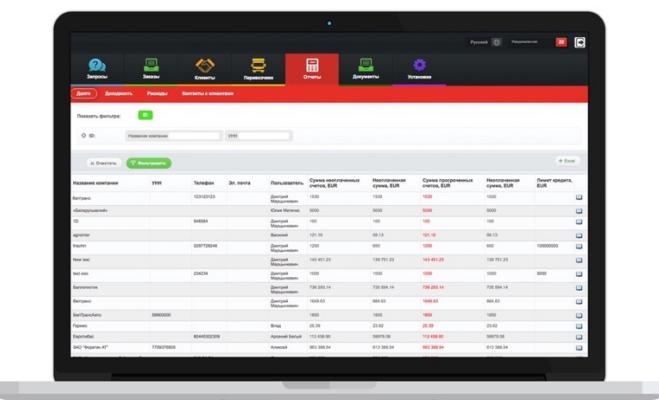


Рисунок 1 – Рабочее окно платформы 4Logist.

### 1.4.2 Битрикс24 (Логистический модуль)

CRM-система с расширенными возможностями для управления заказами и базовыми логистическими функциями. Оптимальна для компаний, где логистика тесно интегрирована с продажами и клиентским сервисом. Ограничена в специализированных функциях для сложной логистики. Рабочее окно модуля представлено на рисунке 2.

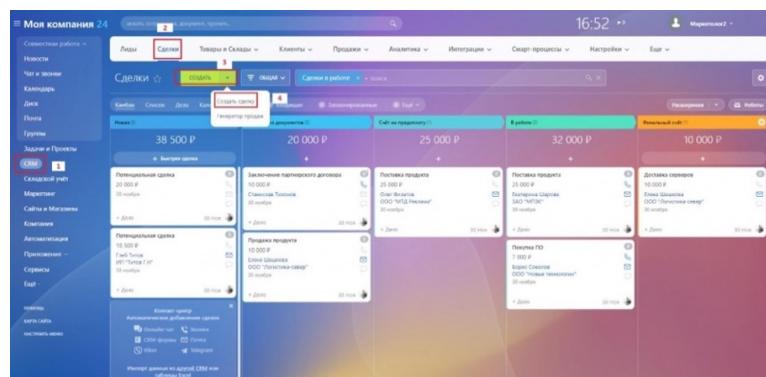


Рисунок 2 – Процесс доставки в Битрикс24.

Изм	Лист	№ докум	Подпись	Дата

МИВЧ.09.03.04-5.000 ПЗ

Лист

11

### 1.4.3 Oracle Transportation Management (OTM)

Корпоративная система управления транспортом класса Enterprise. Обеспечивает глобальное управление цепочками поставок с мощными алгоритмами оптимизации. Требует значительных ресурсов для внедрения и поддержки. Организация рабочего процесса в OTM представлена на рисунке 3.

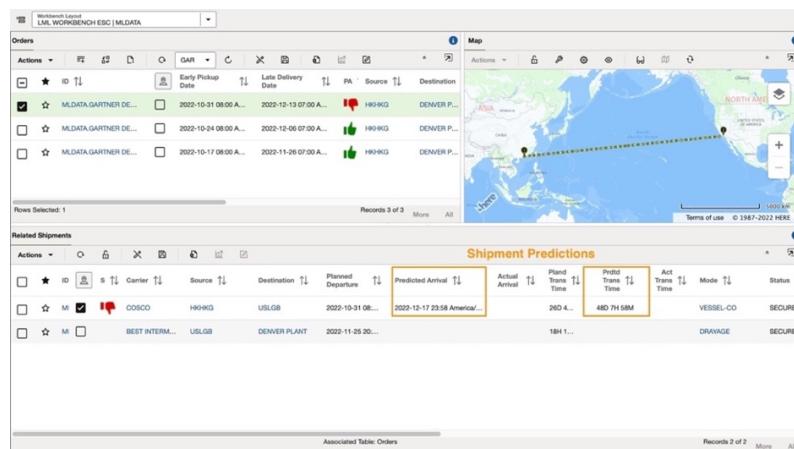


Рисунок 3 – Рабочее окно приложения Oracle Transportation Management.

### 1.4.4 LogistiX

Облачное SaaS-решение для оперативного управления доставками. Отличается быстрым развертыванием и гибкой подпиской. Подходит для малого и среднего бизнеса с типовыми логистическими процессами. Рабочее окно приложения представлено на рисунке 4.

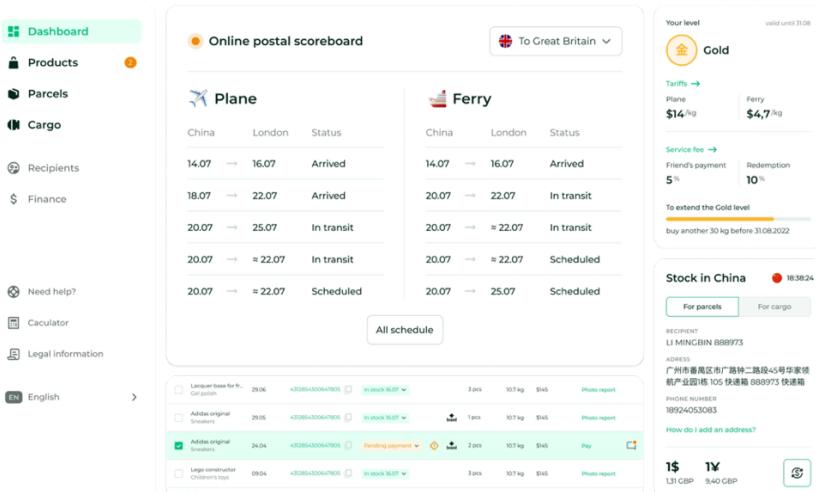


Рисунок 4 – Рабочее окно приложения LogistiX.

Изм	Лист	№ документ	Подпись	Дата

МИВЧ.09.03.04-5.000 ПЗ

Лист

12

## 1.4.5 Сравнение аналогов

Было проведено сравнение характеристик рассмотренных аналогов, которое отображено в таблице 1.

Таблица 1 – Сравнение аналогов разрабатываемой системы

Параметр сравнения	4Laogist.com	Битрикс24 (Логистика)	Oracle TMS	LogistiX
Основные функции	Управление заказами, маршрутизация, складской учет	Управление заказами, CRM, документооборот	Планирование перевозок, оптимизация маршрутов, фрахт	Управление доставками в реальном времени
Интеграции	API для 1С, ERP, маркетплейсов	Встроенная интеграция с продуктами 1С	Интеграция с SAP, Oracle ERP	REST API, Webhooks
Тип решения	Специализированная логистическая платформа	CRM с логистическим модулем	Корпоративная TMS-система	Облачное SaaS-решение
Мобильные приложения	Да (отдельные для клиентов и курьеров)	Только общее мобильное приложение	Нет (только веб-интерфейс)	Да (кроссплатформенное)
Ценовая модель	Подписка (от \$50/пользователь)	Помесячная оплата (от \$20/польз.)	Корпоративные лицензии	Pay-as-you-go
Плюсы	Глубокая специализация на логистике	Удобство интеграции с CRM	Мощные алгоритмы оптимизации	Гибкость и быстрое развертывание
Минусы	Высокий порог входа	Ограниченнная логистическая функциональность	Сложность внедрения	Ограниченная кастомизация
Поддержка	24/7 с SLA	Бизнес-часы	Премиум-поддержка	Чат/email поддержка
Масштабируемость	До 1000+ пользователей	До 500 пользователей	Неограниченная	До 300 пользователей

## 1.5 Функциональные возможности

### 1.5.1 Управление пользователями и доступом

Регистрация и роли пользователей:

Изм	Лист	№ докум	Подпись	Дата	Лист	13
					МИВЧ.09.03.04-5.000 ПЗ	

- а) Многоуровневая регистрация с верификацией:
  - 1) клиенты (email);
  - 2) сотрудники (корпоративная авторизация);
- 3) Расширенные профили с:
  - 3) предпочтениями доставки (клиенты);
  - 4) квалификациями и допусками (сотрудники);
- и) Авторизация и безопасность:
- к) Ролевая модель:
- л) Иерархия ролей (4 уровней доступа):
  - 5) клиент;
  - 6) курьер;
  - 7) менеджер склада;
  - 8) администратор.

### **1.5.2 Управление заказами**

Создание заказа:

- а) Мультиплатформенное оформление:
  - 1) веб-интерфейс;
  - 2) API для интеграций.

### **1.5.3 Складские операции**

Прием грузов:

- а) Автоматизированная сверка:
  - 1) по штрих-кодам/QR;
- м) Контроль качества:
  - 2) весовой контроль.
  - 3)

### **1.5.4 Идентификация грузов**

Для идентификации грузов используются QR-коды, которые выпускает система. QR-коды содержат информацию по грузу и заказу.

Изм	Лист	№ докум	Подпись	Дата

## **1.5.6 Жизненный цикл заказа**

Каждый заказ в системе проходит четко структурированный путь, обеспечивающий прозрачность и контроль на всех этапах. Разработан оптимизированный workflow, который автоматизирует ключевые процессы, минимизирует ручные операции и предоставляет полную видимость статусов для всех участников процесса. Отображение жизненного цикла представлено на рисунке А.6.

## **1.6 Обоснование выбора средств реализации**

Для разработки системы выбраны следующие технологии:

- Backend: Ruby on Rails. Выбор обусловлен высокой производительностью, удобством разработки и поддержкой MVC-архитектуры;
- Frontend: React. Позволяет создавать динамичные и интерактивные пользовательские интерфейсы;
- База данных: PostgreSQL. Надежная СУБД с поддержкой сложных запросов и транзакций;
- Деплой: Docker + AWS/Heroku. Обеспечивает масштабируемость и удобство развертывания.

Использование данного стека технологий позволит создать надежное, производительное и легко поддерживаемое решение.

### **1.6.1 Серверная часть. Ruby on Rails**

Выбор пал на Ruby on Rails как оптимальный фреймворк для backend-разработки благодаря следующим ключевым преимуществам:

- a) Архитектурные преимущества:
  - 1) полноценная поддержка MVC-паттерна;
  - 2) Convention over Configuration принцип;
  - 3) встроенные механизмы RESTful API;
  - 4) Active Record ORM для работы с БД;

Изм	Лист	№ докум	Подпись	Дата

н) Производительность и надежность:

- 5) JIT-компиляция в Ruby 3.x;
- 6) поддержка многопоточности;
- 7) встроенные механизмы кэширования;
- 8) безопасность по умолчанию (защита от OWASP Top 10);

о) Экосистема для логистики:

- 9) Готовые решения для:
  - i) маршрутизации (Routing Engine);
  - ii) работы с временными интервалами;
- 10) Широкий выбор гем'ов для интеграций.

### **1.6.2 Клиентская часть. React + Mui/Joy**

Комбинация React с библиотеками MUI и Joy UI обеспечивает:

Эффективность разработки:

а) Готовые компоненты для:

- 1) интерактивных карт (Mapbox);
- 2) таблиц с виртуализацией (TanStack Table);
- 3) сложных форм (Formik + Yup);

п) ThemeProvider для быстрой смены тем.

Особенности реализации:

```
<ThemeProvider theme={logisticsTheme}>
  <JoyCssVarsProvider>
    <MapDashboard>
      <RouteOptimizer />
      <RealTimeTracking />
    </MapDashboard>
  </JoyCssVarsProvider>
</ThemeProvider>
```

Преимущества Joy UI:

а) Специализированные компоненты для:

- 1) визуализации маршрутов;
- 2) таблиц с большими объемами данных.

Изм	Лист	№ докум	Подпись	Дата

### **1.6.3 База данных PostgreSQL**

База данных: PostgreSQL

Выбор PostgreSQL обусловлен требованиями к логистической системе:

Ключевые возможности:

- поддержка геопространственных данных (PostGIS);
- аконные функции для аналитики;
- полнотекстовый поиск;
- JSONB для гибких схем данных.

Изм	Лист	№ докум	Подпись	Дата

МИВЧ.09.03.04-5.000 ПЗ

Лист

17

## **2 Проектирование архитектуры системы**

Чтобы снизить количество ошибок при проектировании структуры базы данных и бизнес-логики приложения, важно наглядно представить и систематизировать информацию. Для этого были разработаны:

- 1) Диаграмма вариантов использования (отображает взаимодействие пользователей с системой);
- 2) Диаграмма состояний (визуализирует состояния системы и точки перехода);
- 3) Логическая модель данных (описывает сущности и их взаимосвязи без привязки к СУБД);
- 4) Физическая модель данных (определяет конкретную реализацию базы данных с учетом технических ограничений).

Это позволяет четко спланировать архитектуру системы и избежать недочетов на этапе разработки.

### **2.1 Диаграмма вариантов использований**

Эта диаграмма была разработана для наглядного представления функциональных требований к системе. С её помощью удалось:

- определить ключевые сценарии взаимодействия пользователей с системой;
- выделить основные функции, которые система должна предоставлять;
- проанализировать, как эти функции удовлетворяют потребности пользователей;
- позволяет определить роли пользователей.

Таким образом, диаграмма вариантов использования стала важным инструментом для проектирования логики приложения и уточнения его функционала.

Изм	Лист	№ докум	Подпись	Дата

Диаграмма вариантов использования представлена на рисунке 6.

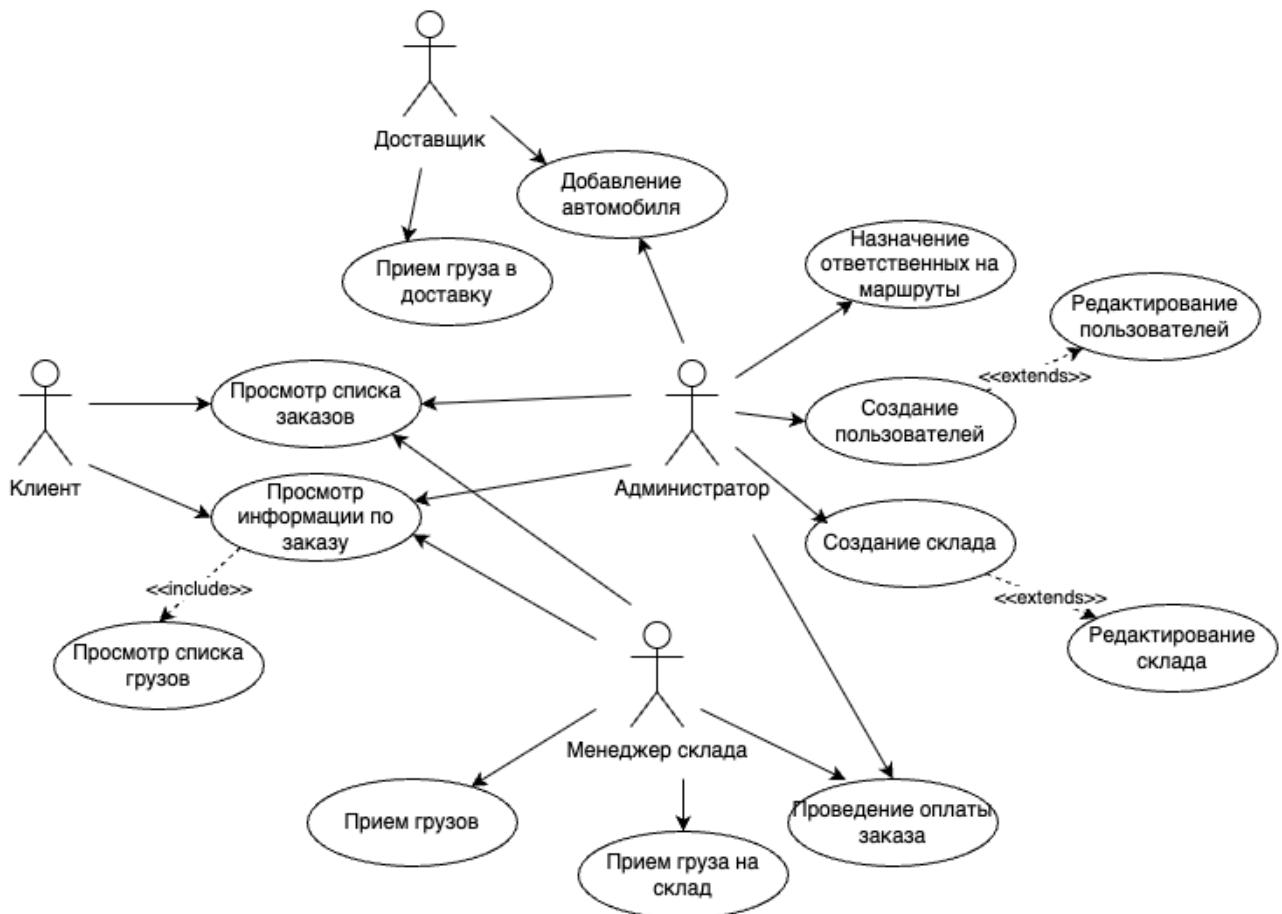


Рисунок 6 – Диаграмма вариантов использования.

