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

Факультет ФИТР
Кафедра ПИн

КУРСОВАЯ РАБОТА

по дисциплине Разработка приложений для мобильных операционных систем

Тема Разработка приложения-терминала для ИС системы доставки грузов

Руководитель

Колпаков А.А.

(оценка

(подпись) (дата)

Члены комиссии

Студент ПИНз-120
(группа)

Чернышев А.Е.

(подпись) (дата)

(подпись) (дата)

(подпись) (дата)

Муром 2024 г.

Муромский институт (филиал) федерального государственного бюджетного образовательного учреждения высшего образования
«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»

Факультет информационных технологий и радиоэлектроники

«УТВЕРЖДАЮ»

Зав. кафедрой ПИН _____А.Л. Жизняков
(подпись)
«___»_____ 2024 г.

З А Д А Н И Е

На курсовую работу по курсу ***Разработка приложений для мобильных операционных систем***

Студенту *Чернышеву А Егр. ПИНз-120*

1. Тема работы Разработка приложения-терминала для ИС системы доставки грузов
2. Сроки сдачи студентом законченного проекта *«21» декабря 2024 г.*
3. Исходные данные к проекту
Разработать приложение для мобильной операционной системы позволяющее работать в системе доставки грузов. Приложение должно представлять собой терминал для курьеров и менеджеров складов. Для курьеров приложение должно предоставлять функционал создания грузоперевозки, добавления грузов в доставку использую считыванием qrcode груза, удаления доставок без грузов. Для менеджера приложение должно предоставлять механизм выбора склада на котором осуществляется приемка грузов и иметь возможность выполнять прием груза используя qrcode грузов.
Дополнительные требования к разрабатываемой системе:
   a) Операционная система Android, язые Kotlin;
   b) Общение с сервером посредством REST API;
   c) Разделение частей приложения для разных ролей;
   d) Репозиторий системы контроля версий – GitHub;
   e) Провести функциональное тестирование разработанного приложения.

4. Содержание расчетно–пояснительной записки (перечень подлежащих разработке вопросов).

Аннотация (на двух языках)
Содержание
Введение
1. Анализ технического задания
2. Разработка алгоритмов
3. Проектирование работы системы
4. Руководство программиста
5. Руководство пользователя
Заключение

Список используемой литературы

Приложение 1. Текст программы

Приложение 2. Снимки окон программы (скриншоты программы)

5. Перечень графического материала (с точным указанием обязательных чертежей и графиков)

Описание структур данных

Текст программы с комментариями

Скриншоты окон программы

6. Рекомендуемая литература

1. *Введение в разработку приложений для смартфонов на ОС Android / А.Семакова – М.: Национальный открытый Университет «ИНТУИТ», 2016*
2. *Колисниченко Д.Н. Программирование для Android 5. Самоучитель. — СПб.: БХВ-Петербург, 2015. — 303 с.*
3. *Дейтел П., Дейтел Х., Уолд А. Android для разработчиков. 3-е изд. — СПб.: Питер, 2016.*
4. *Гриффитс Дэвид, Гриффитс Дон Head First. Программирование для Android. 2-е изд. — СПб.: Питер, 2018. — 912 с.*

/

7. Дата выдачи задания *14.09.2024*

8. Календарный график работы над проектом (на весь период проектирования, с указанием сроков выполнения и трудоемкости отдельных этапов)

Анализ технического задания	*10%, 3 нед*
Разработка моделей данных	*20%, 4 нед.*
Проектирование работы системы	*35%, 6 нед.*
Разработка и реализация системы	*70%, 12 нед.*
Тестирование системы	*90%, 14 нед.*
Оформление пояснительной записки	*100%, 15 нед.*

Руководитель _____

(подпись)

Задание принял к исполнению (дата) _____

Подпись студента _____

Примечание. Это задание прилагается к законченному проекту.

В данной курсовой работе представлена разработка приложения-терминала для сервиса доставки грузов. Приложение связывается с сервером для получения данных по HTTP по стандарту REST API. Разработка осуществлялась с использованием языка программирования Kotlin для Android устройств.

This thesis presents the development of a terminal application for a cargo delivery service. The application connects to a server to retrieve data via HTTP using the REST API standard. The development was carried out using the Kotlin programming language for Android devices.

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. Реализация этих

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

Целью данной курсовой работы является разработка приложения для сервиса доставки грузов, которое будет предоставлять удобный интерфейс для управления доставками и грузами. Приложение реализовано с использованием языка программирования Kotlin для Android-устройств и предназначено для двух ролей пользователей: Доставщик и Менеджер. Доставщик может просматривать список открытых доставок, создавать новые, просматривать информацию по доставкам и добавлять груз с помощью QR-кодов. Менеджер может выбрать склад для приемки грузов и подтвердить приемку с помощью QR-кодов. Все данные обрабатываются на сервере через HTTP с использованием стандартного REST API.

1 Анализ технического задания

1.1 Описание предметной области

Предметной областью данного курсового проекта является система доставки грузов, предназначенная для оптимизации и упрощения процессов управления доставками и грузами. Система ориентирована на два типа пользователей: Доставщика и Менеджера, каждый из которых выполняет свои специфические задачи в процессе работы с доставками.

1. Доставщик — пользователь, который непосредственно занимается физической доставкой грузов. Основные его задачи включают:

   1.1. Просмотр списка текущих открытых доставок, которые ему нужно выполнить.

   1.2. Создание новой доставки с указанием всех необходимых данных о грузе и адресе.

   1.3. Просмотр подробной информации по каждой доставке, включая статус, информацию о грузе и получателе.

   1.4. Добавление груза в доставку с помощью сканирования QR-кода, что позволяет ускорить процесс регистрации товаров и их отслеживание.

2. Менеджер — пользователь, отвечающий за организацию и управление процессом приема и отправки грузов. Его основные функции включают:

   2.1. Выбор склада для приемки грузов, где товары будут проверяться и учитываться.

   2.2. Принятие груза на склад с помощью сканирования QR-кода, что упрощает и ускоряет процесс регистрации товара на складе.

Система работает с сервером через HTTP REST API, что позволяет обеспечивать синхронизацию данных между устройствами пользователей и центральным сервером. Все данные о доставках, грузах, складах и их статусах хранятся на сервере, что обеспечивает единую точку для обработки и хранения информации.

Использование QR-кодов в системе позволяет повысить точность и скорость операций, минимизируя человеческий фактор и ошибки при регистрации грузов, а также облегчая отслеживание каждого этапа доставки. Это решение особенно важно для крупных логистических компаний, где количество доставок и товаров может быть значительным.

Приложение, разработанное с использованием языка программирования Kotlin, ориентировано на мобильные устройства на платформе Android, что обеспечивает удобство и мобильность работы пользователей в реальном времени.

