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

Кафедра «Прикладная математика и информатика»
(наименование)

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

Разработка программного обеспечения
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка мобильного приложения для курьерской службы»

Обучающийся Д.А. Азямов
(Инициалы Фамилия) (личная подпись)

Руководитель к.т.н., доцент, О.В. Аникина
ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

Тема: «Разработка мобильного приложения для курьерской службы»

Объектом работы являются принципы функционирования и разработки мобильных приложений, предметом – разработка мобильного приложения курьерской службы.

Структура работы представлена введением, тремя главами основной части, заключением, списком использованных источников, и тремя приложениями.

В первой главе работы проводится описание предметной области, моделирование бизнес-процессов предметной области. Также проводится обзор аналогов и выбор средств проектирования и разработки, в результате чего для разработки приложения был выбран фреймворк Android Studio, а в качестве субд SQLite. Также проведена постановка задачи на разработку системы, описаны ее основные функции.

Вторая глава посвящена разработке приложения. Проведено UML моделирование работы системы, позволившее определить основные функции и состав пользователей, после чего проведена разработка мобильного приложения.

В последней главе работы проведено тестирование разработанного приложения и описаны его основные функции.

Данная бакалаврская работа состоит из 31 рисунков, 3 таблиц, списка используемых источников из 24 источников на русском языке, 6 источников на иностранном языке.

Оглавление

Введение

В условиях цифровой трансформации нефтегазовой отрасли надежная и эффективная работа телекоммуникационной инфраструктуры становится критически важным фактором обеспечения бесперебойной деятельности предприятий. ООО «Газпромнефть Информационно-Технологический оператор» (Газпромнефть ИТО) играет ключевую роль в поддержании работоспособности систем связи, передачи данных и цифровой инфраструктуры для «Газпром нефти». Компания сталкивается с уникальными вызовами, такими как географическая распределенность объектов, суровые климатические условия и высокие требования к кибербезопасности, что обуславливает необходимость оптимизации бизнес-процессов и внедрения современных цифровых решений.

Объектом исследования в данной выпускной квалификационной работе (ВКР) выступает деятельность ООО «Газпромнефть ИТО» в сфере обеспечения телекоммуникационных и ИТ-услуг.

Предметом исследования являются бизнес-процессы компании, связанные с доставкой оборудования и документооборотом, а также возможности их оптимизации с помощью цифровых инструментов.

Целью бакалаврской работы является разработка мобильного приложения для курьерской службы.

Для достижения поставленной цели необходимо решить следующие задачи:

- изучить организационную структуру и ключевые процессы компании, выявив узкие места;
- провести моделирование бизнес-процессов «как есть» с использованием методологий IDEF0 и BPMN;
- сравнить текущие процессы с лучшими отраслевыми практиками (ITIL, ITSM);

- оценить техническую оснащенность компании и предложить решения по интеграции новых цифровых инструментов;
- разработать прототип мобильного приложения для оптимизации курьерской доставки документов с функционалом отслеживания статусов в реальном времени, формирования цифровых отчетов.

В работе применяются такие методы исследования как:

- анализа документации и регламентов компании;
- моделирования бизнес-процессов;
- сравнительного анализа с отраслевыми стандартами;
- проектирования пользовательских интерфейсов.

Практическая значимость исследования заключается в том, что внедрение предложенных решений позволит повысить прозрачность процессов через единую систему учета.

Научная новизна работы состоит в адаптации современных цифровых инструментов к специфике телекоммуникационной инфраструктуры нефтегазовой отрасли.

Выпускная квалификационная работа состоит из введения, трех глав, заключения и списка литературы. В первой главе рассматривается деятельность компании и анализируются текущие бизнес-процессы. Во второй главе выполняется проектирование системы с использованием UML-моделирования, разрабатывается мобильное приложение и личный кабинет администратора. описывается процесс тестирования, представляются результаты проверки работоспособности приложения.