## 2.2 Диаграмма последовательностей

Диаграмма последовательности действий была использована для наглядного представления взаимодействия между модулями системы и её участниками в хронологическом порядке. Такой подход необходим для получения целостного представления о поведении системы во времени. Диаграмма позволила подробно отразить порядок вызовов, передаваемые параметры и характер взаимодействия компонентов, что способствовало раннему выявлению и устранению потенциальных проблем в архитектуре.

Этот тип диаграммы помогает обнаружить скрытые зависимости и возможные узкие места, которые могут негативно повлиять на производительность или привести к сбоям в работе системы. Проведённый анализ

Изм	Лист	№ докум	Подпись	Дата
-----	------	---------	---------	------

на основе диаграммы последовательности повысил качество проектных решений и позволил значительно сократить время, затрачиваемое на выявление и исправление ошибок в процессе тестирования и внедрения.

Данная диаграмма представлена в приложение А (рисунок 1-5).

### **2.3 Логическая модель данных**

Логическая модель данных представляет собой абстрактное описание структуры информации в системе, включающее сущности, их атрибуты и взаимосвязи между ними, без учета технических аспектов хранения. Она играет ключевую роль в проектировании информационных систем, позволяя формализовать и организовать данные, необходимые для работы системы. Благодаря ей обеспечивается целостность, согласованность и корректность данных, что существенно упрощает последующую разработку физической модели, уже учитывающей конкретные технологические и аппаратные условия хранения.

Кроме того, логическая модель данных способствует упрощению и ускорению процесса проектирования, повышая качество и надежность итоговой системы. Она позволяет выявить и устраниить потенциальные ошибки на ранних этапах разработки, улучшить согласованность данных и создать эффективную архитектуру для обработки и хранения больших объемов информации.

Логическая модель представлена на рисунке Б.1.

### **2.4 Физическая модель данных**

Физическая модель данных была разработана для конкретизации структуры данных с учетом особенностей реализации в СУБД PostgreSQL. При построении данной модели было определено, как именно данные будут физически храниться, индексироваться и управляться в базе данных. Это включало описание таблиц,

Изм	Лист	№ докум	Подпись	Дата

столбцов, типов данных, индексов, первичных и внешних ключей, а также взаимосвязей между таблицами.

Разработка физической модели данных позволила повысить производительность системы, обеспечить эффективное хранение информации и быстрый доступ к ней. Модель также послужила основой для создания и настройки объектов базы данных в рамках проектируемой системы, что значительно упростило процесс интеграции с существующей инфраструктурой.

Процесс создания модели включал анализ и выбор оптимальных структур хранения с учетом специфики PostgreSQL, а также использование профильной литературы. Определение ключей и связей между таблицами обеспечило соблюдение целостности и согласованности данных в системе.

Физическая модель данных представлена на рисунке Б.2.

Изм	Лист	№ докум	Подпись	Дата

### **3 Реализация системы**

#### **3.1 Выбор и обоснование алгоритмов**

Разрабатываемое приложение в соответствии с техническим заданием представляет из себя клиент-серверное приложение. Для серверной части приложения использован Ruby и фреймворк Ruby on Rails, в качестве базы данных PostgreSQL, Redis в качестве кэш-хранилища. Для клиентской части приложения использовался фреймворк React.

##### **3.1.1 Ruby on Rails (RoR)**

Ruby — это интерпретируемый объектно-ориентированный язык программирования, известный своей лаконичностью и читаемостью. Его основные преимущества:

- простота и выразительность — код легко читается и пишется;
- объектно-ориентированность — всё является объектом, включая примитивы;
- большое сообщество и множество библиотек (гемов);
- поддержка метaprogramмирования — позволяет писать гибкий и расширяемый код.

Ruby on Rails — это популярный веб-фреймворк, написанный на Ruby, следящий принципу "Convention over Configuration" (соглашения важнее конфигураций) и "Don't Repeat Yourself" (не повторяйся). Для изучения особенностей работы с Ruby on Rails использовалась литература [1-3].

Основные возможности RoR:

- MVC-архитектура (Model–View–Controller) — разделение логики приложения;
- автоматическая генерация кода (генераторы scaffolding);
- интеграция с базой данных через Active Record;
- маршрутизация, обработка запросов, RESTful API;

Изм	Лист	№ докум	Подпись	Дата

- система миграций для управления схемой БД.
- В Rails используется ORM Active Record:
  - модели соответствуют таблицам БД;
  - позволяет выполнять SQL-запросы с помощью Ruby-методов (`User.find(1)`, `Post.where(published: true)`);
  - миграции позволяют версионировать структуру базы данных.
  - Rails хорошо работает с PostgreSQL, включая поддержку JSONB, индексов, транзакций и расширений.
  - Для выполнения долгих или периодических задач используются фоновые воркеры. Распространённые библиотеки:
    - Sidekiq — асинхронная очередь заданий, использующая Redis;
    - позволяет выполнять задачи в фоне (например, отправку писем, обработку данных);
    - интегрируется с Rails и Active Job.

### 3.1.2 PostgreSQL

PostgreSQL — это объектно-реляционная система управления базами данных с открытым исходным кодом, которая отличается высокой надёжностью, масштабируемостью и соответствием стандартам SQL. Для изучения особенностей работы с PostgreSQL использовалась литература [4].

Преимущества PostgreSQL:

- надёжность и стабильность — используется во многих критически важных системах;
- расширяемость — поддержка пользовательских типов данных, функций, индексов и расширений (например, PostGIS для геоданных);
- поддержка ACID — транзакции обеспечивают согласованность и целостность данных;
- мощный язык запросов SQL — включая оконные функции, CTE (with-запросы), подзапросы;

Изм	Лист	№ докум	Подпись	Дата

- поддержка JSON и JSONB — позволяет использовать PostgreSQL как гибридную реляционно-документную СУБД;
- индексация — поддержка различных типов индексов: B-tree, GIN, GiST, BRIN и др.;
- репликация и масштабируемость — встроенная поддержка потоковой репликации, логической репликации, шардирования через сторонние решения.

Использование в проекте:

В разработанном приложении PostgreSQL используется как основная реляционная база данных для хранения информации о пользователях, записях и связанной бизнес-логике. Благодаря хорошей интеграции с Ruby on Rails через ORM Active Record, осуществляется удобное взаимодействие с базой данных на уровне моделей.

Также применяются миграции, которые позволяют версионировать схему базы данных, а транзакции и индексы обеспечивают надёжность хранения и высокую производительность при больших объёмах данных.

### 3.1.3 Redis

Redis (REmote DIctionary Server) — это высокопроизводительное хранилище структур данных в памяти с открытым исходным кодом. Оно поддерживает различные структуры данных, такие как строки, хэши, списки, множества, отсортированные множества и другие.

Преимущества Redis:

- скорость — хранение данных в оперативной памяти обеспечивает время доступа в пределах миллисекунд и ниже;
- поддержка различных структур данных — позволяет реализовать кэширование, очереди, сессии и счётчики;
- простота использования — удобный синтаксис команд, возможность работы напрямую с ключами и значениями;
- поддержка Pub/Sub — встроенная реализация публикации/подписки для реализации real-time механизмов;

Изм	Лист	№ докум	Подпись	Дата

- персистентность (по желанию) — поддержка снапшотов (RDB) и логирования изменений (AOF);
- масштабируемость — возможность горизонтального масштабирования с помощью Redis Cluster;
- интеграция с фреймворками — легко подключается к Ruby on Rails и Sidekiq для работы с фоновыми задачами.

В данном приложении Redis используется в следующих целях:

- Очереди задач — в связке с Sidekiq Redis используется как хранилище фоновых задач (например, отправка писем, обработка уведомлений).

Использование Redis позволяет повысить производительность и масштабируемость приложения, особенно при высоких нагрузках и большом количестве параллельных запросов.

### 3.1.4 React

React — это популярная JavaScript-библиотека для создания пользовательских интерфейсов. Она используется при разработке одностраничных приложений (SPA) и позволяет эффективно обновлять и отображать данные без перезагрузки страницы. React широко применяется в коммерческой разработке благодаря своей гибкости, масштабируемости и большому сообществу. Для изучения особенностей работы с React использована литература [5-7].

Почему выбран React:

- компонентная архитектура позволяет разбивать интерфейс на независимые модули, что облегчает разработку, тестирование и повторное использование компонентов;
- интерактивность и производительность React использует виртуальный DOM, что делает обновления интерфейса быстрыми и плавными — критично для динамичных приложений, таких как логистические системы;

Изм	Лист	№ докум	Подпись	Дата

- поддержка современных UI-библиотек Совместимость с MUI и Joy UI позволяет быстро создавать адаптивный, функциональный интерфейс с готовыми компонентами (таблицы, карты, формы);
- широкая экосистема и поддержка React активно развивается, имеет множество сторонних библиотек для работы с формами, состоянием, API и др., что ускоряет разработку и повышает надёжность решения.

Таким образом, React обеспечивает оптимальный баланс между удобством разработки, высокой производительностью и качеством пользовательского опыта, что делает его логичным выбором для реализации клиентской части логистической системы.

## **3.2 Руководство программиста**

Данный раздел предназначен для разработчиков, которые будут сопровождать, развивать или интегрировать представленную информационную систему. Он содержит описание архитектуры клиентской и серверной частей, рекомендации по развёртыванию, настройке и запуску проекта, а также пояснения к основным модулям и технологиям, использованным при разработке.

Руководство поможет быстрее освоить структуру проекта, понять принципы взаимодействия компонентов и обеспечить бесперебойную поддержку системы в дальнейшем. Особое внимание уделено стандартам кодирования, организации каталогов, использованию сторонних библиотек и механизмам расширения функционала.

### **3.2.1 Структура Ruby on Rails приложения**

Ruby on Rails приложение (далее RoR-приложение) имеет устоявшуюся базовую структуру компоновки приложения.

Основные компоненты системы представлены в таблице 2.

Изм	Лист	№ докум	Подпись	Дата

Таблица 2 – Основные компоненты Ruby on Rails

Папка / Файл	Описание
Ядро приложения (app/)	
controllers/	Контроллеры для обработки HTTP-запросов (например, DeliveriesController)
models/	Бизнес-логика и ORM-модели (например, Delivery, Warehouse)
views/	Шаблоны представления (ERB, HAML, SLIM)
services/	Сервисные классы для сложной бизнес-логики
policies/	Правила авторизации (Pundit)
jobs/	Фоновые задачи (Active Job)
Конфигурация (config/)	
routes.rb	Определение маршрутов REST API
database.yml	Настройки подключения к PostgreSQL
puma.rb	Конфигурация веб-сервера Puma
environments/	Окружения: development, production, test
initializers/	Инициализационные скрипты
Работа с данными (db/)	
migrate/	Миграции для изменения структуры базы данных
schema.rb	Текущее состояние схемы базы данных
seeds.rb	Начальное заполнение базы данных
Тестирование (spec/)	
models/	Модельные тесты
requests/	Интеграционные тесты
factories/	Фабрики тестовых данных (FactoryBot и др.)
Документация API (swagger/)	
swagger/	Общая документация API
schemas/	Схемы данных
v1/	Описание API endpoints версии 1
Корневые файлы	
Gemfile	Список Ruby-зависимостей
Gemfile.lock	Фиксированные версии гемов
Config.ru	Rack-конфигурация
Dockerfile	Конфигурация контейнера docker
docker-compose.yaml	Оркестрация сервисов
Rakefile	Пользовательские задачи

## Продолжение таблицы 2

Папка / Файл	Описание
Makefile	Автоматизация команд
redis.conf	Конфигурация Redis для кэширования

Характеристики структуры:

- модульность: Четкое разделение компонентов;
- масштабируемость: Готовность к росту функционала;
- поддержка API: Оптимизирована для RESTful сервисов;
- DevOps-готовность: Docker-конфигурации из коробки [9];

### 3.2.2 Использование AASM для Ruby on Rails

AASM (Acts As State Machine) — это гем (библиотека) для Ruby on Rails, который позволяет удобно реализовывать конечные автоматы состояний (state machines) в моделях ActiveRecord. Он помогает описывать бизнес-логику, связанную с переходами между различными состояниями объекта.

Преимущества AASM:

- 1) простая DSL-синтаксис для описания состояний и переходов;
- 2) явная структура состояний — легко отследить текущий статус объекта;
- 3) поддержка колбэков до/после переходов;
- 4) поддержка сохранения состояний в базе данных.

AASM может использоваться для управления статусами заказов, задач, заявок, платежей и других сущностей, состояние которых меняется во времени. Это упрощает контроль переходов и делает бизнес-логику более читаемой и предсказуемой.

Ниже представлен пример использования AASM для состояний заказа:

```
class Order < ActiveRecord::Base
  include AASM
  belongs_to :sender, class_name: 'User'
  belongs_to :receiver, class_name: 'User', optional: true
  belongs_to :start_warehouse, class_name: 'Warehouse'
  belongs_to :end_warehouse, class_name: 'Warehouse'
```

Изм	Лист	№ докум	Подпись	Дата

МИВЧ.09.03.04-5.000 ПЗ

Лист

28

```

has_many :cargos, dependent: :destroy

after_create :schedule_status_check

aasm column: 'status' do
  state :created, initial: true
  state :wait_payment
  state :paid
  state :in_delivery
  state :awaiting_pickup
  state :completed
  state :canceled
  state :return_in_process
  state :returned

  event :cargo_accepted do
    transitions from: :created, to: :wait_payment
  end

  event :pay do
    transitions from: :wait_payment, to: :paid
  end

  event :accept_for_delivery do
    transitions from: :paid, to: :in_delivery
  end

  event :stock_received do
    transitions from: :in_delivery, to: :awaiting_pickup
  end

  event :complete do
    transitions from: :awaiting_pickup, to: :completed
  end

  event :cancel, guards: [:not_paid?] do
    transitions from: %i[created], to: :canceled
  end
end

```

end

Изм	Лист	№ докум	Подпись	Дата

МИВЧ.09.03.04-5.000 ПЗ

Лист

29

Использование AASM в модели добавляет ей методы проверки возможности перехода по состояниям (`may_complete?`, `may_pay?`), а так же добавляет методы соответствующие именам событий для изменения статуса состояния.

```
unless @order.may_pay?  
  render json: { errors: 'Payment not available' }, status:  
:unprocessable_entity  
  raise ActiveRecord::Rollback  
end
```

### 3.2.3 Авторизация и аутентификация в приложении.

В системе реализована безопасная аутентификация и ролевая авторизация на основе JWT (JSON Web Token). Это позволяет:

- аутентифицировать пользователей через механизм токенов, обеспечивая защищенный доступ к API;
- контролировать права доступа с помощью ролей (например, клиент, менеджер, администратор), хранящихся в модели User.

Ключевые компоненты:

- JWT-токены – используются для верификации пользователей после входа.
  - Ролевая модель – определяет доступные действия для каждого типа пользователя.
  - Шифрование паролей – с помощью bcrypt для безопасного хранения в БД.
- Аутентификация происходит при каждом запросе к защищенным endpoints, а авторизация проверяется через политики (Pundit). Примеры кода и настройки приведены ниже.

#### 3.2.3.1 Управление доступами с помощью pundit в Ruby on Rails

Гем pundit предоставляет простой и гибкий механизм для управления правами доступа (авторизации) на основе политик (policies). Позволяет разделять логику проверки прав доступа для разных ролей пользователей.

Изм	Лист	№ докум	Подпись	Дата

## Использование в проекте:

Создаются политики (например, OrderPolicy) для контроля доступа к действиям (создание, редактирование, удаление) в зависимости от роли пользователя (клиент, менеджер, администратор, курьер).

Пример реализации класса политик представлен ниже:

```
class OrderPolicy < ApplicationPolicy
  class Scope < Scope
    def resolve
      if user.high_rule?
        scope.all
      else
        scope.where('sender_id = ? OR receiver_id = ?', user.id, user.id)
      end
    end
    def index?
      show?
    end
    def show?
      return @record.sender == user || @record.receiver == user if user.low_rule?
      true
    end
    def create?
      return record.sender_id == user.id if user.low_rule?
    end
    def hand_over?
      cargo_accepted?
    end
    def destroy?
      user.admin?
    end
  end
end
```

Данный гем так же предоставляет реализацию для управления списками посредством переопределения метода resolve для внутреннего класса Scope. Используя данную реализацию можно легко определять условные выборки из базы данных.