1.2 Формирования требований к программе

В рамках разработки приложения для системы доставки грузов, необходимо учитывать ряд системных требований, которые обеспечат корректную работу приложения, безопасность, удобство и эффективность взаимодействия между пользователями, а также надежную синхронизацию данных между мобильным приложением и сервером. Рассмотрим эти требования подробнее.

1. Роль Доставщика:

1.1. Просмотр списка открытых доставок. Доставщик должен иметь возможность просматривать список всех доставок, которые находятся в стадии выполнения. Система должна отображать актуальную информацию по каждой доставке (номер, адрес отправления, адрес получения, статус).

1.2. Создание новой доставки. Доставщик должен иметь возможность создать новую доставку, указав доступный для перевозки объем. Это действие инициирует создание записи в базе данных, которую будет отслеживать система.

1.3. Просмотр информации по доставке. Доставщик должен иметь доступ к подробной информации о каждой доставке, включая статус доставки, начальную и конечную точки, список грузов и т.д.

1.4. Добавление груза в доставку с помощью QR-кода. Доставщик должен иметь возможность сканировать QR-код, чтобы добавить груз в соответствующую доставку. Это может включать сканирование уникальных

кодов товаров и привязку их к конкретной доставке для упрощения учета и отслеживания.

2. Роль Менеджера:

2.1. Выбор склада для приемки грузов. Менеджер должен иметь возможность выбрать склад для приемки грузов.

2.2. Принятие груза с помощью QR-кода. Менеджер должен иметь возможность принять груз на складе, используя QR-код. Это действие также должно автоматически фиксировать груз в системе и обновлять его статус.

3. Обмен данными с сервером:

3.1. Приложение должно взаимодействовать с сервером через HTTP REST API, что позволяет обновлять и получать данные о доставках, складах и грузах. API будет обрабатывать запросы на добавление, изменение, удаление и получение информации.

1.3 Анализ технических требований

Для обеспечения стабильной работы системы, безопасности данных и эффективного взаимодействия между пользователями и сервером необходимо четко определить используемые технологии, инструменты разработки, а также требования к производительности и масштабируемости. В этом разделе рассматриваются требования к платформе Android, выбору языков программирования, архитектуре серверной части и взаимодействию между компонентами через REST API.

1. Мобильное приложение:

Платформа: Android 8.0 (API 26) и выше.

Язык программирования: Kotlin.

Инструменты разработки: Android Studio.

Библиотеки: Retrofit для работы с HTTP REST API, Gson для парсинга JSON, ZXing для работы с QR-кодами.

2. Сервер:

Язык серверной разработки: JS (Nestjs).

База данных: Реляционная база данных PostgreSQL.

API: REST API, с поддержкой стандартных HTTP методов (GET, POST, PUT, DELETE).

Безопасность: HTTPS для всех API-запросов, аутентификация через JWT.

2. Разработка алгоритмов

2.1 Разработка моделей

На серверной части проекта были разработаны основные модели данных, отражающие ключевые сущности системы доставки грузов, такие как Доставка, Груз, Пользователь (с ролями Доставщика и Менеджера), Склад. Эти модели данных стали основой для построения схемы базы данных (рисунок 1), которая обеспечивает хранение и эффективную обработку информации о доставках, грузах и пользователях. Разработка модели данных включала в себя определение взаимосвязей между сущностями, а также описание атрибутов и ограничений для каждой из них, что позволило обеспечить целостность и консистентность данных при работе с приложением.

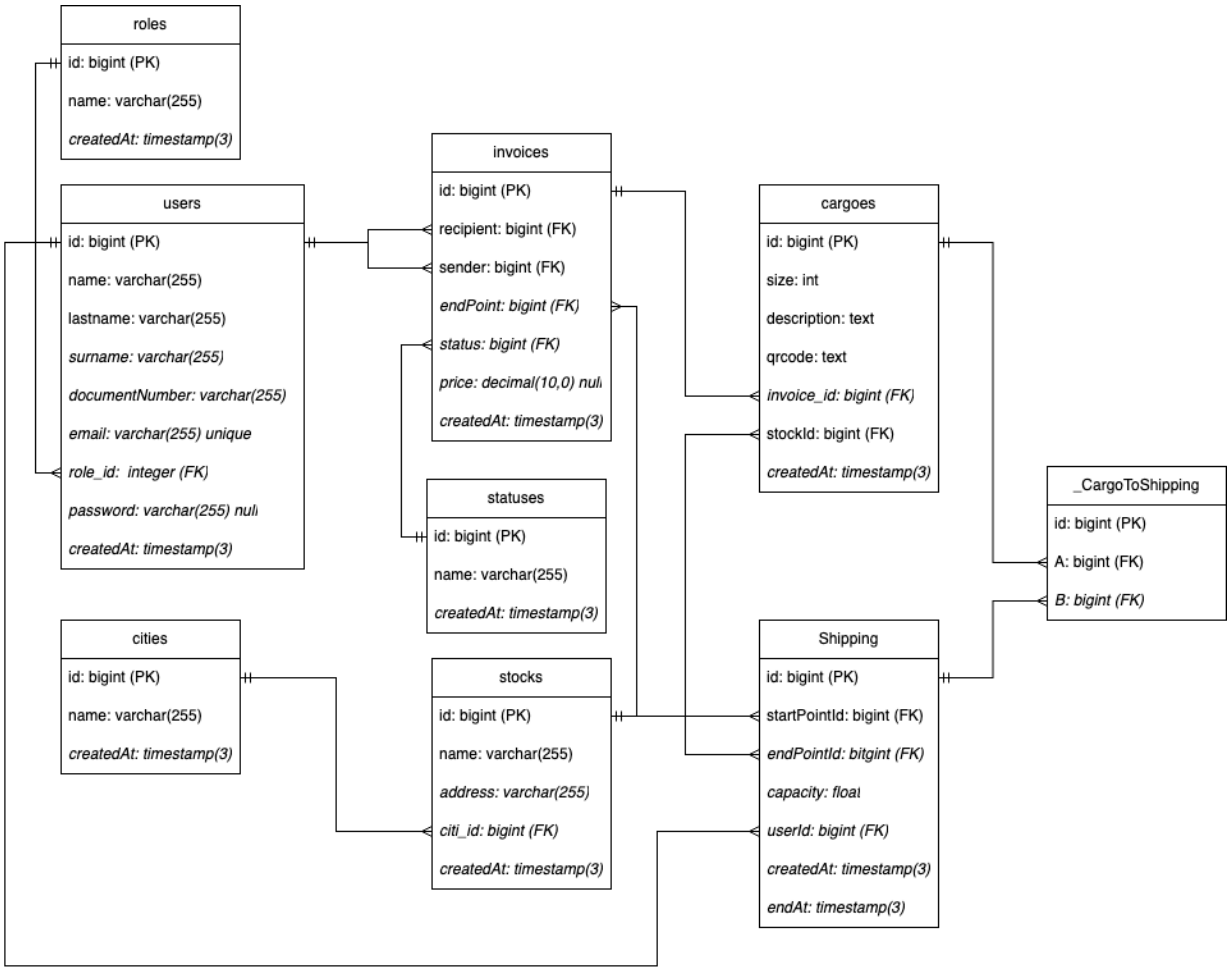

Рисунок 1 – Физическая схема базы данных.

На основе анализа предметной области, в проектируемой информационной системе предполагается две основные группы пользователей: курьеры, которые оформляют доставку грузов, и менеджеры, отвечающие за управление грузами на складах. На рисунке 2 представлена схема вариантов использования с двумя типами пользователей. Различия в вариантах использования связаны с особенностями функционала для каждой группы, а также с различием прав доступа к определенным функциям системы в зависимости от роли пользователя.

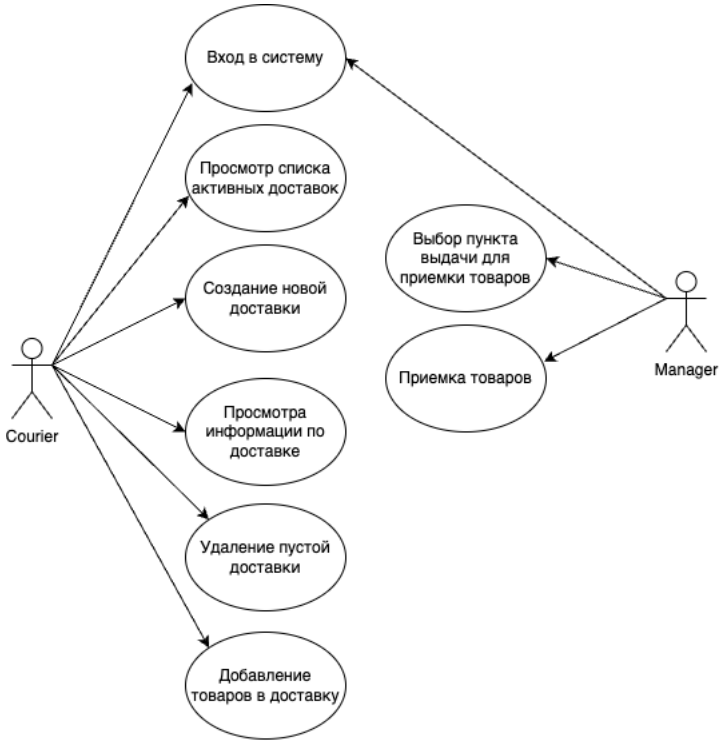

Рисунок 2 – Диаграмма вариантов использования.

На основе построенной схемы базы данных и рассмотрения вариантов использования приложения можно выделить основные алгоритмы, требующие разработки:

1. Алгоритм входа в приложение;
2. Для курьера:
   2.1. Алгоритм работы со страницей просмотра списка активных доставок;
   2.2. Алгоритм создания доставки;
   2.3. Алгоритм добавления новой доставки;
   2.4. Алгоритм просмотра информации о доставке;

2.5. Алгоритм добавления груза в доставку.

3. Для менеджера:

3.1.1. Алгоритм выбора склада для приема товара;

3.1.2. Алгоритм приема товара.

1.2 Разработка алгоритмов курьера

Разработка алгоритмов для курьера в системе доставки грузов является важным аспектом для обеспечения эффективного и своевременного выполнения доставок. Алгоритмы курьера должны учитывать различные факторы, такие как оптимизация маршрутов, взаимодействие с системой для получения информации о доставках, добавление и отслеживание грузов, а также обработка данных с помощью QR-кодов.

Целью данного раздела является описание алгоритмов, которые курьер использует для выполнения своих задач, включая взаимодействие с приложением и сервером, получение актуальной информации о доставках, создание новых заказов и добавление грузов в доставку. Особое внимание уделяется оптимизации работы курьера, минимизации времени на выполнение операций и повышению точности в учете и доставке товаров.

Процесс разработки алгоритмов включает в себя определение последовательности действий, обработку данных в реальном времени и создание логики взаимодействия с пользователем и системой. В результате разработки этих алгоритмов обеспечивается надежная и эффективная работа курьера в системе, что напрямую влияет на качество и скорость выполнения доставок.