Изм	Лист	№ докум	Подпись	Дата
-----	------	---------	---------	------

Использование политики для определения доступа к конкретному экземпляру ресурса осуществляется через вызов метода `authorize`, в методе контроллера, и передачи в качестве параметра экземпляра.

```
def payment
  authorize @order
  # todo some
end
```

Если переданный экземпляр не удовлетворяет параметрам то `pundit` выбрасывает исключение `Pundit::NotAuthorizedError`, которое имеет глобальный обработчик в `ApplicationController`.

```
rescue _ from Pundit::NotAuthorizedError, with: :user_not_authorized

def user_not_authorized(exception)
  render json: { errors: 'You are not authorized to perform this action',
    policy: exception.policy.class.to_s, # Название политики
    action: exception.query }, status: :forbidden # действие,
которое запрещено
end
```

### **3.2.4 Расширение запросов с помощью гемов `kaminary` и `ransack` для RoR**

В современных веб-приложениях работа с большими объемами данных требует эффективных инструментов для пагинации, поиска и фильтрации. В данном проекте эти задачи решаются с помощью двух мощных гемов:

1) `kaminari` – обеспечивает удобное разбиение данных на страницы с поддержкой метаданных (общее количество записей, текущая страница и т. д.).

2) `ransack` – предоставляет гибкий механизм для построения сложных запросов, включая фильтрацию, сортировку и поиск по связанным моделям.

Их совместное использование позволяет:

1) Оптимизировать нагрузку на базу данных, загружая только нужные записи.

Изм	Лист	№ докум	Подпись	Дата

- 2) Реализовать удобные интерфейсы для пользователей (таблицы с пагинацией, фильтры в CRM).
- 3) Легко интегрировать функционал с API, передавая параметры запроса (page, per, q).

#### **3.2.4.1 Пагинирование запросов с помощью Kaminary в Ruby on Rails**

Гем kaminari предоставляет удобный и гибкий механизм для пагинации (разбивки на страницы) данных в Rails-приложениях. Он поддерживает работу с ActiveRecord и позволяет настраивать отображение элементов на странице, общее количество страниц и стилизацию пагинации.

Использование в проекте:

- 1) Применяется для разбивки больших списков (например, заказов, пользователей или товаров) на страницы с возможностью переключения между ними;
- 2) Интегрируется с фронтендом (React) через API, возвращая метаданные (общее количество страниц, текущая страница и т. д.).

#### **3.2.4.1 Поиск и фильтрация с помощью Ransack в Ruby on Rails**

Гем ransack предназначен для поиска и фильтрации данных в Rails-приложениях. Он позволяет строить сложные запросы к базе данных через простой синтаксис, включая сортировку, фильтрацию по условиям и полнотекстовый поиск.

Использование в проекте:

- 1) Реализует поиск по моделям (например, фильтрация заказов по дате, статусу или клиенту).
- 2) Поддерживает сортировку (например, по имени пользователя или дате создания).
- 3) Интегрируется с API, принимая параметры запроса (q) и возвращая отфильтрованные данные.

Изм	Лист	№ докум	Подпись	Дата

Для осуществления поиска к запросу добавляются query-параметры которые представляют из себя сочетание:

q[имя\_параметра\_правило\_поиска]=значение.

Ransack имеет богатый набор правил поиска:

- 1) Поиск по точному и частичному совпадению (\_eq, \_cont, \_match);
- 2) Поиск по большему и меньшему значению (\_lt, \_gt, \_lteq, \_gteq);
- 3) Поиск по присутствию значений (\_null, \_not\_null);
- 4) И т.д.

Для работы Ransack в модели нужно указать какие поля будут доступны для поиска.

```
def self.ransackable_attributes(_auth_object = nil)
  %w[created_at end_warehouse_id id price receiver_id sender_id
  start_warehouse_id status
  updated_at] + _ransackers.keys
end
```

Так же он поддерживает поиск по цепочки связей. Запрос для поиска по цепочки формируется следующим образом:

q [имя\_связи\_имя\_параметра\_параметр\_поиска]=значение.

Пример: /api/users?q[roles\_name\_eq]=courier.

```
def self.ransackable_associations(_auth_object = nil)
  authorizable_ransackable_associations + ['roles']
end
```

В случае если для решения задач не хватает стандартных Ransack-правил, гем предоставляет интерфейс для написания собственных правил. Собственные правила уже предоставляют полное управление запросами. Пример реализации собственного правила представлен ниже. Данное правило выполняет поиск по нескольким таблицам используя join для объединения и поиска складов по которым, по каким-либо причинам, не были созданы автоматически маршруты, либо по маршрутам не назначены ответственные.

Изм	Лист	№ докум	Подпись	Дата
-----	------	---------	---------	------

```

scope :with_unassigned_or_no_routes, lambda {
    joins('LEFT JOIN routes AS from_routes ON from_routes.start_warehouse_id =
warehouses.id')
    .joins('LEFT JOIN routes AS to_routes ON to_routes.end_warehouse_id =
warehouses.id')
    .where(
        'from_routes.id IS NULL OR from_routes.user_id IS NULL OR
cardinality(from_routes.delivery_days) = 0 OR ' \
        'to_routes.id IS NULL OR to_routes.user_id IS NULL OR
cardinality(to_routes.delivery_days) = 0'
    )
    .distinct
}
}

def self.ransackable_scopes(_auth_object = nil)
  %i[with_null_routes with_assigned_routes with_unassigned_or_no_routes]
end

```

### 3.2.5 Тестирование и документирование API RoR

В разработке надежного API критически важны два аспекта:

- Автоматизированное тестирование – для проверки корректности работы endpoints и бизнес-логики;
- Документирование – для удобства интеграции фронтенда и сторонних сервисов.

В проекте эти задачи решаются с помощью следующих инструментов:

- rspec-rails - главный фреймворк для тестирования Rails-приложений.

Позволяет писать юнит-тесты, интеграционные тесты и тесты запросов к API;

- Faker - это библиотека для генерации реалистичных фейковых данных (имена, email, адреса, даты и т. д.). Используется при тестировании, заполнении базы демо-данными и создании fixtures;

- Rswag - инструмент для автоматической генерации Swagger-документации на основе RSpec-тестов. Создает интерактивную документацию в формате OpenAPI [8].

### 3.2.6 Структура приложения React

Изм	Лист	№ докум	Подпись	Дата

МИВЧ.09.03.04-5.000 ПЗ

Лист

35

Проект организован по методологии Feature-Sliced Design (FSD), которая разделяет код по бизнес-логике и обеспечивает масштабируемость.

Основные слои (layers) представлены в таблице 3.

Таблица 3 – Основные слои FSD.

Имя каталога	Назначение
app	<ul style="list-style-type: none"><li>– инициализация приложения, глобальные настройки (роутинг, стор, стили);</li><li>– инициализация Redux, Router.</li></ul>
pages	<ul style="list-style-type: none"><li>– собирает фичи и виджеты в полноценные экраны;</li><li>– минимальная реализация бизнес-логики, только композиция компонентов.</li></ul>
widgets	<ul style="list-style-type: none"><li>– компоненты решающие комплекс бизнес-задач (например OrderListWithFilters)</li></ul>
features	<ul style="list-style-type: none"><li>– бизнес-фичи (например auth, list, notification);</li><li>– UI-логика.</li></ul>
entities	<ul style="list-style-type: none"><li>– бизнес сущности (например user, order);</li><li>– переиспользуемые модели и API-взаимодействия.</li></ul>
shared	<ul style="list-style-type: none"><li>– общие низкоуровневые компоненты и утилиты;</li><li>– UI-кит;</li><li>– хелперы (formatDate, fetch);</li><li>– константы, глобальные типы.</li></ul>

В корне проекта так же расположен ряд конфигурационных файлов, их список представлен в таблице 4.

Таблица 4 – Список корневых файлов React-приложения

Файл	Назначение
vite.config.ts	Конфигурация сборки (Vite).
docker-compose.yml	Настройка контейнеризации (Nginx, сервер).
Dockerfile	Конфигурация контейнера Docker
eslint.config.js	Правила линтера.
tsconfig.*.json	Настройки TypeScript для клиента/сервера.

Принципы FSD в проекте:

1) Слоистая архитектура:

a) app → pages → widgets → features → entities → shared;

2) Изоляция: Фичи/сущности не зависят от страниц;

3) Переиспользование: shared и entities используются во всех слоях.

### 3.2.7 Описание API сервера

API системы разделено на несколько ключевых групп маршрутов, обеспечивающих управление пользователями, заказами, складами, перевозками и другими сущностями. Все маршруты требуют аутентификации через JWT-токен (кроме публичных). Список основных маршрутов представлен в таблице 5.

Таблица 5 – Список основных маршрутов.

Раздел	Метод и маршрут	Описание
Аутентификация	POST /api/login	Вход в систему. Возвращает JWT-токен и данные пользователя
Пользователи (users)	GET /api/users	Список пользователей с фильтрацией по ролям (Ransack)
	GET /api/users/{id}	Информация о конкретном пользователе
	PUT /api/users/{id}	Обновление данных пользователя (ФИО, документы)
	POST /api/users/{id}/add_roles	Добавление ролей пользователю
Заказы (orders)	GET /api/orders	Список заказов с фильтрацией по статусу (Ransack)
	POST /api/users/{id}/remove_roles	Удаление ролей
	POST /api/orders	Создание заказа
	GET /api/orders/{id}	Детали заказа (со статусами, складами, отправителем/получателем)
	POST /api/orders/{id}/cargo_accepted	Подтверждение груза и расчет стоимости
	POST /api/orders/{id}/payment	Оплата заказа
Грузы (cargos)	GET /api/orders/{order_id}/cargos	Список грузов в заказе (с пагинацией)
	POST /api/orders/{order_id}/cargos	Добавление груза

## Продолжение таблицы 5

Раздел	Метод и маршрут	Описание
Грузы (cargos)	POST /orders/{order_id}/cargos/{id}/hand_over	Изменение статуса груза на 'выдан'
Перевозки (shippings)	GET /api/shippings	Список перевозок с фильтрацией по статусу
	GET /api/shippings/{id}	Детали перевозки (маршрут, водитель, статус)
	POST /api/shippings/{id}/start_load	Начало погрузки
	POST /api/shippings/{id}/start_delivery	Начало доставки
Склады (warehouses)	GET /api/warehouses	Список складов с фильтрацией по городу
	POST /api/warehouses	Создание склада
	POST /api/warehouses/{warehouse_id}/uploaded_cargo/{id}	Прием груза на склад
Автомобили пользователей (user cars)	GET /api/users/{user_id}/cars	Список автомобилей пользователя
	POST /api/users/{user_id}/cars	Добавление автомобиля
Города (cities)	GET /api/cities	Список городов (публичный, без аутентификации)

Общие особенности:

- фильтрация: Для списков (orders, users, shippings) используется Ransack (пример: q[status\_eq]=completed);
- безопасность: Все маршруты (кроме /api/login и /api/cities) требуют заголовка Authorization: Bearer <token>;
- ошибки: Стандартные HTTP-коды (403 — доступ запрещен, 422 — неверные данные).

Полная спецификация доступна в Swagger UI (/api-docs) [11].

### 3.2.8 Описание алгоритма оплаты и распределения грузов по доставкам

Проведение оплаты по заказ осуществляется менеджером вручную, так как по техническому заданию подключение платежного сервиса в систему на данном этапе заказчику не требуется. Оплата выполняется запросом к серверу POST /api/orders/{id}/payment. При выполнении данного запроса проверяются доступы пользователя на выполнение данной операции и возможность

проведения по оплаты по заказу. После чего запускается метод класса `CargoDistributor#distribute`.

Данный метод выполняет валидацию заказа по наличию в нем грузов для доставки, а также маршрутов доставки на их наличие и назначение ответственных курьеров. Если валидация успешна то запускается цикл распределения каждого груза по доставкам. В ходе цикла по маршруту находится ближайшая доставка в которую может быть размещен груз, если такая доставка не найдена, то создается новая доставка на дату, соответствующую дням перевозки по данному маршруту. После чего груз прикрепляется к доставке.

Графическое исполнение алгоритма представлено на изображении (Приложение В, Рисунок 1).

### **3.3 Руководство пользователя**

В данном подпункте описано руководство пользователя для приложения «Система управления доставками».

#### **3.3.1 Авторизация в личном кабинете**

Для доступа к личному кабинету, не зависимо от роли пользователя, необходимо пройти авторизацию в системе. Для того чтобы открыть окно ввода данных необходимо нажать на кнопку «Войти» в верхней навигационной панели (рисунок 7).

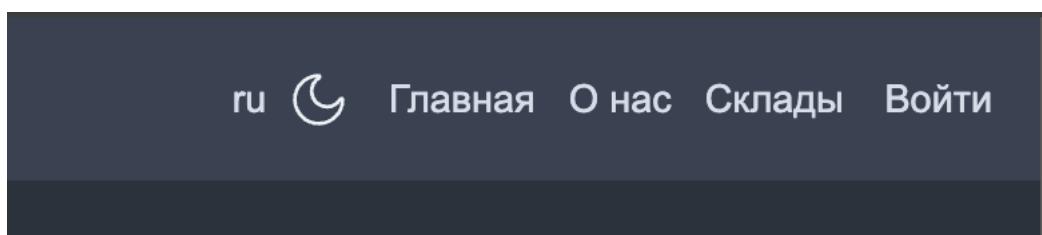


Рисунок 7 – Меню навигации с кнопкой входа в личный кабинет.

Пользователю будет предоставлено модальное окно для ввода его Email и пароля (рисунок 8).

Изм	Лист	№ докум	Подпись	Дата	Лист	39
					МИВЧ.09.03.04-5.000 ПЗ	

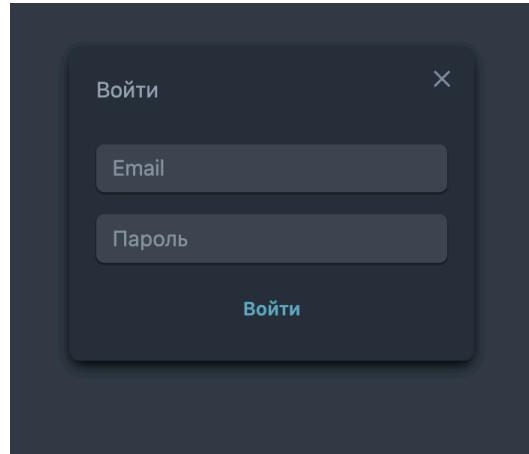


Рисунок 8 – Модальное окно ввода персональных данных для авторизации.

Если пользователем указаны некорректные данные для входа, ему будет выведено сообщение с ошибкой (рисунок 9).

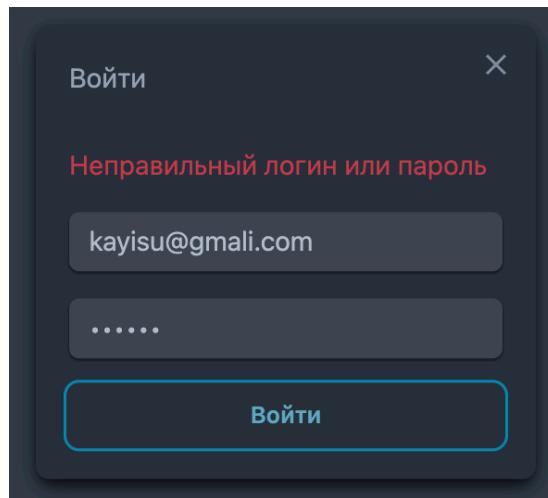


Рисунок 9 – Вывод ошибки при попытке входа.

Если данные введены правильно, то пользователь будет перенаправлен на домашнюю страницу личного кабинета в соответствии с ролью:

- 1) Для клиента, менеджера, администратора – список заказов;
- 2) Для курьера – список активных доставок.

### 3.3.2 Просмотр списка доступных складов

Для всех пользователей (включая гостей) доступна возможность просмотра складов, относительно которых может осуществляться доставка. Для просмотра складов необходимо нажать на кнопку «Склады» в верхней панели приложения,

Изм	Лист	№ докум	Подпись	Дата

после чего пользователь будет перенаправлен на страницу списка складов (рисунок 10)

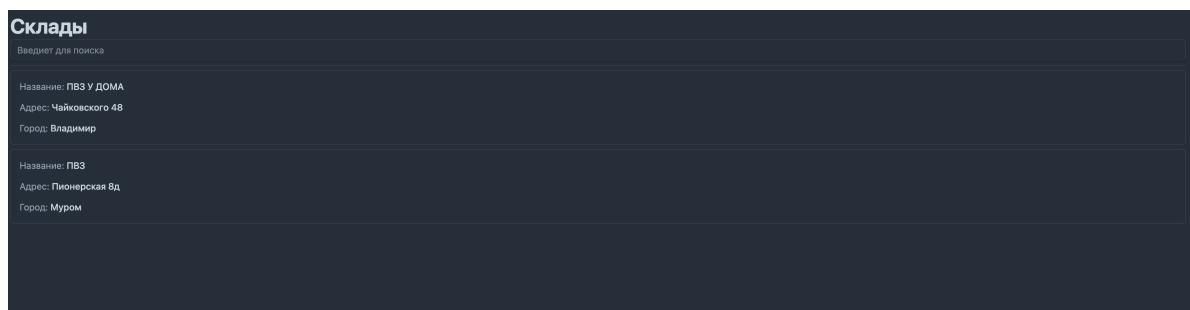


Рисунок 10 – Список складов.

Для удобства пользователей присутствует поиск складов по параметрам: название, адрес, город. Поиск осуществляется с помощью ввода поискового запроса в строку поиска и последующим выводом информации в списке (рисунок 11).

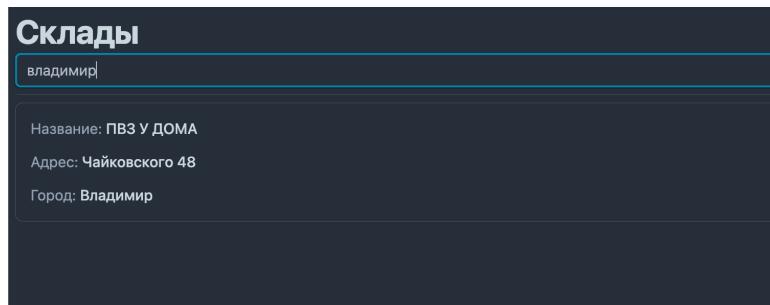


Рисунок 11 – Поиск по городу.

Для администратора на данной странице доступен дополнительный фильтр «Показать склады с ошибками» (рисунок 12). Данный фильтр отображает все склады, у которых:

- 1) Нет назначенных ответственных по маршрутам;
- 2) Есть проблемы с назначением маршрутов.

Изм	Лист	№ докум	Подпись	Дата

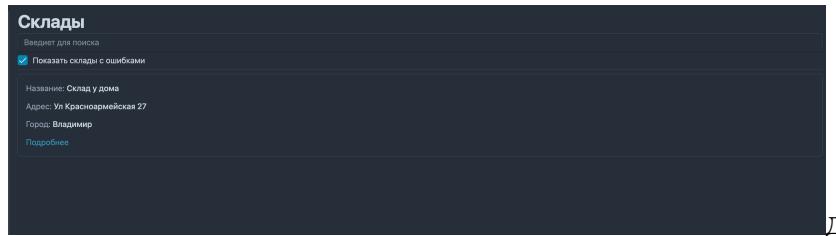


Рисунок 12 – Список складов с ошибками.

Также администратору доступен функционал перейти на детальную информацию по складу, для этого необходимо нажать кнопку «Подробнее» рядом с интересующим складом.

### 3.3.3 Навигация по блокам в приложении

Для навигации по основным блокам приложения предусмотрено меню, расположенное в боковой панели приложения (рисунок 13). В зависимости от роли пользователя количество пунктов меню различается.

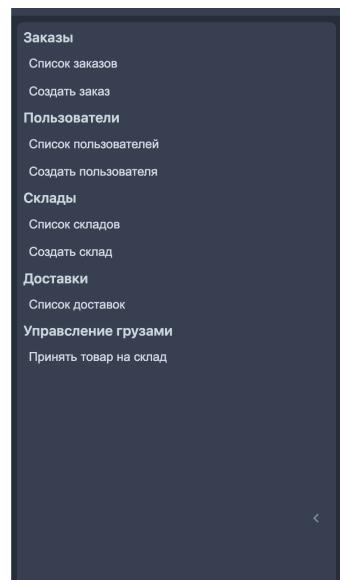


Рисунок 13 – Боковое меню.

### 3.3.4 Список заказов

Для просмотра списка заказов необходимо открыть страницу «Список заказов», для клиентов, менеджеров и администраторов так же можно перейти по ссылке «Главная» (рисунок 14).

Изм	Лист	№ докум	Подпись	Дата

The screenshot shows a dark-themed user interface for managing orders. On the left, there's a sidebar with navigation links: 'Заказы' (Orders), 'Список заказов' (List of orders), 'Создать заказ' (Create order), 'Пользователи' (Users), 'Список пользователей' (List of users), 'Создать пользователя' (Create user), 'Склады' (Warehouses), 'Список складов' (List of warehouses), 'Создать склад' (Create warehouse), 'Доставки' (Deliveries), 'Список доставок' (List of deliveries), 'Управление грузами' (Cargo management), and 'Принять товар на склад' (Accept goods to warehouse). The main area is titled 'Список заказов' (List of orders) and has a sub-header 'Номер заказа' (Order number). It includes a search bar 'Введите заказа' (Enter order) and filter buttons: 'Новый' (New), 'Ожидает оплаты' (Waiting for payment), 'Оплачено' (Paid), 'Доставляется' (Being delivered), 'Ожидает выдачи' (Waiting for delivery), and 'Статус заказа' (Order status). Below these are two checkboxes: 'Вы отправитель' (I am the sender) and 'Вы получатель' (I am the recipient). A table row shows an order with ID 10, status 'Доставляется' (Being delivered), sender 'courier@gmail.com', and recipient 'testCourier2@gmail.com'. A 'Подробнее' (More details) button is at the bottom right.

Рисунок 14 – Список заказов пользователя.

Для более быстрого поиска по заказам предоставляются фильтры. В фильтрах можно осуществлять поиск по номеру заказа, статусу заказа, отправителю или получателю (Рисунок 15).

This screenshot shows the same 'List of Orders' page as above, but with filters applied. In the 'Статус заказа' (Order status) dropdown, 'Завершен' (Completed) is selected. The 'Вы отправитель' (I am the sender) checkbox is checked, while 'Вы получатель' (I am the recipient) is unchecked. The table below shows one order with ID 8, status 'Завершен' (Completed), sender 'admin@gmail.com', and recipient 'kaynius2@gmail.com'. A 'Подробнее' (More details) button is visible on the right.

Рисунок 15 – Применение фильтров для заказов.

Для получения более подробной информации по заказу необходимо нажать на кнопку «Подробнее», после чего пользователь будет перенаправлен на страницу информации по заказу.

Изм	Лист	№ докум	Подпись	Дата

### 3.3.5 Создание заказа

Для создания заказа менеджеру или администратору необходимо выбрать соответствующий пункт в боковом меню. Для создания заказа необходимо заполнить форму заказа, указав: начальный и конечный склады и отправителя и получателя заказ (рисунок 16).

Создать заказ

Начальный склад  
Склад "Бобрик" Владимир Красносельская 1д

Конечный склад  
ПВЗ Муром Пионерская 8д

Отправитель  
kayn23@yandex.ru

Получатель  
kaynius2@gmail.com

Сохранить

Рисунок 16 - Интерфейс создания заказа.

После создания заказа пользователь будет перенаправлен на страницу информации по заказу.

### 3.3.6 Информация по заказу

Страница информации по заказу включает основные блоки: информация об отправителе и получателе, информация по маршруту, информация по грузам, информация по статусу и цене заказа (рисунок 17).

Для клиента и курьера информация представляет из себя просто справочную информацию по заказу.

Для менеджера и администратора имеется расширенный функционал по заказу. В зависимости от статуса заказа предоставляется различный интерфейс для выполнения действий с заказом: добавление грузов в заказ, проведение оплаты (рисунок 18).

Изм	Лист	№ докум	Подпись	Дата

МИВЧ.09.03.04-5.000 ПЗ

Лист

44

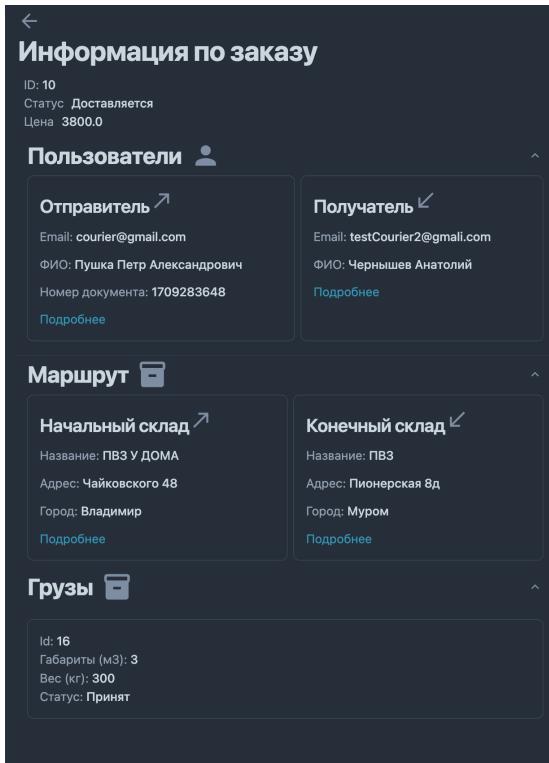


Рисунок 17 – Детальная информация по заказу.

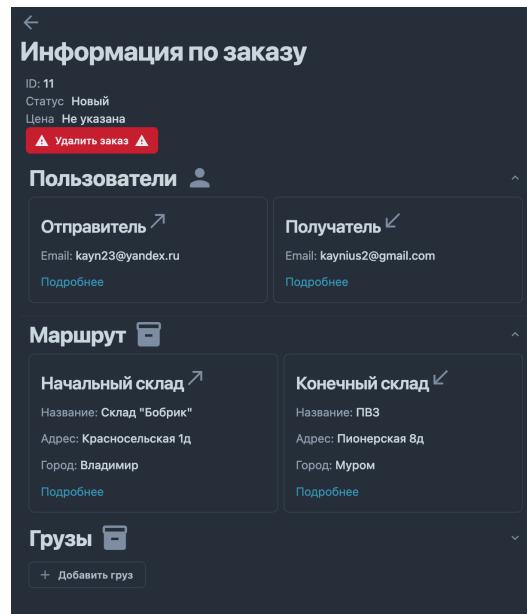


Рисунок 18 – Функционал при статусе заказа «Новый».

В новом заказе менеджер может добавить грузы в заказ. При нажатии кнопки «Добавить груз» будет открыто окно добавления груза в заказ, в котором нужно указать параметры груза (рисунок 19), после чего нажать кнопку «Добавить».

						Лист
Изм	Лист	№ докум	Подпись	Дата	МИВЧ.09.03.04-5.000 ПЗ	45

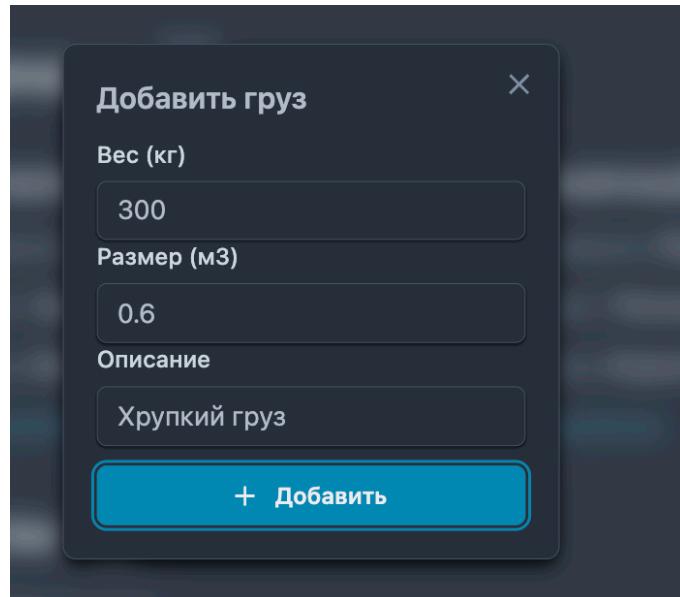


Рисунок 19 – Добавление груза в заказ.

После добавления груза в заказ, у каждого груза будет уникальный QR-код, который можно посмотреть, нажав соответствующую пиктограмму рядом с информацией о грузе. Данный код печатается и прикрепляется к грузу, и является его идентификатором на протяжение всей доставки.

Также становится доступна кнопка «Подтвердить грузы» (рисунок 20), нажатие на которую активирует возможность указать цену заказа (рисунок 21) и перевести заказ в статус «Ожидание оплаты».

Когда заказ находится в статусе «Ожидается оплата» менеджеру доступна кнопка проведения оплаты по заказу (рисунок 22). После проведения оплаты грузы из заказа автоматически распределяются по доставкам.

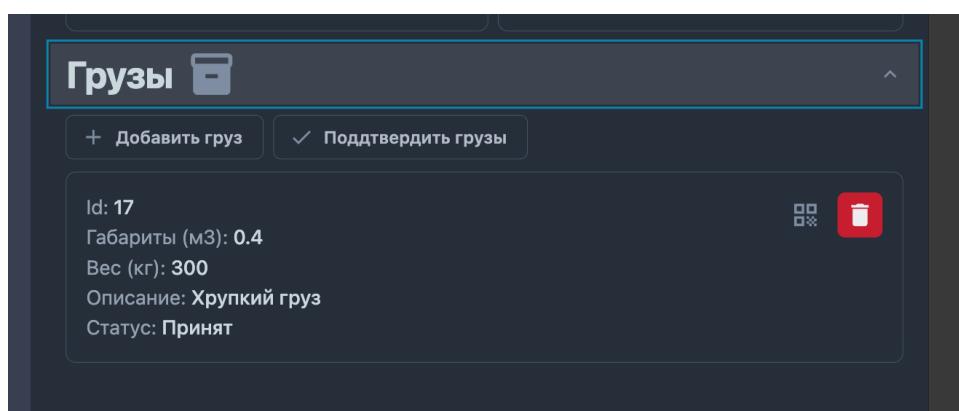


Рисунок 20 – Список грузов и подтверждение грузов.

Изм	Лист	№ докум	Подпись	Дата

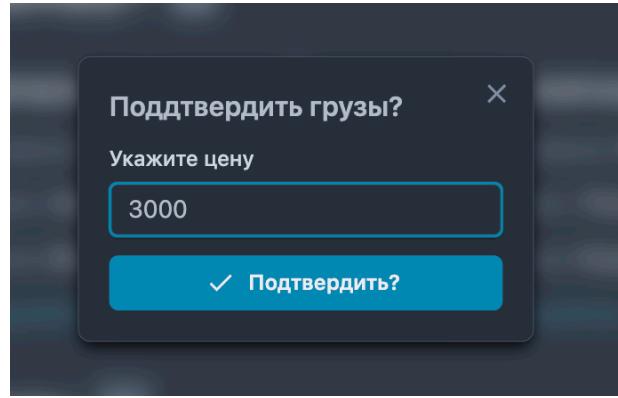


Рисунок 21 – Подтверждение принятия грузов, указание цены за доставку.

Рисунок 22 – Кнопка проведения оплаты.

Возможность удаления заказа доступна менеджеру или администратору только когда заказ находится в статусе «Новый», «Ожидает оплату». При попытке удалить заказ пользователь должен подтвердить удаление заказа (рисунок 23), так как после удаления заказа, восстановить его будет невозможно. Данную операцию следует выполнять с максимальной осторожностью.

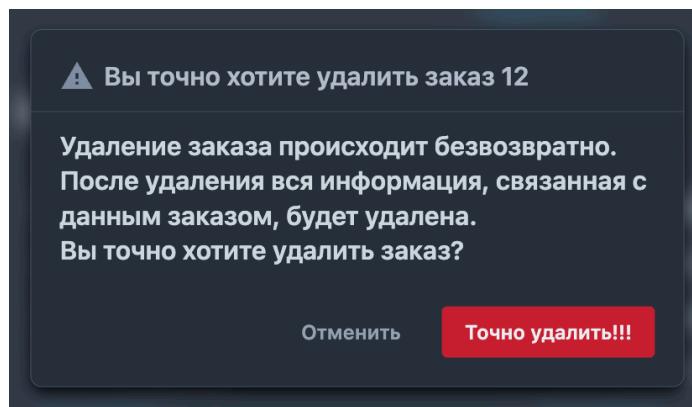


Рисунок 23 – Диалоговое окно удаления заказа.