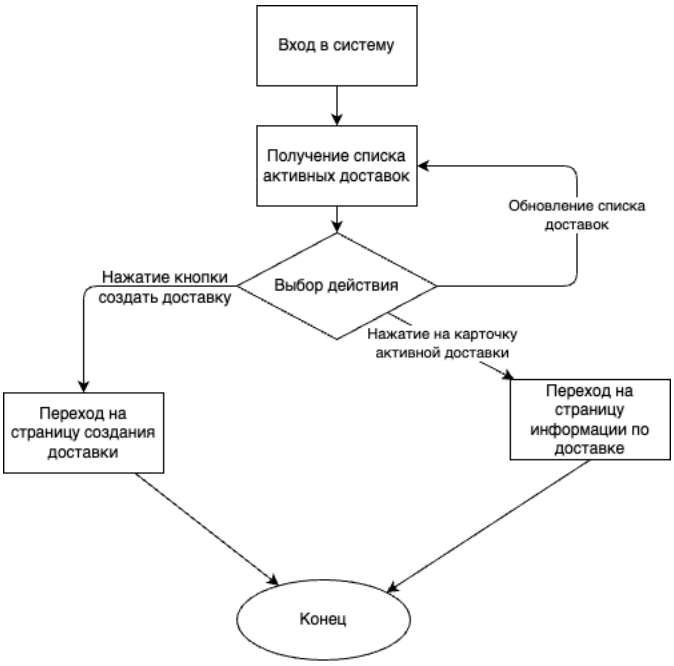

Рисунок 3 – Алгоритм главного экрана курьера

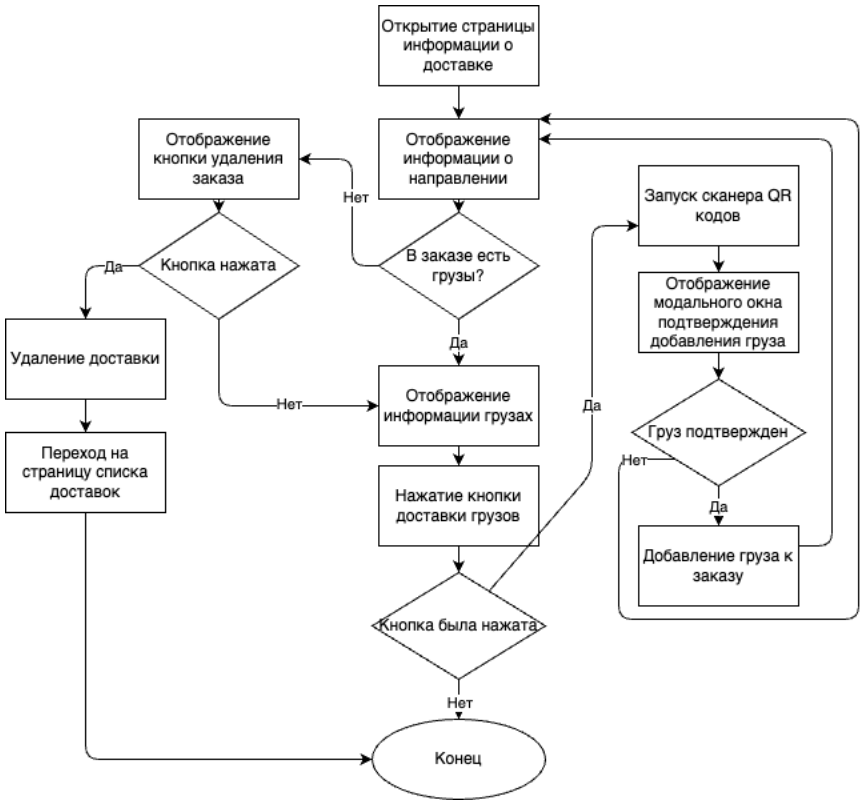

Рисунок 4 – Алгоритм отображения информации о доставке

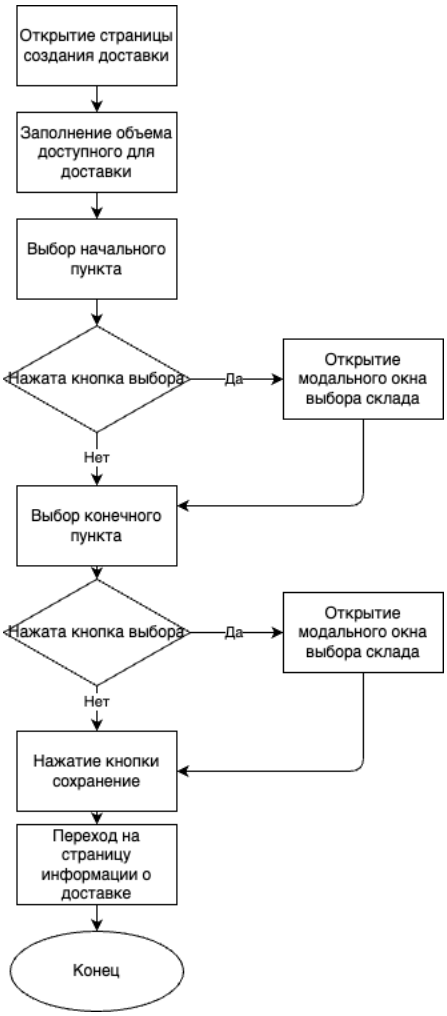

Рисунок 5 – Алгоритм создания новой доставки

2.3 Разработка алгоритмов менеджера

Разработка алгоритмов для роли менеджера в системе доставки грузов направлена на обеспечение эффективного управления процессами приемки и учета грузов на складах, а также взаимодействия с курьерами и отслеживания состояния доставок. Алгоритмы для менеджера должны учитывать особенности работы с большими объемами данных, такие как выбор склада для приемки, контроль за поступлением и учетом товаров, а также взаимодействие с системой через сканирование QR-кодов для автоматической регистрации грузов.

Цель данной подглавы — описание алгоритмов, которые менеджер использует для выполнения своих ключевых задач. Включая выбор склада для приема, подтверждение доставки и автоматизированное управление информацией

о грузах. Алгоритмы должны обеспечивать надежность, точность и оперативность при обработке данных, минимизируя возможность ошибок и задержек в процессе учета.

Разработка алгоритмов для менеджера включает создание логики, которая позволяет эффективно организовать работу с данными и взаимодействие с другими пользователями системы, такими как курьеры. Это позволяет повысить уровень автоматизации процессов на складе и обеспечить корректное функционирование всей системы доставки грузов.

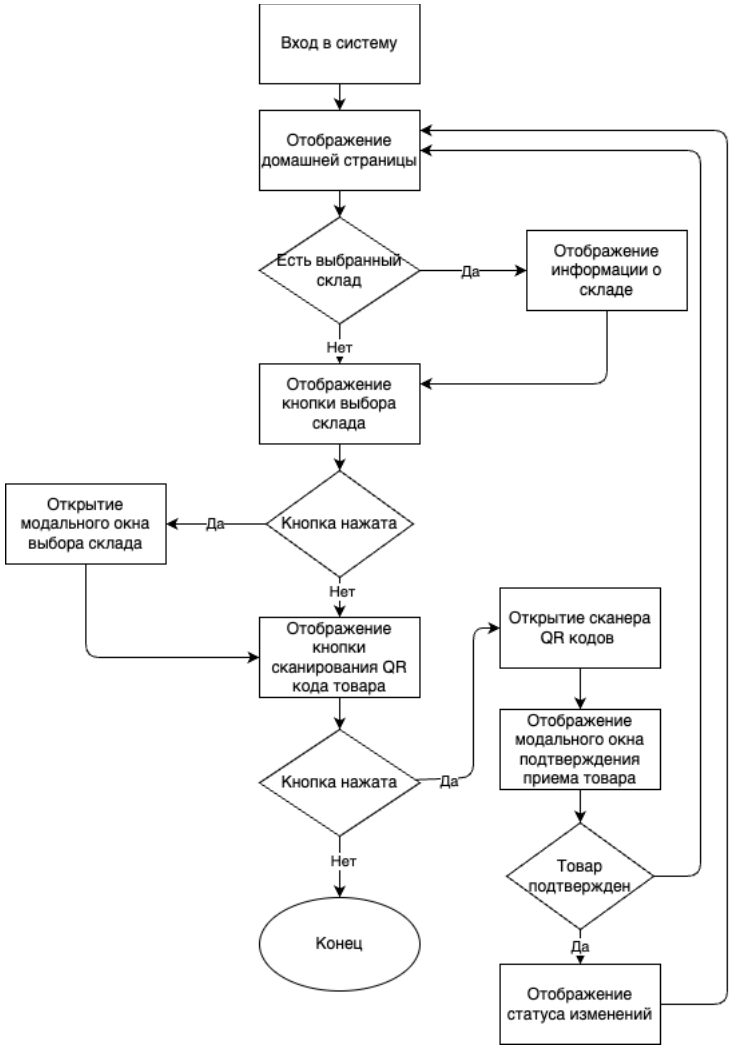

Рисунок 6 - Алгоритм работы кабинета менеджера

3 Руководство программиста

3.1 Описание архитектуры проекта

Проект представляет собой мобильное приложение для системы доставки грузов, разработанное с использованием языка Kotlin. Архитектура приложения разделена на два уровня пользователей:

Доставщик — может просматривать список открытых доставок, создавать новую доставку, добавлять груз в доставку с помощью QR-кода.

Менеджер — управляет складом, может принять груз с использованием QR-кода.