Изм	Лист	№ докум	Подпись	Дата

После того как все грузы из заказа доставлены в конечный пункт выдачи заказ изменяет статус на «Ожидает выдачи», на странице информации о заказе менеджеру и администратору становится доступна кнопка «Выдать груз» (рисунок 24)

The screenshot shows the 'Information about the order' page. At the top, it displays the order ID (ID: 10), status ('Ожидает выдачи'), and price (Цена 3800.0). Below this, there are sections for 'Пользователи' (User) with 'Отправитель' (Courier) and 'Получатель' (Recipient) details, and 'Маршрут' (Route) with 'Начальный склад' (Initial warehouse) and 'Конечный склад' (Final warehouse) details. At the bottom, there is a 'Грузы' (Goods) section with a 'Выдать груз' (Deliver goods) button.

Рисунок 24 – Статус заказа «Ожидает выдачи»

При нажатие на данную кнопку менеджеру или администратору будет открыто окно сканирования qrcode груза и окно подтверждения выдачи (рисунок 25).

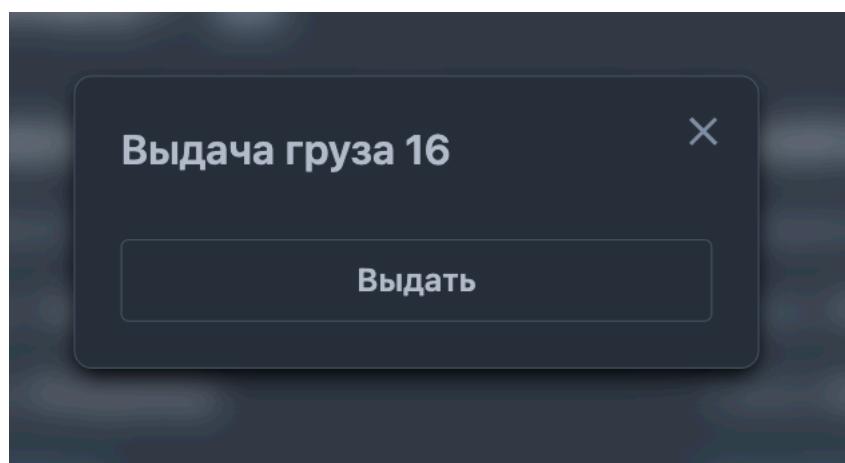


Рисунок 25 – Окно подтверждения выдачи груза.

Изм	Лист	№ докум	Подпись	Дата

После того как все грузы из заказа будут выданы заказ автоматически перейдет в статус «Завершен» (рисунок 26).

## Информация по заказу

ID: 10

Статус Завершен

Цена 3800.0

Рисунок 26 – Статус заказа «Завершен».

### 3.3.7 Управление доставками

Список доставок и взаимодействие с ними доступны курьеру и администратору. Страница с списком доставок является домашней страницей для пользователя с ролью курьер. Для того чтобы попасть на данную страницу с других страниц нужно выбрать пункт «Список доставок» в боковом меню.

Список доставок отображает все активные доставки пользователя (рисунок 27).

## Список доставок

ID 5

Статус Новая

Дата доставки 26.05.2025

Откуда ПВЗ У ДОМА, Чайковского 48, Владимир

Куда РЦ, Заводская 2д, Владимир

[Подробнее](#)

ID 7

Статус Новая

Дата доставки 06.06.2025

Откуда Склад "Бобрик", Красносельская 1д, Владимир

Куда РЦ, Заводская 2д, Владимир

[Подробнее](#)

Рисунок 27 - Список доставок.

Изм	Лист	№ докум	Подпись	Дата

МИВЧ.09.03.04-5.000 ПЗ

Лист

49

Нажав кнопку «Подробнее», пользователь попадает на страницу работы с доставкой (рисунок 28).

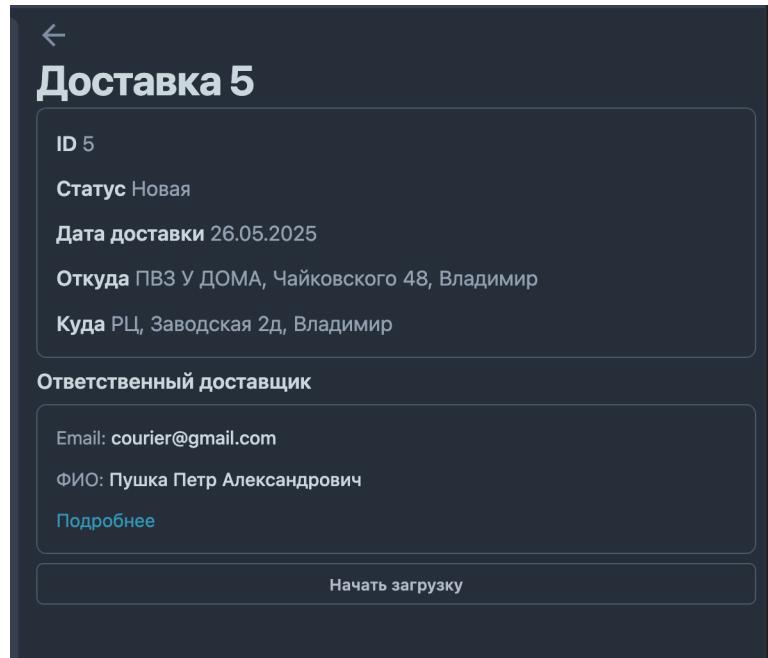


Рисунок 28 - Информация по доставке.

Когда доставка находится в статусе «Новая» пользователь может начать загрузку товара, после чего у него появится возможность принимать грузы в доставку с помощью сканирования QR-кодов грузов (рисунок 29).

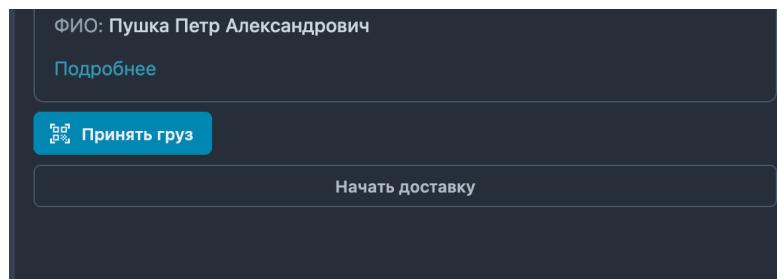


Рисунок 29 - Кнопка «Принять груз»

Приняв груз в доставку выводится оповещение о том что груз загружен (рисунок 30).

Изм	Лист	№ докум	Подпись	Дата

МИВЧ.09.03.04-5.000 ПЗ

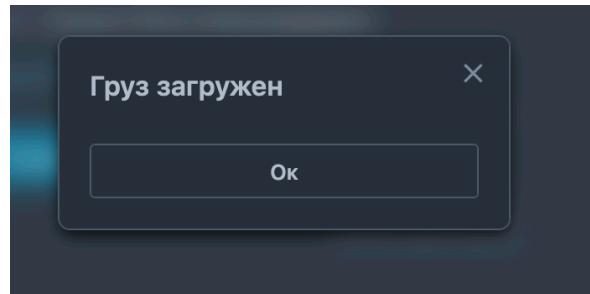


Рисунок 30 - Окно подтверждения загрузки груза.

Завершив загрузку грузов необходимо перевести доставку в статус «Доставляется» нажатием кнопки «Начать доставку».

### 3.3.8 Прием грузов на склад

Прием грузов на склад является важным технологическим процессом. Данная операция в ходе доставки выполняется минимум два раза: при поступлении груза в распределительный центр, при поступлении груза в конечный склад. Выполнение данного процесса доступно только менеджерам и администраторам приложения.

Для получения доступа к данному функционалу необходимо перейти на страницу «Принять товар на склад» в боковом меню.

Чтобы приступить к приему грузов на склад необходимо выбрать склад (рисунок 31), на котором осуществляется приемка товара. Выбор склада имеет возможности поиска, для которого необходимо начать вводить название, адрес или город. Подтверждение выбора осуществляется нажатием кнопки «Ок».

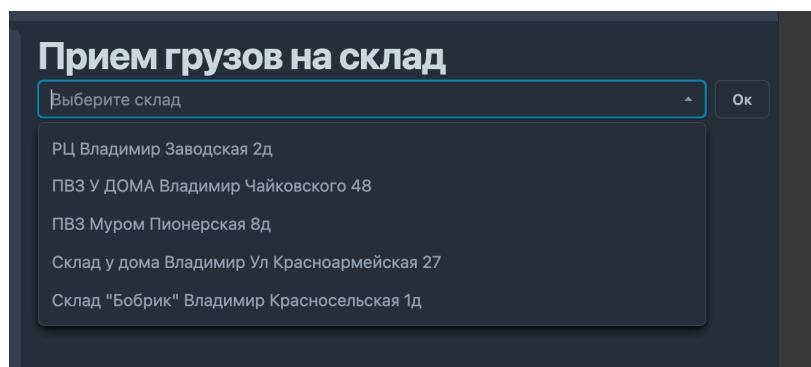


Рисунок 31 - Выбор склада приема грузов.

Изм	Лист	№ докум	Подпись	Дата
-----	------	---------	---------	------

После чего пользователю станут доступна кнопка «Принять груз» (рисунок 32) для открытия сканера QR-кодов. Сканером необходимо произвести сканирование грузов, после чего подтвердить прием груза на склад (рисунок 33).

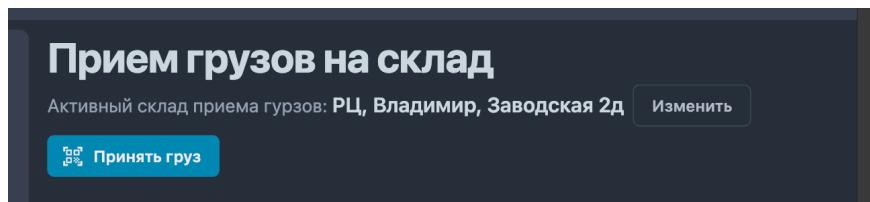


Рисунок 32 - Кнопка приема груза на склад.

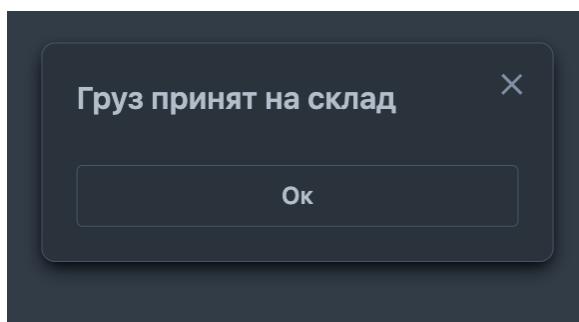


Рисунок 33 - Окно подтверждения приема груза.

Как только будут приняты все грузы из доставки, статус доставки автоматически изменится на «Закончена».

### 3.3.9 Управление пользователями

Добавление и просмотр списка пользователей доступны менеджерам и администраторам системы.

Для того чтобы посмотреть список пользователей необходимо выбрать в боковом меню пункт «Список пользователей», после чего будет представлена страница с списком пользователей (рисунок 34).

Для удобства поиска определенного пользователя на данной странице предоставлены фильтры и поиск. Поиск доступен по email или ФИО. Так же доступна сортировка по роли пользователя. Пример поиска представлен на изображении (рисунок 35).

Изм	Лист	№ докум	Подпись	Дата

The screenshot shows a dark-themed web application interface. On the left, there is a sidebar with navigation links: 'Заказы' (Orders), 'Список заказов' (List of orders), 'Создать заказ' (Create order); 'Пользователи' (Users), 'Список пользователей' (List of users), 'Создать пользователя' (Create user); 'Склады' (Warehouses), 'Список складов' (List of warehouses), 'Создать склад' (Create warehouse); 'Доставки' (Deliveries), 'Список доставок' (List of deliveries); 'Управление грузами' (Cargo management), 'Принять товар на склад' (Accept goods to warehouse). The main area is titled 'Список пользователей' (User list). It features a 'Фильтровать по роли' (Filter by role) dropdown set to 'Выберите роли' (Select roles), a 'Поиск по email или фио' (Search by email or name) input field containing 'введите email или ФИО', and a search result table. The first result shows 'Email: admin@gmail.com', 'Роли: Клиент, Админ', and a 'Подробнее' (More details) button. The second result shows 'Email: kayn23@yandex.ru', 'Роли: Клиент, Менеджер', and a 'Подробнее' button. The third result shows 'Email: kaynius2@gmail.com', 'Роли: Клиент', and a 'Подробнее' button. The fourth result shows 'Email: courier@gmail.com', 'ФИО: Пушка Петр Александрович', 'Номер документа: 1709283648', 'Роли: Клиент, Курьер', and a 'Подробнее' button.

Рисунок 34 – Страница «Список пользователей».

This screenshot shows the same 'User List' page as in Figure 34, but with a different state. The 'Фильтровать по роли' dropdown now has 'Курьер' (Courier) selected. The search results table is identical to Figure 34, displaying the same four user entries with their respective details and 'Подробнее' buttons.

Рисунок 35 – Поиск по пользователям.

При нажатии на кнопку «Подробнее» менеджер попадает на страницу информации по конкретному пользователю (рисунок 36). На данной странице предоставлен функционал для внесения изменений в данные пользователя (рисунок 37), изменения ролей пользователя (рисунок 38) и добавление информации по автомобилю для курьеров (рисунок 39).

Изм	Лист	№ докум	Подпись	Дата

<

### Информация о пользователе id: 4

Email: courier@gmail.com  
ФИО: Пушка Петр Александрович  
Номер документа: 1709283648  
Роли: Клиент, Курьер

Добавить роль    Изменить данные

Добавить автомобиль

**Название** Газель  
**Объем** 50 м3  
**Грузоподъемность** 2000 кг  
**Активная машина**

Изменить

**Название** транспортер  
**Объем** 7 м3  
**Грузоподъемность** 700 кг

Изменить

Рисунок 36 – Информация по пользователю.

### Изменить информацию о пользователе

Введите email\*  
courier@gmail.com

Введите имя  
Петр

Введите отчество  
Александрович

Введите фамилию  
Пушка

Введите паспортные данные  
1709283648

Сохранить

Рисунок 37 – Изменение данных пользователя.

### Добавить пользователю роль

Клиент ✕   Курьер ✕  

✓ Сохранить

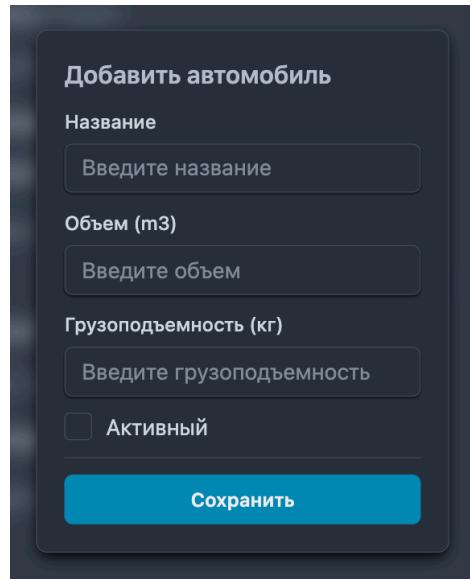
Рисунок 38 – Изменение ролей пользователя.

Изм	Лист	№ докум	Подпись	Дата

МИВЧ.09.03.04-5.000 ПЗ

Лист

54



Добавить автомобиль

Название  
Введите название

Объем (м3)  
Введите объем

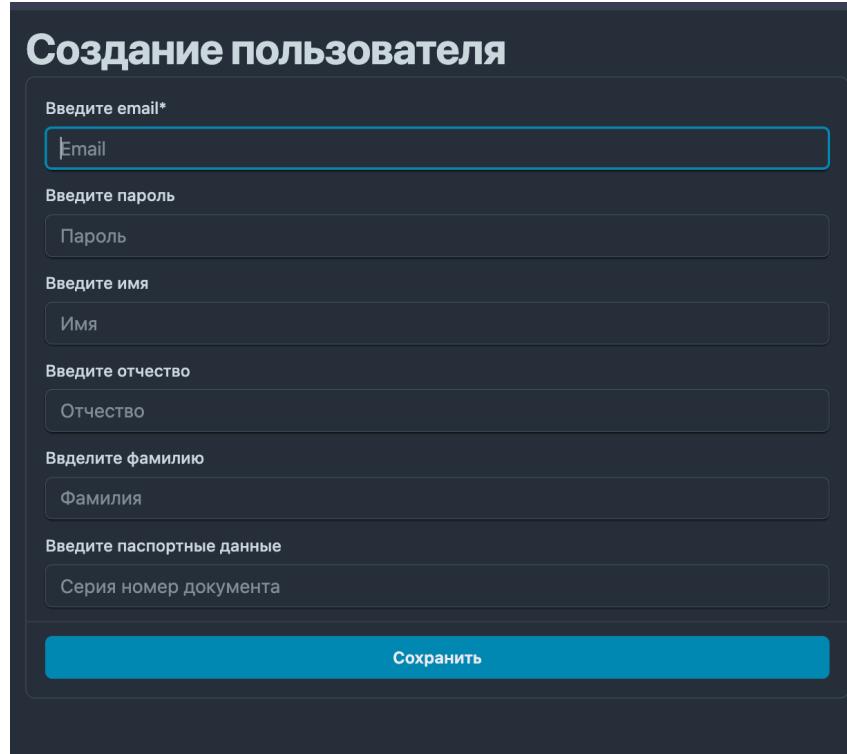
Грузоподъемность (кг)  
Введите грузоподъемность

Активный

**Сохранить**

Рисунок 39 – Добавление информации по автомобилю.