Приложение взаимодействует с сервером через HTTP Rest API для получения данных о доставках и складах.

Структура приложения

Проект использует стандартную архитектуру MVVM (Model-View-ViewModel), что способствует разделению логики и интерфейса, улучшает тестируемость и поддерживаемость кода.

Model — содержит данные (например, информация о доставках, складах).

View — интерфейс для взаимодействия с пользователем (например, активности и фрагменты).

ViewModel — управляет данными, полученными от модели, и обрабатывает логику отображения данных на UI.

Для взаимодействия с сервером используется библиотека Retrofit, а для работы с QR-кодами — библиотека ZXing.

3.2 Основные компоненты приложения

1. UI (Пользовательский интерфейс):
    1.1. Содержит экраны приложения и ViewModel для данных экранов.
    1.2. Интерфейс разделен на две роли: "Доставщик" и "Менеджер".
2. Network Layer:

2.1. Взаимодействует с сервером для получения данных о доставках, складах и грузе через HTTP Rest API.

2.2. Использует библиотеку Retrofit для упрощения запросов и обработки ответов от сервера.

3. Data Layer:

3.1. Репозитории для работы с данными, получаемыми с сервера.

3.2. Модели данных, представляющие информацию о доставках, складах и грузах.

4. QR Code Scanner:

4.1. Используется для сканирования QR-кодов, которые используются для добавления грузов в доставку и приемки на складе.

3.3 Основные классы

IpServerManager, StockInfoManager, TokenManager – классы реализующие механизм DataStore Preference, сохраняют конфигурацию в виде «ключ-значение» в кэше приложения и ползволяют получить к ней доступ после перезапуска приложения.

NetworkModule – объект предоставляющий собой модуль для интеграции Dependency Injection для сервисов Retrofit.

AuthInterceptor – класс, выполняющий добавление к запросам Retrofit заголовок авторизации jwt. Так же предоставляет аннотацию для указания в интерфейсах сервисов что запрос должен быть подписан токеном.

BaseUrlInterceptor – класс, выполняющий замену url в запросах, служащий для обновления из DataStore.

AppDestination – класс, предоставляющий список доступных роутов в приложении.

BaseViewModel – класс в котором реализован метод safeApiCall, который предоставляет безопасный вызов к серверу и выполняет обработку ошибок. Класс является наследником для всех ViewModel, используемых в приложении.

RequestResult – класс, описывающий возможные варианты результата вызовов safeApiCall.

StockListViewModel – ViewModel-класс для загрузки и обработки списка складов.

ShippingListViewModel – ViewModel-класс для загрузки и обработки списка доставок.

AuthViewModel – ViewModel-класс для обработки входа в приложение.

CreateShippingViewModel – ViewModel-класс для обработки создания доставки.

ShippingInfoViewModel – ViewModel-класс для обработки отображения информации о доставке и манипуляции с ней.

AcceptCargoViewModel – ViewModel-класс для обработки действия принятия товара менеджером.

StockSelectScreenViewModel – ViewModel-класс для обработки сохранения информации о выбранном складе для приемки товара.

3.4 Основные технологии и библиотеки

Kotlin — основной язык разработки.

Retrofit — для взаимодействия с сервером через HTTP.

DataStore Preferences — для хранения локальных данных (если требуется).

ZXing — для сканирования QR-кодов.

Jetpack ViewModel и LiveData — для управления состоянием UI.

3.5 Инструкции по настройке и запуску проекта

Установите Android Studio (если еще не установлено).

Клонируйте репозиторий с проектом:

git clone <URL>

Откройте проект в Android Studio.

Синхронизируйте проект с Gradle (если необходимо).

Настройте зависимости в файле build.gradle, если серверная часть требует настройки (например, базовый URL для API).

Запустите приложение на эмуляторе или реальном устройстве Android.

3.6 Взаимодействие с сервером

Для взаимодействия с сервером приложение использует Retrofit. Основные эндпоинты:

POST /auth/login – авторизация в системе.

GET /stocks – получение списка складов.

POST /cargo/acceptCargo – прием товара на склад.

GET /shipping — получение списка доставок.

POST /shipping — создание новой доставки.

GET /shipping/{id} – получение информации по доставке.

POST /shipping/{id}/add-cargo – добавление груза в доставку.

DELETE /shipping/{id} – удаление доставки.

Пример сервиса с использованием Retrofit:

```
interface ShippingService {
   @AuthRequired
   @GET("shipping")
   suspend fun getShippings(
      @QueryMap query: Map<String, String>
   ): Response<ShippingList>

   @AuthRequired
   @GET("shipping/{id}")
   suspend fun getShipping(@Path("id") id: Int): Response<ShippingOne>

   @AuthRequired
   @POST("shipping")
   suspend fun createShipping(@Body body: CreateShippingDto): Response<ShortShippingOne>
```

```
    @AuthRequired
    @POST("shipping/{id}/add-cargo")
    suspend fun addCargoToShipping(@Path("id") id: Int, @Body addCargoToShippingDto:
AddCargoToShippingDto): Response<ShippingOne>

    @AuthRequired
    @DELETE("shipping/{id}")
    suspend fun removeShipping(@Path("id") id: Int): Response<ShortShippingOne>
}
```

3.7 Описание взаимодействия с QR-кодом

Для сканирования QR-кодов используется библиотека ZXing. В приложении реализован сканер, который позволяет доставщику и менеджеру сканировать QR-коды для добавления грузов в доставку или приемки грузов на складе.

Пример кода для интеграции с ZXing:

```
val context = LocalContext.current

    val scannerLauncher = rememberLauncherForActivityResult(
        contract = ActivityResultContracts.StartActivityForResult()
    ) { result ->
        if (result.resultCode == Activity.RESULT_OK) {
            val data = result.data
            val qrCodeResult = data?.getStringExtra("SCAN_RESULT")
            qrCodeResult?.let {
                val data = parseQRCode<CargoQrData>(it)
                if (data != null) {
                    onScanned(data)
                }
            }
        }
    }
```

```
fun scanCargo() {
    val intent = Intent(context, CaptureActivity::class.java)
    scannerLauncher.launch(intent)
}
```

4 Руководство пользователя

4.1 Руководство для всех пользователей

4.1.1 Вход в систему

Вход в систему выполняется путем ввода email и пароля. Заполнение обоих полей является обязательным. После заполнения полей нужно нажать кнопку «login» (Рисунок 7). Если авторизация в системе была успешная, то пользователь будет перенаправлен на главную страницу в зависимости от роли.

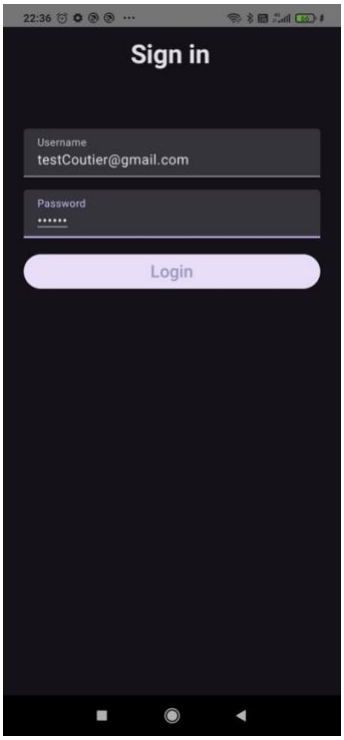

Рисунок 7 – Заполнение полей входа.

Если в процессе в процессе входа возникли какие-либо ошибки, то пользователю будет выведено сообщение ошибки под формой входа (Рисунок 8).

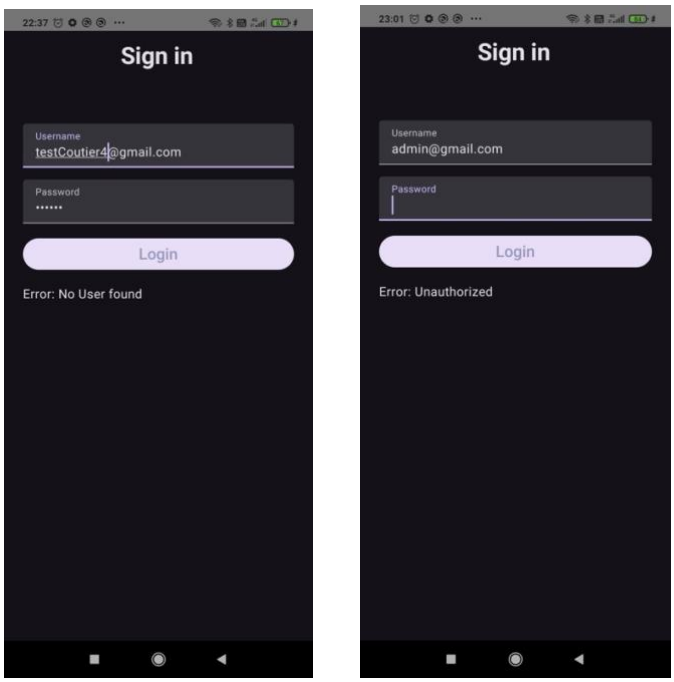

Рисунок 8 – Ошибка «Пользователь не найден в системе»(слева), Ошибка «Некорректные данные для входа»(справа)

4.1.2 Ошибка подключения к серверу

Приложение работает по принципе клиент-сервер, то есть все данные для работы приложение получает с сервера путем HTTP-запросов. Если в процессе работы сервер станет недоступен, то пользователю откроется окно с предложением указать другой адрес сервера (Рисунок 9)/. Пользователь может указать другой адрес сервера или закрыть окно и попробовать повторить выполнить действие.

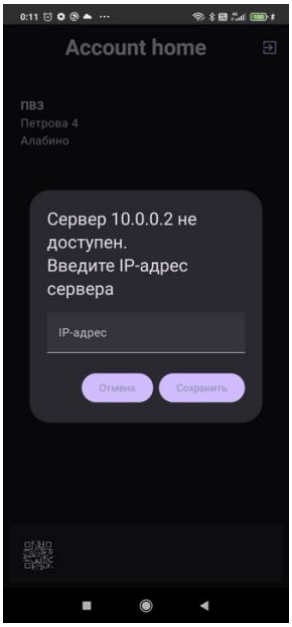

Рисунок 9 – Окно ввода нового адреса сервера.

4.2 Руководство для менеджеров

4.2.1 Выбор склада для приема грузов

При входе в систему с ролью пользователя менеджер открывается главный экран личного кабинета менеджера (рисунок 10). На данном экране нам предоставляется возможность выбрать, на каком складе мы сейчас находимся и начать приемку грузов используя кнопку-qrcode. Приемку товаров можно начинать только после того, как будет выбран склад. Чтобы выбрать склад нужно нажать на текстовую кнопку «изменить пункт приема товаров».

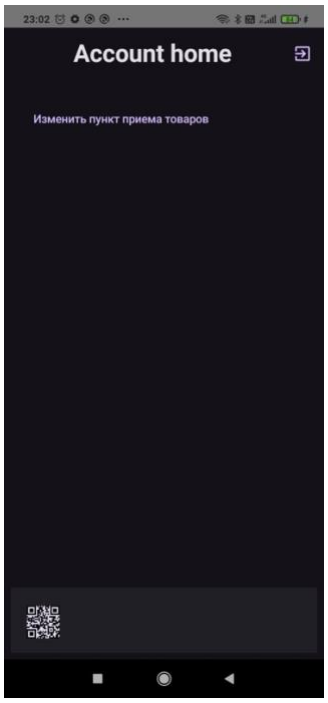

Рисунок 10 – Главный экран без выбора склада.

После чего откроется модальное окно (рисунок 11), в котором будет отображен список складов и текстовое поле для фильтрации. Фильтрация может осуществляться по следующим параметрам: город, название, адрес. При наборе текста в поле ввода программа будет автоматически обновлять список.

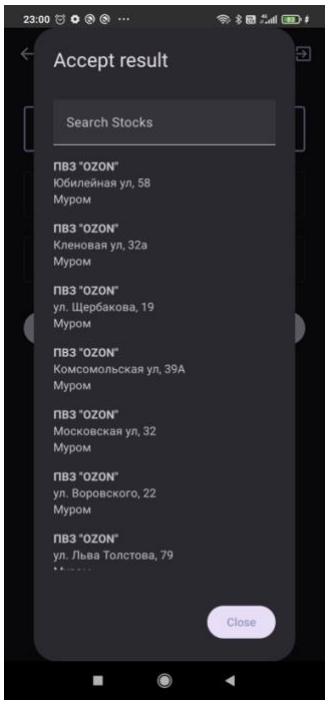

Рисунок 11 – Окно выбора склада для приема гурзов.

Для выбора склада нужно нажать на него. После чего сведения о выбранном складе отобразятся на главном экране (рисунок 12). Так же отобразится иконка для сканирования qrcode для приемки товаров.

Рисунок 12 – Главный экран с информацией о выбранном складе.

4.2.2 Прием грузов

Основная задача менеджера принимать грузы на склад. Чтобы выполнить прием груза необходимо нажать на кнопке с изображением qrcode на главном экране приложения. Откроется сканер qr-кодов. Отсканировав qr-код на грузе менеджеру будет отображено модальное окно с подтверждением о приемке груза на склад (Рисунок 13).