Для того чтобы создать нового пользователя в системе необходимо перейти на страницу «Создать пользователя» из бокового меню, после чего заполнить форму нового пользователя (рисунок 40). После создания пользователя, пароль выдается ему менеджером.



Создание пользователя

Введите email\*  
Email

Введите пароль  
Пароль

Введите имя  
Имя

Введите отчество  
Отчество

Введите фамилию  
Фамилия

Введите паспортные данные  
Серия номер документа

**Сохранить**

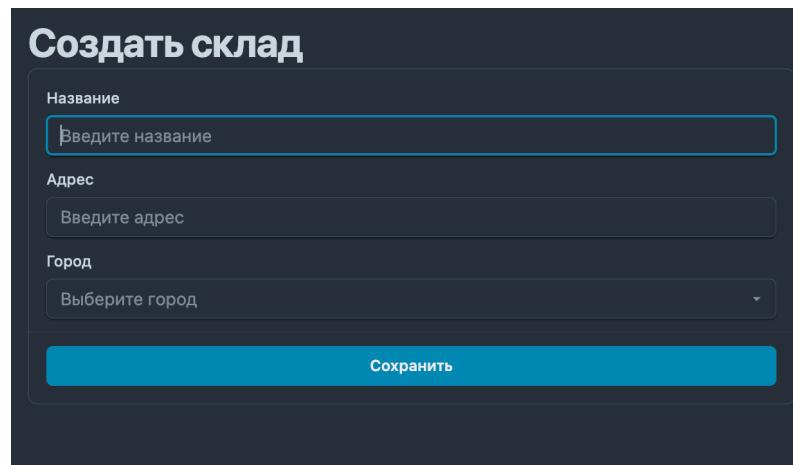
Рисунок 40 – Форма создания нового пользователя.

Изм	Лист	№ докум	Подпись	Дата
-----	------	---------	---------	------

### 3.3.10 Управление складами

Функционал создания и изменения информации по складам доступна только администратору системы.

Для того чтобы создать новый склад необходимо перейти в соответствующий пункт меню в боковой панели «Создать склад», после чего заполнить форму нового склада (рисунок 41).



Создать склад

Название  
Введите название

Адрес  
Введите адрес

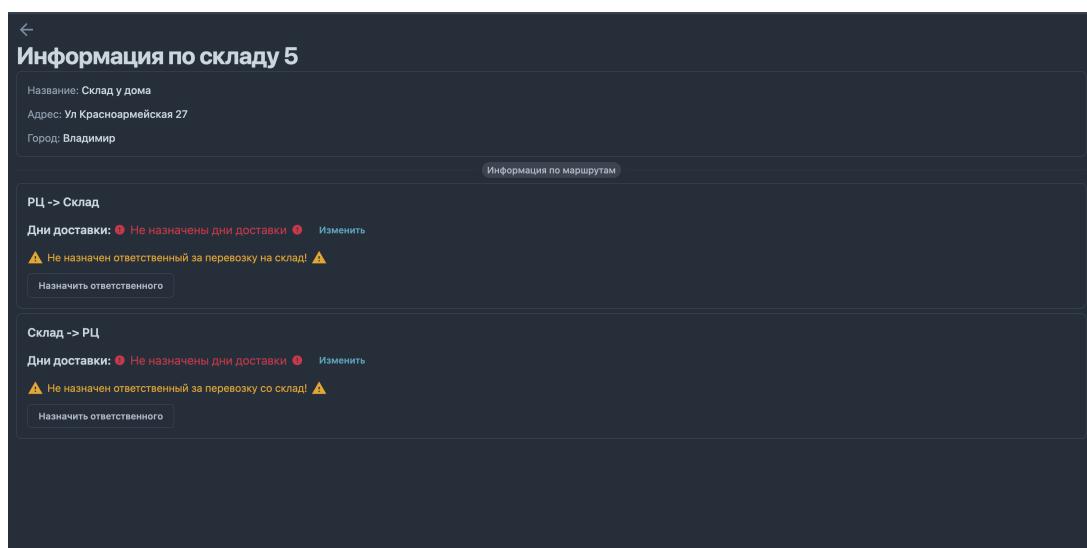
Город  
Выберите город

Сохранить

Форма создания нового склада с полями для ввода названия, адреса и города, а также кнопкой "Сохранить".

Рисунок 41 – Форма создания нового склада.

После чего администратор будет перенаправлен на страницу информации по складу (рисунок 42).



←  
Информация по складу 5

Название: Склад у дома  
Адрес: Ул Красноармейская 27  
Город: Владимир

(Информация по маршрутам)

РЦ -> Склад  
Дни доставки: Не назначены дни доставки Изменить  
⚠ Не назначен ответственный за перевозку на склад! Изменить  
Назначить ответственного

Склад -> РЦ  
Дни доставки: Не назначены дни доставки Изменить  
⚠ Не назначен ответственный за перевозку со склада! Изменить  
Назначить ответственного

Форма информации по складу с двумя блоками: РЦ -> Склад и Склад -> РЦ. В каждом блоке есть сообщение о том, что не назначены дни доставки и отсутствует ответственный за перевозку, с кнопками для изменения.

Рисунок 42 - Информация по складу с не назначенными ответственными по маршрутам.

Изм	Лист	№ докум	Подпись	Дата

МИВЧ.09.03.04-5.000 ПЗ

Лист

56

Для того чтобы склад стал доступен для оформления заказов для него необходимо назначить ответственного (рисунок 43) и назначить дни перевозки (рисунок 44) для маршрутов до распределительного центра и из распределительного центра.

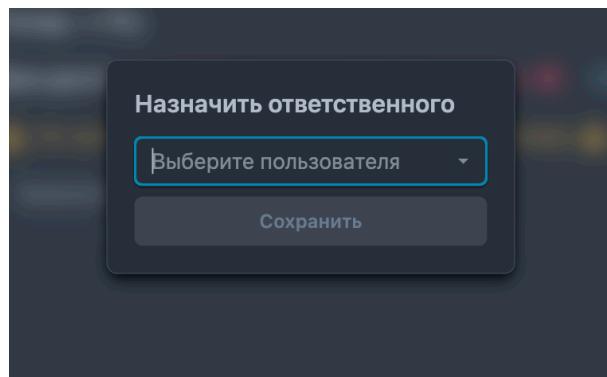


Рисунок 43 - Назначение ответственного по маршруту.

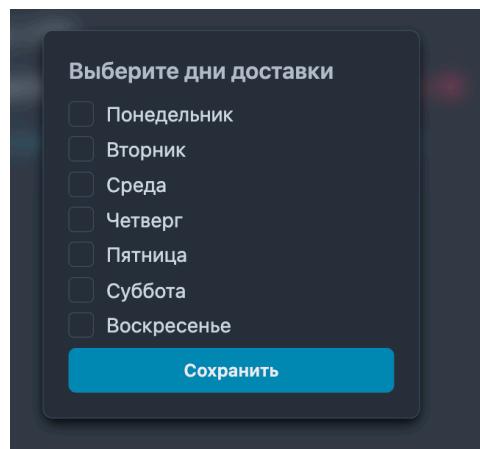


Рисунок 44 - Назначение дней доставки по маршруту.

Изм	Лист	№ докум	Подпись	Дата
-----	------	---------	---------	------

## 4 Тестирование системы

В процессе разработки программного продукта особое внимание уделялось контролю качества разрабатываемых компонентов. На ранних и последующих этапах разработки для серверной части системы проводилось модульное тестирование с использованием инструмента rswag, что позволило не только проверить корректность бизнес-логики и API-интерфейсов, но и одновременно сформировать актуальную документацию в формате Swagger/OpenAPI. Такой подход обеспечил быстрое выявление ошибок и недочётов, а также способствовал прозрачности при интеграции с клиентской частью приложения.

Следующим этапом верификации системы стало ручное тестирование, направленное на комплексную проверку функциональной корректности всех компонентов. Данный вид тестирования позволил удостовериться в работоспособности пользовательского интерфейса, корректности взаимодействия между модулями, а также в соответствии реализованной логики требованиям технического задания. Ручное тестирование охватило все ролевые модели пользователей и охватывало как типовые, так и пограничные сценарии использования системы.

Благодаря совокупному применению модульного и ручного тестирования удалось повысить надёжность и стабильность работы системы, а также обеспечить её готовность к эксплуатации в условиях реального бизнеса.

Чек-лист тестирования приложения представлен в таблице 6.

Таблица 6 – Чек-лист тестирования приложения

№	Проверка	Результат
Авторизация и проверка уровня доступа		
1	Авторизация в систему с валидными данными	Успешная авторизация
2	Авторизация в системе с невалидными данными	Сообщение об ошибке "Неправильные логин или пароль"
3	Авторизация с данными клиента	Переход на список заказов
4	Авторизация с данными курьера	Переход на список доставок

## Продолжение таблицы 6

№	Проверка	Результат
4	Авторизация с данными курьера	Переход на список доставок
5	Попытка перехода на страницу без необходимых ролей	Перенаправление на главную страницу пользователя
6	Попытка получения данных без права доступа	Отображение ошибки доступа
Работа с списком заказов		
7	Фильтрация заказов по статусу	Отображаются все заказы с выбранным статусом
8	Поиск заказа по идентификатору	Отображаются заказы подходящие под введенный идентификатор
9	Фильтрация заказов по получателю	Отображаются все заказы в которых пользователь является получателем
10	Фильтрация заказов по отправителю	Отображаются все заказы в которых пользователь является отправителем
11	Переход на детальную информацию по заказу	Открывается страница с детальной информацией по заказу
Работа с заказом		
12	Добавление груза в заказ	Открывается сканер QR-кодов, груз добавляется в заказ.
13	Просмотр QR-кода груза	Открывается модальное окно с изображением QR-кода
14	Подтверждение приема грузов в заказ	Открывается окно для указания цены доставки. Изменяется статус заказа на "Ожидается оплата"
15	Подтверждение оплаты	Изменяется статус заказа на "Доставляется", грузы добавляются в доставки
16	Выдача грузов	Открывается сканер QR-кодов, изменяется статус заказа на "Выполнен"
Создание заказа		
17	Заполнение формы заказа	Создается заказ, перенаправляется на детальную информацию по заказу
Работа с списком пользователей		
18	Фильтрация по роли пользователя	Отображаются все пользователи у которых присутствует выбранная роль
19	Фильтрация по email	Отображаются все пользователи, email которых подходит под введенное значение

## Продолжение таблицы 6

№	Проверка	Результат
<b>Работа с списком пользователей</b>		
18	Фильтрация по роли пользователя	Отображаются все пользователи у которых присутствует выбранная роль
19	Фильтрация по email	Отображаются все пользователи, email которых подходит под введенное значение
20	Фильтрация по ФИО	Отображаются все пользователи, ФИО которых подходит под введенное значение
21	Переход на детальную информацию о пользователе	Открывается страница с детальной информацией о пользователе
<b>Работа с пользователем</b>		
22	Добавить роль пользователю	Добавляется роль у пользователя
23	Удалить роль пользователю	Удаляется роль у пользователя
24	Изменить данные по пользователю	Изменяются данные в базе у пользователя
25	Добавить новый автомобиль пользователю	Добавляется новый автомобиль, остальные автомобили становятся неактивными
26	Изменить данные по автомобилю	Изменяются данные в базе по автомобилю
27	Изменить статус автомобиля на активный	Запись по автомобилю становится активной, остальные автомобили изменяют статус на не активный
<b>Создание пользователя</b>		
28	Заполнение формы пользователя	Создается запись о пользователе, открывается страница детальной информации по пользователю
<b>Создание склада</b>		
29	Заполнение формы склада	Создается запись о складе, открывается страница детальной информации по складу.
<b>Работа с списком складов</b>		
30	Поиск по имени склада	Отображаются все склады, название которых подходит под введеное значение
31	Поиск по адресу склада	Отображаются все склады, адрес которых подходит под введеное значение

## Продолжение таблицы 6

№	Проверка	Результат
Работа с списком складов		
30	Поиск по имени склада	Отображаются все склады, название которых подходит под введеное значение
31	Поиск по адресу склада	Отображаются все склады, адрес которых подходит под введеное значение
32	Поиск по городу	Отображаются все склады, город которых подходит под введенное значение
33	Фильтрация по складам с ошибками	Отображаются все склады у которых отсутствуют назначенные ответственные, не назначены дни перевозки, или у ответственного нет активного автомобиля.
34	Переход на детальную информацию о складе	Открывается страница с детальной информацией по складу
Работа с складом		
35	Просмотр информации по складу	Отображается описание склада и информация по маршрутам, связанным с данным складом
36	Отсутствие назначенного ответственного за маршрут	Отображается ошибка "Не назначен ответственный за перевозку"
37	Отсутствует назначение дней перевозки	Отображается ошибка "На назначены дни доставки"
38	Нажать изменить дни доставки	Отображается окно изменения дней доставки
39	Изменить дни доставки	Изменяются данные в базе по дням доставок по маршруту
40	Нажать назначить ответственного	Открывается окно выбора ответственного по маршруту
41	Назначить ответственного по маршруту	Изменяются данные в базе по ответственному по маршруту
42	Нажать изменить ответственного	Открывается окно выбора ответственного по маршруту
43	Изменить ответственного по маршруту	Изменяются данные в базе по ответственному по маршруту

## **Заключение**

В ходе выполнения бакалаврской работы была разработана информационная система для автоматизации логистических процессов службы доставки. Основное внимание было уделено созданию удобного и функционального инструмента для работы с заказами, складами, пользователями и доставками, с возможностью масштабирования.

Разработка включала в себя проектирование архитектуры системы, построение логической и физической моделей данных, реализацию клиентской части на основе библиотеки React и серверной части на Ruby on Rails с использованием PostgreSQL, Redis и Docker. Особое внимание было уделено вопросам безопасности, надёжности хранения данных и удобству пользовательского интерфейса.

В процессе реализации были внедрены механизмы ролевой авторизации, взаимодействие с QR-кодами, автоматическое распределение грузов по доставкам, а также интеграция аналитических инструментов. Для серверной части были написаны модульные тесты с использованием фреймворка rswag, что позволило обеспечить не только тестирование API, но и его автоматическую документацию. Также было проведено ручное тестирование пользовательских сценариев, что позволило убедиться в корректной работе системы на всех этапах.

Результатом проделанной работы стал полнофункциональный программный продукт, соответствующий требованиям технического задания и обеспечивающий автоматизацию ключевых операций логистической службы. Разработанная система может быть использована как основа для дальнейшего внедрения и развития цифровых решений в сфере управления доставками и логистикой.

Вся разработанная кодовая база представлена на GitHub [10].