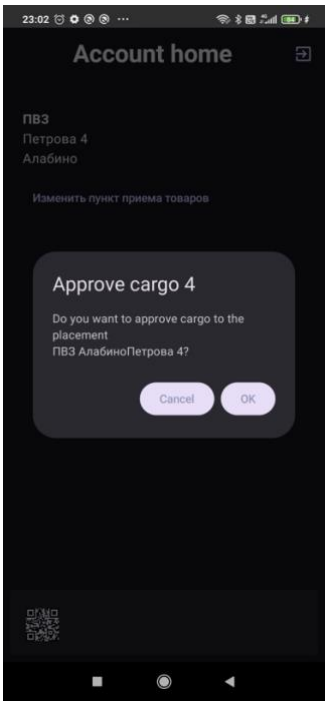

Рисунок 13 - Окно подтверждения приема груза.

После нажатия на кнопку «ок» будет отправлен запрос на сервер с информацией о принимаемом грузе. В результате выполнения запроса менеджеру будет выведено модальное окно с результатами выполнения запроса (рисунок 14). Если в результате приема груза будет завершен заказа или доставка, то менеджер будет оповещен об этом в данном модальном окне.

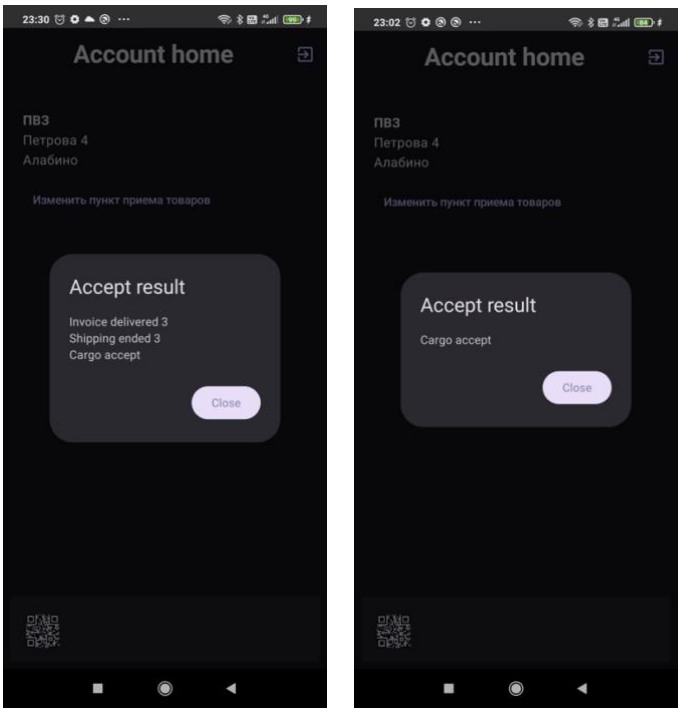

Рисунок 14 - Окно с информацией о завершении доставки и заказа после приема груза (слева), Окно с информацией об успешном приеме груза (справа).

Если после выполнения запроса к серверу будет получена ошибка, то она будет выведена в модальном окне (рисунок 15).

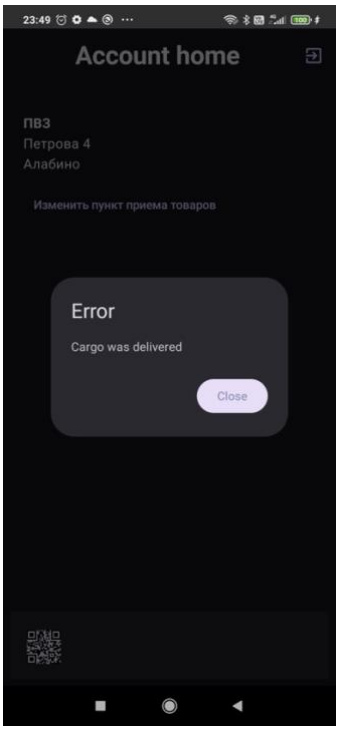

Рисунок 15 - Окно с информацией об ошибке при приеме груза.

4.3 Руководство для доставщиков

4.3.1 Работа со списком активных доставок

Войдя в приложение с ролью курьера, приложение загрузит список активных доставок (рисунок 16). Список доставок предоставляет краткую информацию о доставке: направление, заполнение объема, дата создания. Чтобы просмотреть подробную информацию нужно нажать на кнопку продолжить в карточке. Так же на главной странице курьера отображается кнопка для создания новой доставки.

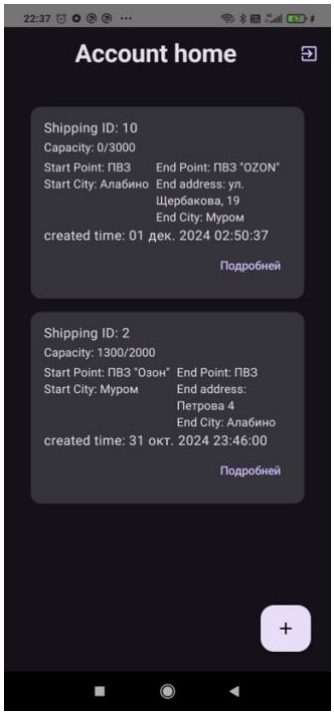

Рисунок 16 – Главный экран курьера.

4.3.2 Создание новой доставки

Чтобы создать новую доставку необходимо на главном экране курьера нажать кнопку с изображением плюса, после чего пользователь будет перенаправлен на экран заполнения информации по доставке (рисунок 17).

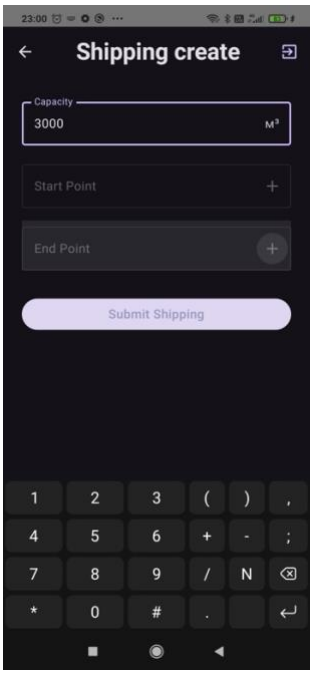

Рисунок 17 – Заполнение доступного объема для доставки.

Заполнение информации состоит из:

1. Заполнение объема, который курьер может взять для доставки;
2. Выбора склада, из которого будет осуществляться доставка;
3. Выбор склада, в который будет осуществляться доставка.

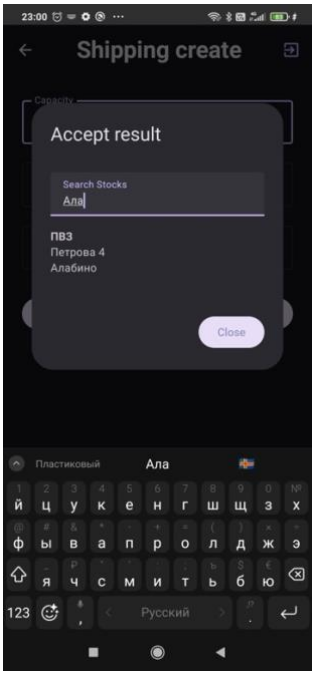

Рисунок 18 – Выбор склада для начальной и конечной точек.