Изм	Лист	№ докум	Подпись	Дата

## Список литературы

- 1) Hartl, P. Ruby on Rails Tutorial: Learn Web Development with Rails. – 7th ed. – Boston: Addison-Wesley, 2022. – 784 p.
- 2) Руби, С. Разработка веб-приложений на Rails 7 : [пер. с англ.] / С. Руби, Д. Томас, Д. Ханссон. — Роли : Прагматик Буксхелф, 2022. — 558 с.
- 3) Официальная документация Ruby on Rails: сайт. — URL: <https://guides.rubyonrails.org> (дата обращения: 10.05.2025).
- 4) Obe, R., Hsu, H. PostgreSQL: Up and Running. – 4th ed. – Sebastopol: O'Reilly Media, 2022. – 342 p.
- 5) Изучаем React: современные шаблоны для разработки приложений : [пер. с англ.]. — 3-е изд. / А. Бэнкс, Э. Порсельо. — Севастополь : O'Reilly Media, 2023. — 412 с.
- 6) Документация Feature-Sliced Design: сайт. — URL: <https://feature-sliced.design> (дата обращения: 10.05.2025).
- 7) Официальная документация React: сайт. — URL: <https://react.dev> (дата обращения: 10.05.2025).
- 8) Книга RSpec: разработка через поведение с RSpec, Cucumber и друзьями : [пер. с англ.] / Д. Челимски, Д. Астелс, Б. Хелмкамп. — Роли : Прагматик Буксхелф, 2010. — 450 с.
- 9) Docker: запуск и эксплуатация : [пер. с англ.]. — 3-е изд. / С. Кейн, К. Маттиас. — Севастополь : O'Reilly Media, 2023. — 298 с.
- 10) Github репозиторий проекта: сайт. — URL: [https://github.com/kayn23/diplom\\_docs](https://github.com/kayn23/diplom_docs)
- 11) Swagger документация к проекту: сайт — URL: <https://intent-buzzard-sensible.ngrok-free.app/api-docs>.

Изм	Лист	№ докум	Подпись	Дата

МИВЧ.09.03.04-5.000 ПЗ

Лист

63

## Приложение А

### Бизнес процессы

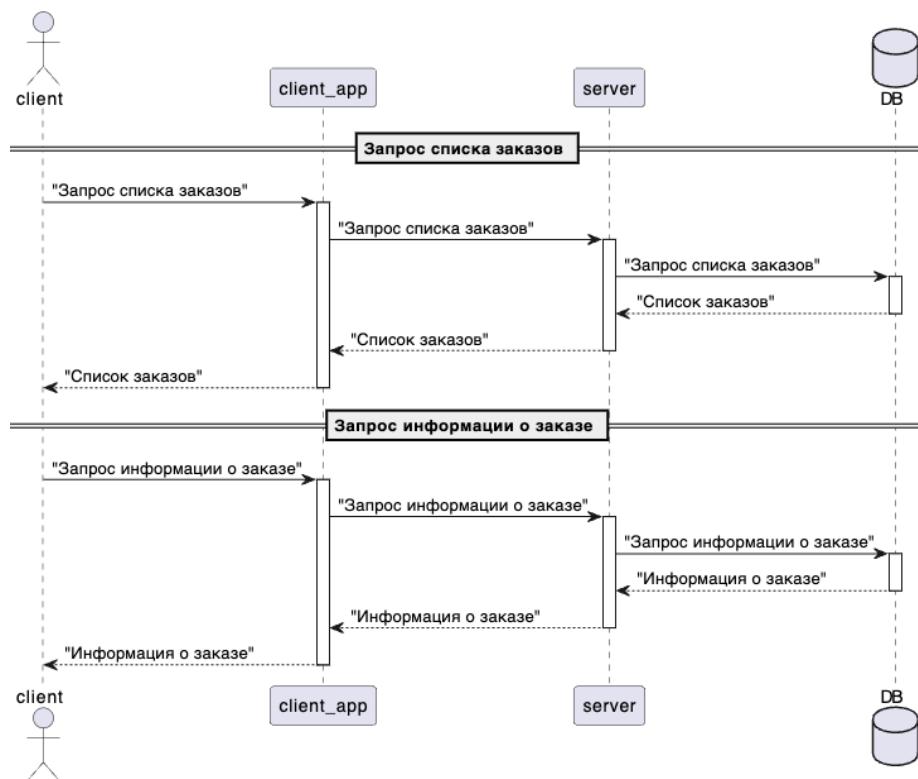


Рисунок 1 – Диаграмма последовательностей пользователя.

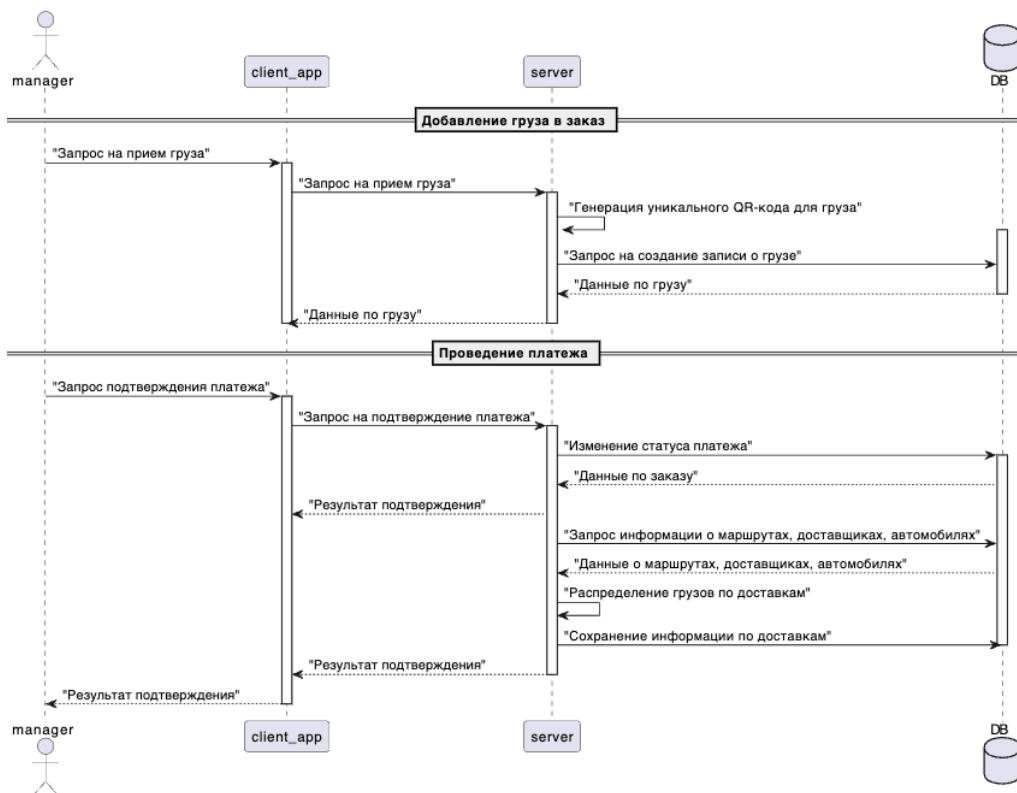


Рисунок 2 – Диаграмма последовательностей для менеджера склада, часть 1.

Изм	Лист	№ докум	Подпись	Дата

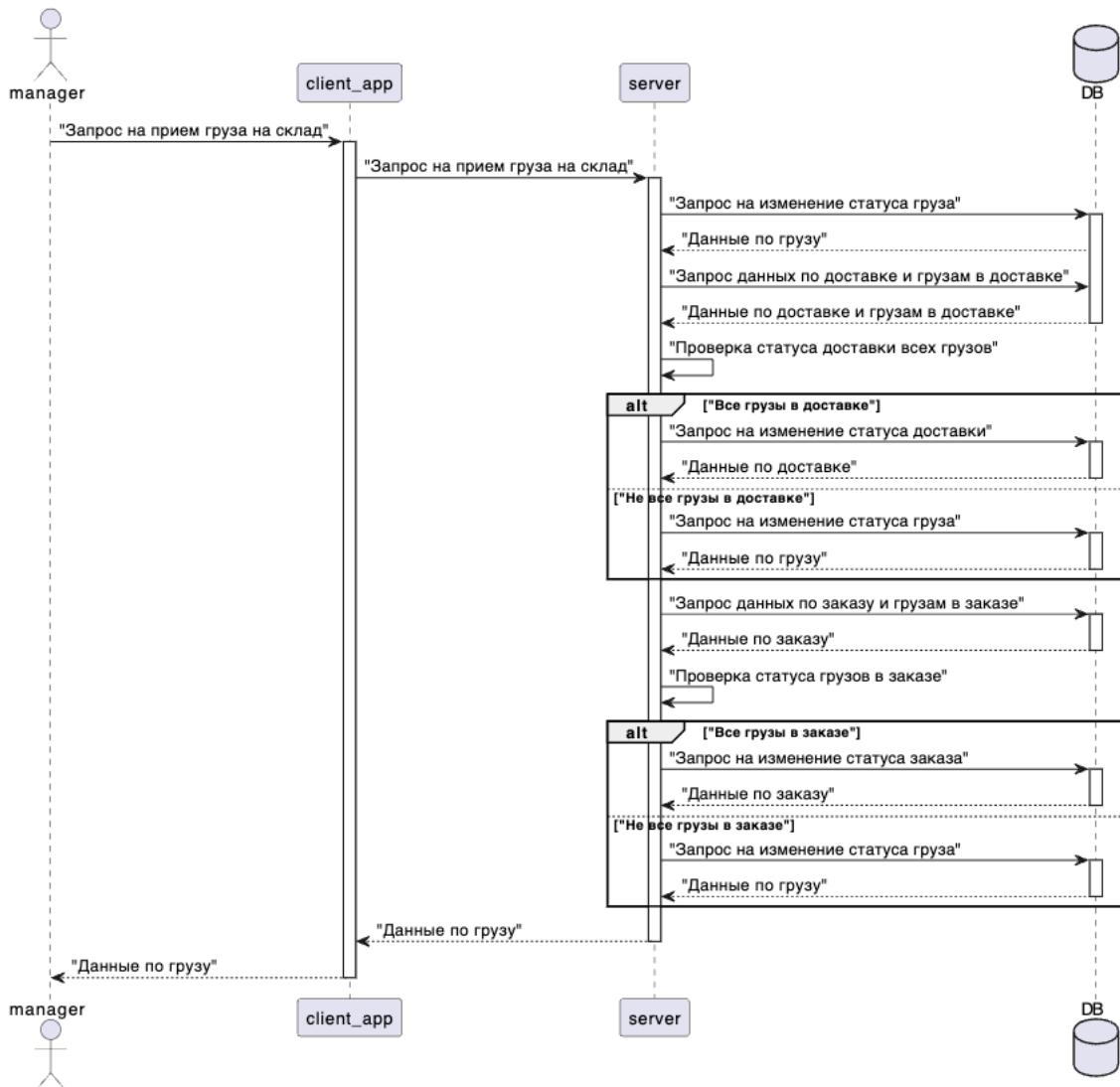


Рисунок 3 – Диаграмма последовательностей для менеджера склада, часть 2.

Изм	Лист	№ докум	Подпись	Дата

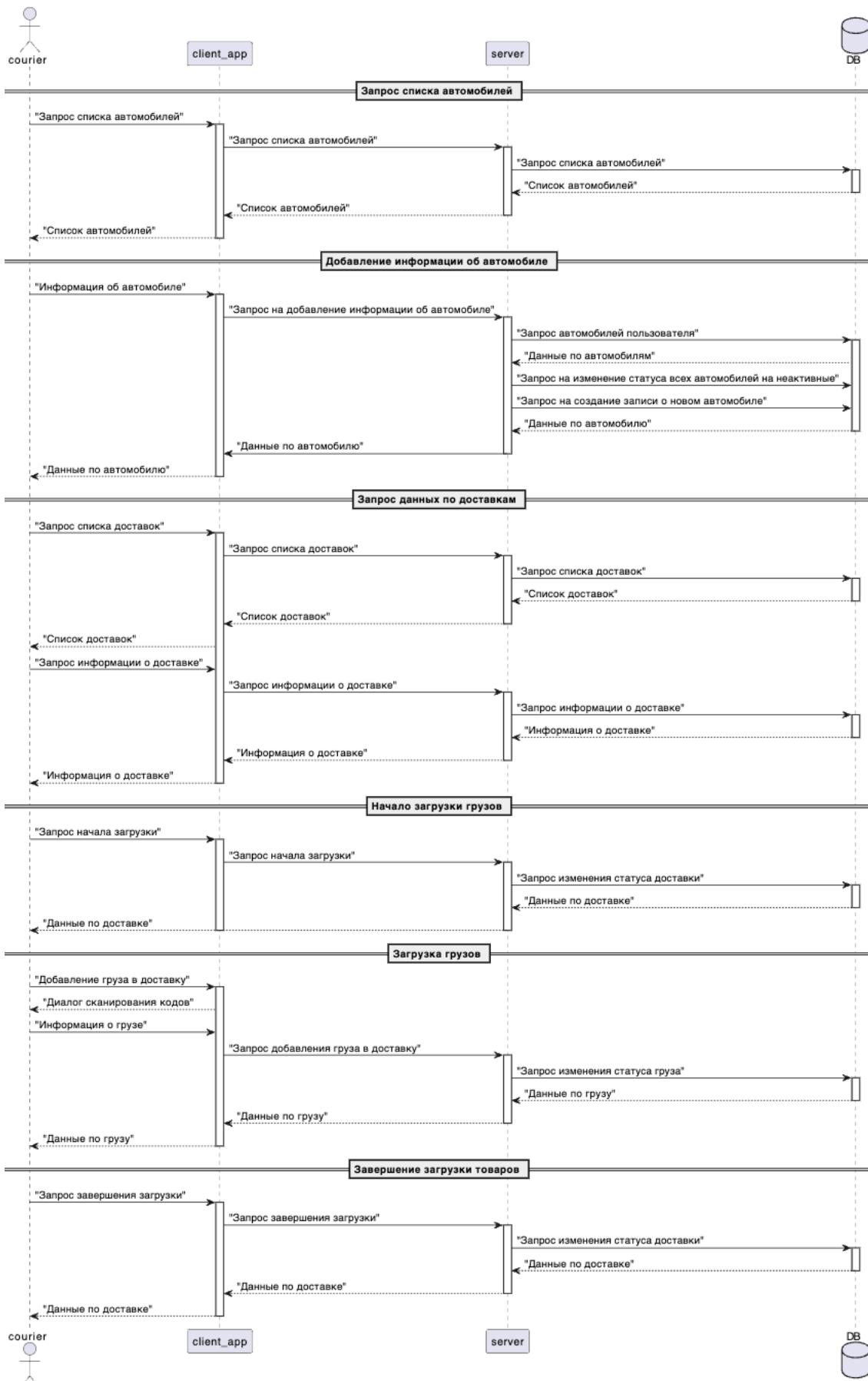


Рисунок 4 – Диаграмма последовательностей для курьера.

Изм	Лист	№ докум	Подпись	Дата

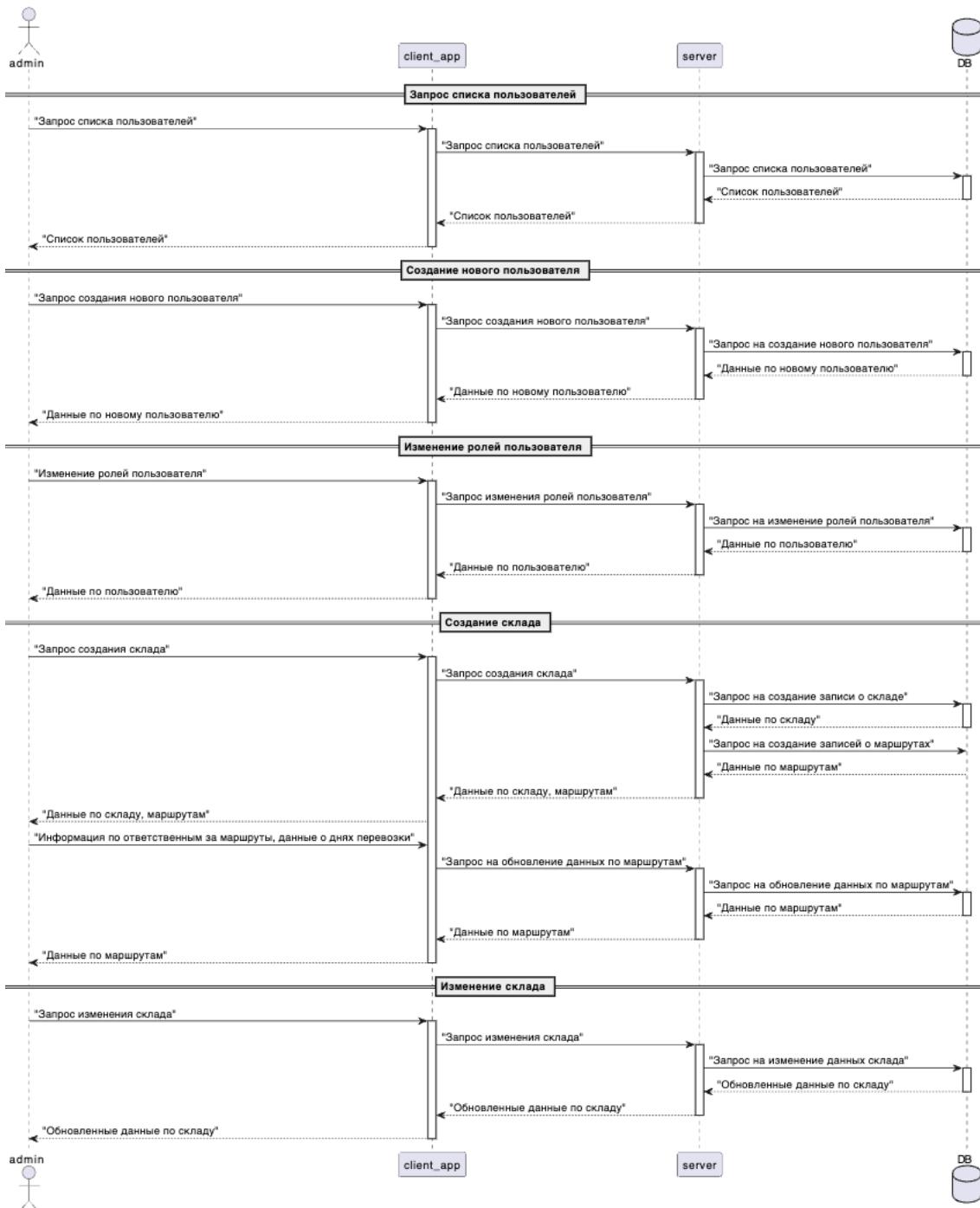


Рисунок 5 – Диаграмма последовательностей для администратора.