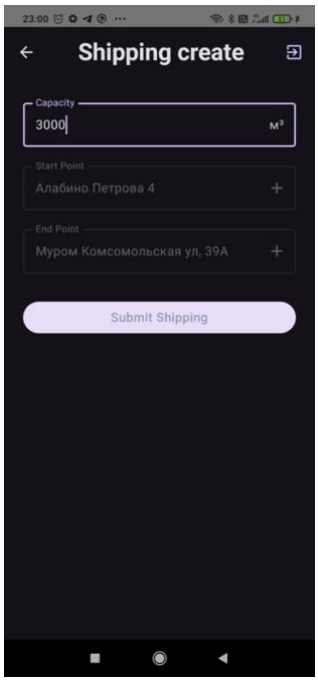

Рисунок 19 – Полностью заполненная форма для создания доставки.

После заполнения всех полей и нажатия кнопки отправки, пользователь будет перенаправлен на экран с информацией о доставке.

4.3.3 Удаление пустой доставки

Если доставка была создана по ошибке и в нее еще не было добавлено ни одного груза, то данную доставку можно удалить. Удаление доступна на странице информации о доставке. Рядом с основной информацией о заказе будет отображаться кнопка в форме корзины (рисунок 20). При нажатии на нее доставка будет удалена, а пользователь перенаправлен на экран с списком активных доставок.

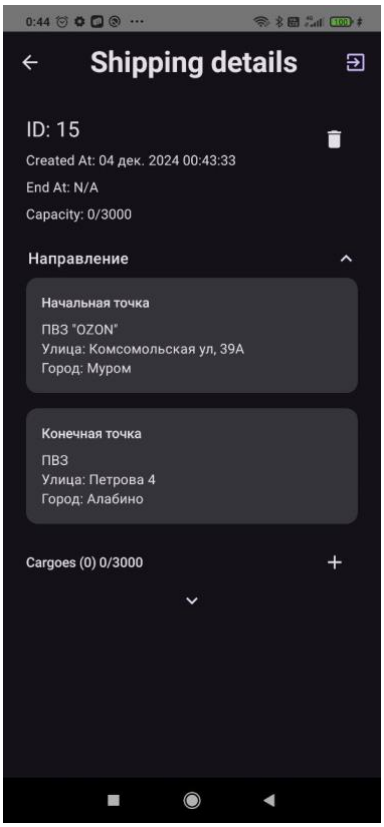

Рисунок 20 – Экран информации о доставке.

4.3.4 Добавление груза в доставку

Добавление груза в заказ осуществляется с экрана информации о заказе, нажав на значок плюс рядом с информацией о грузах откроется окно сканирования qr-кода. После сканирования qr-кода пользователю будет выведено модальное окно с подтверждением о добавлении груза в доставку (рисунок 21). Подтвердив добавление груз будет прикреплен к доставке и отобразится в списке грузов на экране информации о доставке (рисунок 22).

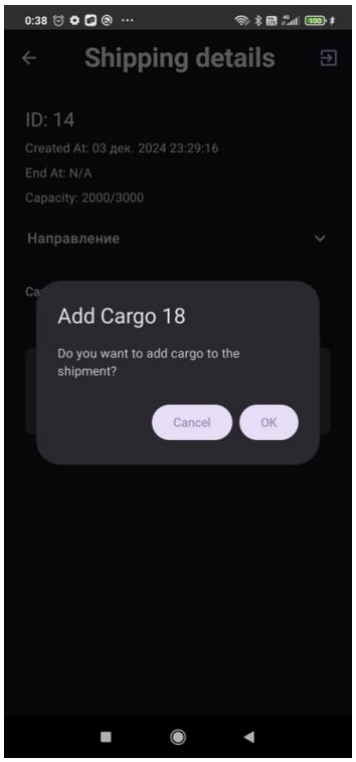

Рисунок 21 – Окно подтверждения добавления груза в доставку.

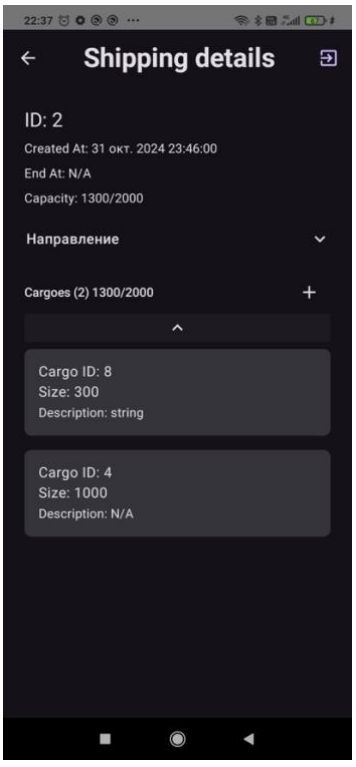

Рисунок 22 – Список грузов на экране информации о доставке.

Заключение

В ходе выполнения курсовой работы была успешно разработана распределённая информационная система для учета грузоперевозок в форме мобильного приложения. Приложение предоставляет функционал для двух ролей пользователей: курьеров и менеджеров. Реализация осуществлялась с использованием современных технологий, включая язык программирования Kotlin для платформы Android и REST API для взаимодействия с сервером.

В процессе разработки выполнены следующие этапы:

1. Анализ технического задания, выявление и формирование требований к системе;
2. Проектирование моделей данных и архитектуры приложения;
3. Реализация функционала для обеих ролей пользователей с учетом их особенностей;
4. Проведение функционального тестирования, подтверждающего корректность работы приложения.

Приложение предоставляет удобный интерфейс для управления процессами доставки, включая создание, редактирование и обработку заказов. Использование QR-кодов позволило существенно упростить и ускорить операции по добавлению и учету грузов, что минимизирует человеческий фактор. Реализация разделения ролей улучшает безопасность и организацию процессов.

Полученные результаты демонстрируют, что разработанная система удовлетворяет требованиям современного рынка логистики и может быть эффективно использована для автоматизации процессов учета и управления грузоперевозками.

Код проекта предоставлен в github: https://github.com/kayn23/small_delivery_mobile_terminal

Список литературы

1. Семакова, А. Введение в разработку приложений для смартфонов на ОС Android. — Москва: Национальный открытый университет «ИНТУИТ», 2016. — 275 с.

2. Колисниченко, Д. Н. Программирование для Android 5. Самоучитель. — Санкт-Петербург: БХВ-Петербург, 2015. — 303 с.

3. Дейтел, П., Дейтел, Х., Уолд, А. Android для разработчиков. — 3-е изд. — Санкт-Петербург: Питер, 2016. — 672 с.

4. Гриффитс, Д., Гриффитс, Д. Head First. Программирование для Android. — 2-е изд. — Санкт-Петербург: Питер, 2018. — 912 с.