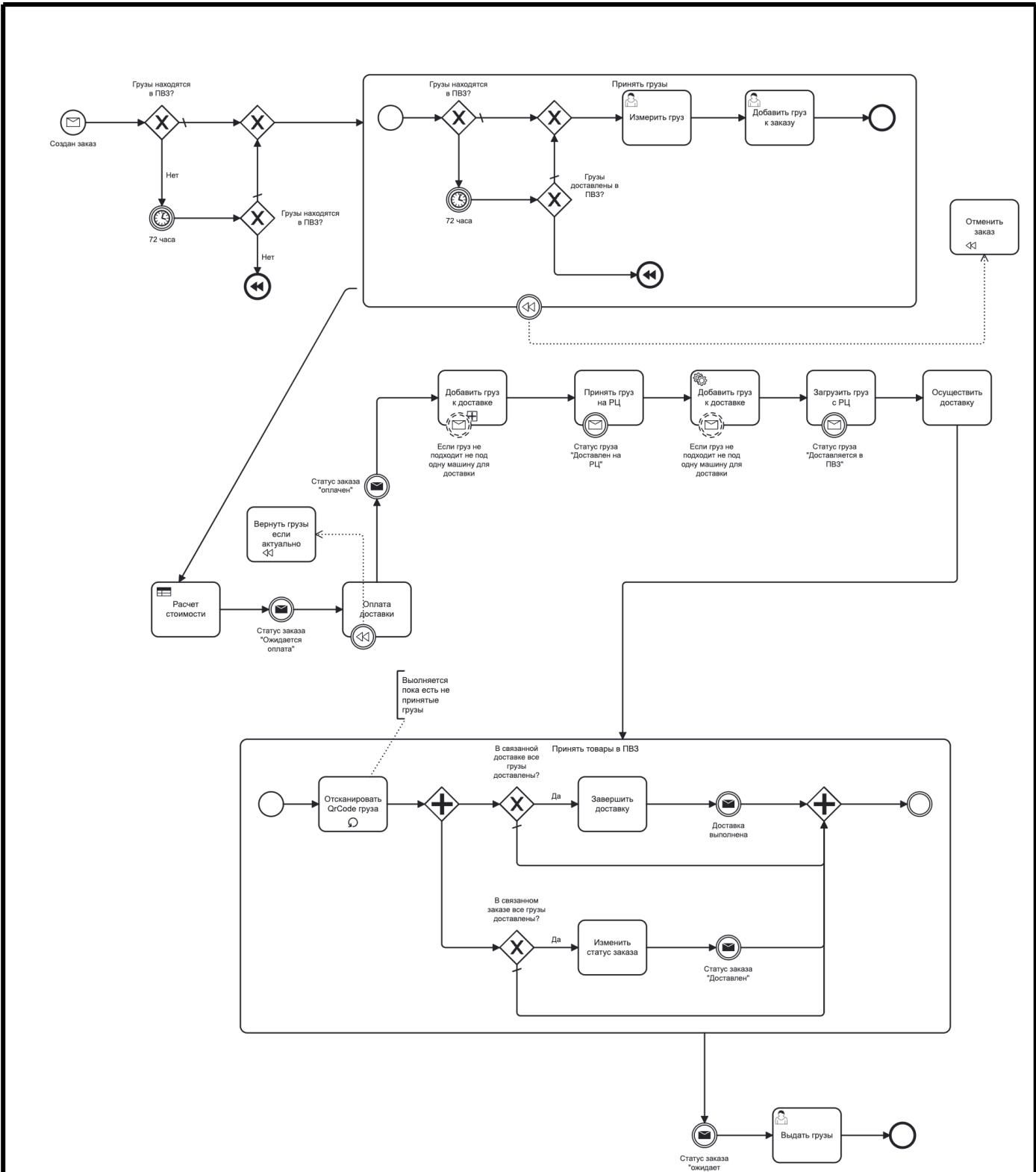


Рисунок 6 – Схема бизнес-процесса.

Изм	Лист	№ докум	Подпись	Дата

## Приложение Б

### Моделирование данных

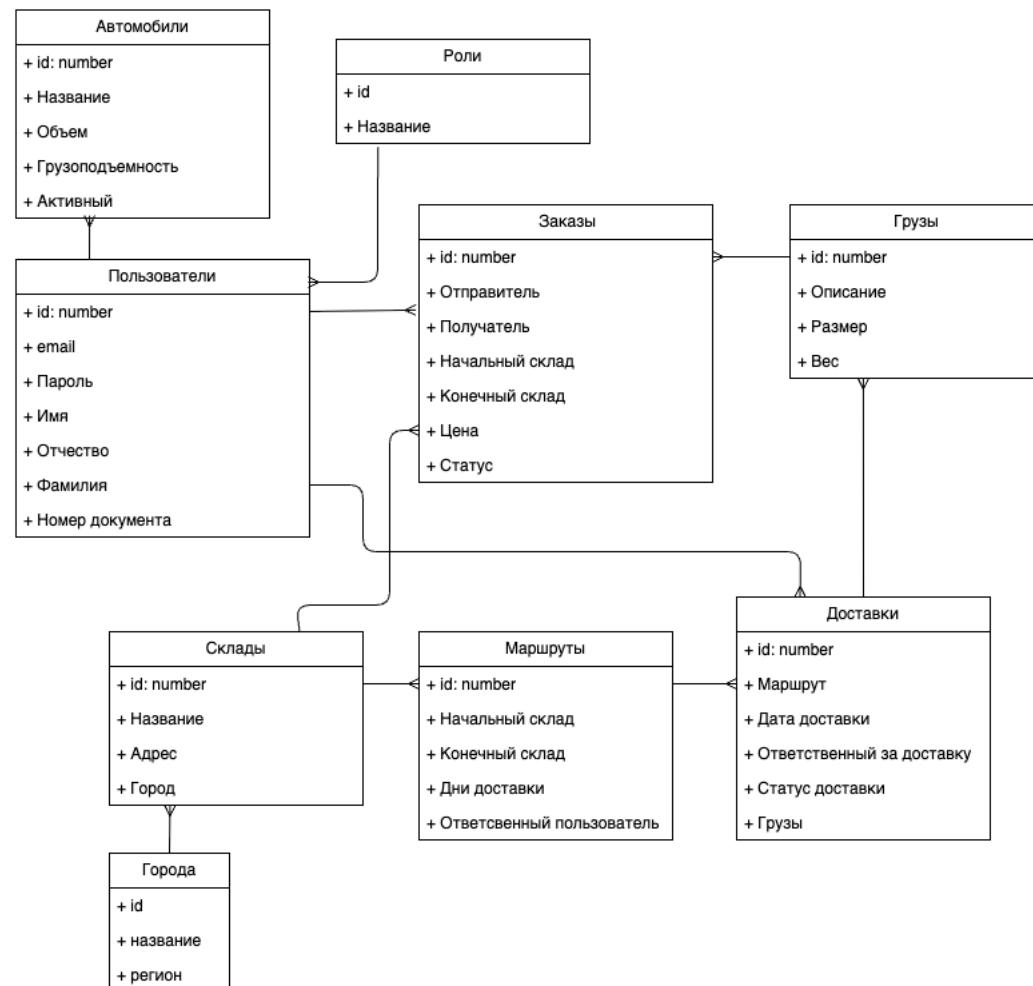


Рисунок 1 – Логическая модель данных

Изм	Лист	№ докум	Подпись	Дата

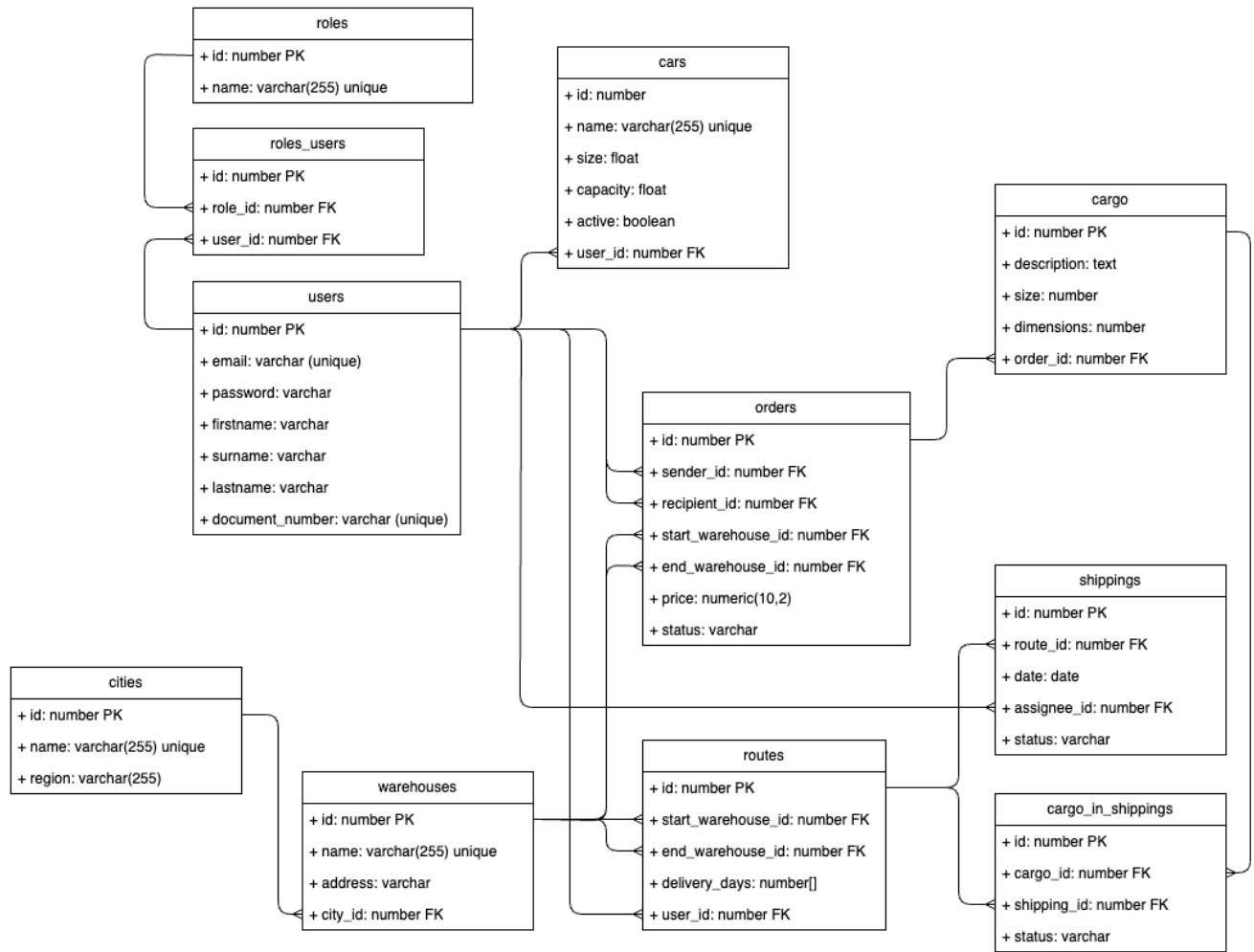


Рисунок 2 – Физическая модель данных.

## Приложение В

### Алгоритмы

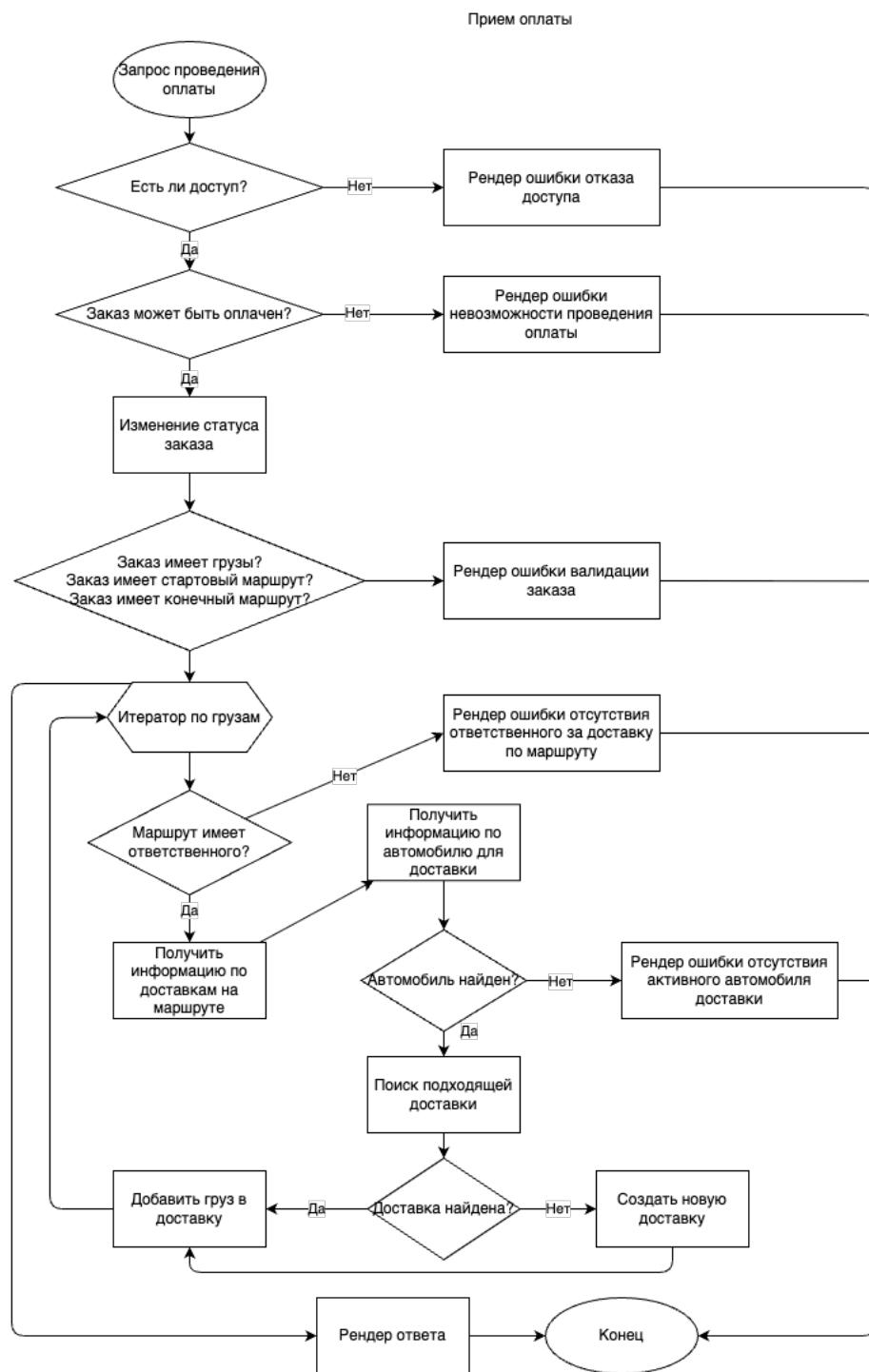


Рисунок 1 – Алгоритм приема оплаты.

Изм	Лист	№ докум	Подпись	Дата