Глава 1 Аналитическая часть

1.1 Описание предметной области

В современном мире, где мобильные технологии становятся всё более распространёнными, персональные компьютеры постепенно уступают место смартфонам и планшетам. Чтобы оставаться на связи с клиентами, компании активно разрабатывают мобильные приложения или адаптируют свои сайты для удобного просмотра с мобильных устройств.

Мобильное приложение — это специализированное программное обеспечение, созданное для работы на смартфонах и планшетах. Оно выполняет конкретные задачи и должно быть не только функциональным, но и простым в использовании. Если интерфейс окажется неудобным, пользователи вряд ли станут его скачивать [23].

Стоит различать мобильное приложение и адаптивную версию сайта. Адаптивный сайт — это упрощённая версия десктопного сайта, где элементы интерфейса увеличиваются для удобного отображения на небольшом экране. Дизайн при этом остаётся практически неизменным. В отличие от этого, мобильное приложение — это отдельный продукт, разработанный специально для смартфонов с учётом их возможностей. Хотя в нём могут сохраняться фирменные цвета и стиль, его структура и функционал полностью перерабатываются под мобильные устройства.

Мобильные приложения отличаются рядом важных особенностей:

- оптимизированный интерфейс – разработан специально для сенсорного управления, без использования мыши и клавиатуры;
- удобный дизайн – продуманный и интуитивно понятный, обеспечивающий комфортную работу пользователя;
- расширенные возможности – в отличие от мобильной версии сайта, приложение способно работать в фоновом режиме и даже без

подключения к интернету. Среди таких функций: push-уведомления, напоминания, геолокация и многое другое;

- хранение персональных данных пользователя, например, паспортные данные, номер телефона, адрес, номер медицинского полиса и другое;
- различные приложения могут отслеживать геопозицию, биологические ритмы и многие другие данные в зависимости от того, какое это приложение [1].

Мобильные приложения можно разделить на клиентские и корпоративные. Клиентские приложения предназначены для конечных пользователей и служат инструментом взаимодействия бизнеса с аудиторией. К ним относятся банковские сервисы, системы бронирования, интернет-магазины и программы лояльности. Корпоративные приложения используются внутри компаний для организации рабочих процессов и повышения эффективности сотрудников.

По способу разработки мобильные приложения бывают трех основных типов. Веб-приложения представляют собой адаптированные мобильные версии сайтов, которые просты в разработке, но обладают ограниченной функциональностью.

Гибридные приложения совмещают в себе элементы веб- и нативных решений, предлагая баланс между кроссплатформенностью и производительностью.

Нативные приложения разрабатываются отдельно для каждой операционной системы, обеспечивая максимальную производительность и полный доступ к функциям устройства, но требуют значительных временных и финансовых затрат на создание.

Выбор типа приложения зависит от конкретных бизнес-задач, целевой аудитории и доступного бюджета.

Гибридные приложения имеют более широкий функционал, нежели мобильные приложения и веб-сайты, например, пуш-уведомления, а также могут скачиваться на телефон пользователя через магазины приложений.

Разработка нативного приложения намного трудозатратнее, так как требует разработку под каждую операционную систему отдельно (iOS, Android, Windows Phone). Но отличается самой высокой функциональностью 8 среди других типов приложений, например, может использовать контакты, камеру, микрофон и так далее.

Программное обеспечение для смартфонов и цифровых помощников, созданное для операционных систем Android и iOS, может быть установлено на устройства уже на этапе производства, скачано из специализированных магазинов или доступно через интернет-браузеры.

При создании таких приложений разработчики используются язык программирования и разметки, среди которых Java, Swift, C# и HTML5 [3].

1.2 Моделирование бизнес-процессов

Нотация IDEF0 представляет собой методологию функционального моделирования и графическую нотацию, предназначенную для формализации и описания бизнес-процессов.

Проведем моделирование бизнес-процессов, которые будут происходить при взаимодействии с приложением курьерской службы, то есть составим диаграмму «КАК ДОЛЖНО БЫТЬ».

На рисунке 1 представлена контекстная диаграмма процесса доставки документа «как должно быть» в нотации IDEF0.

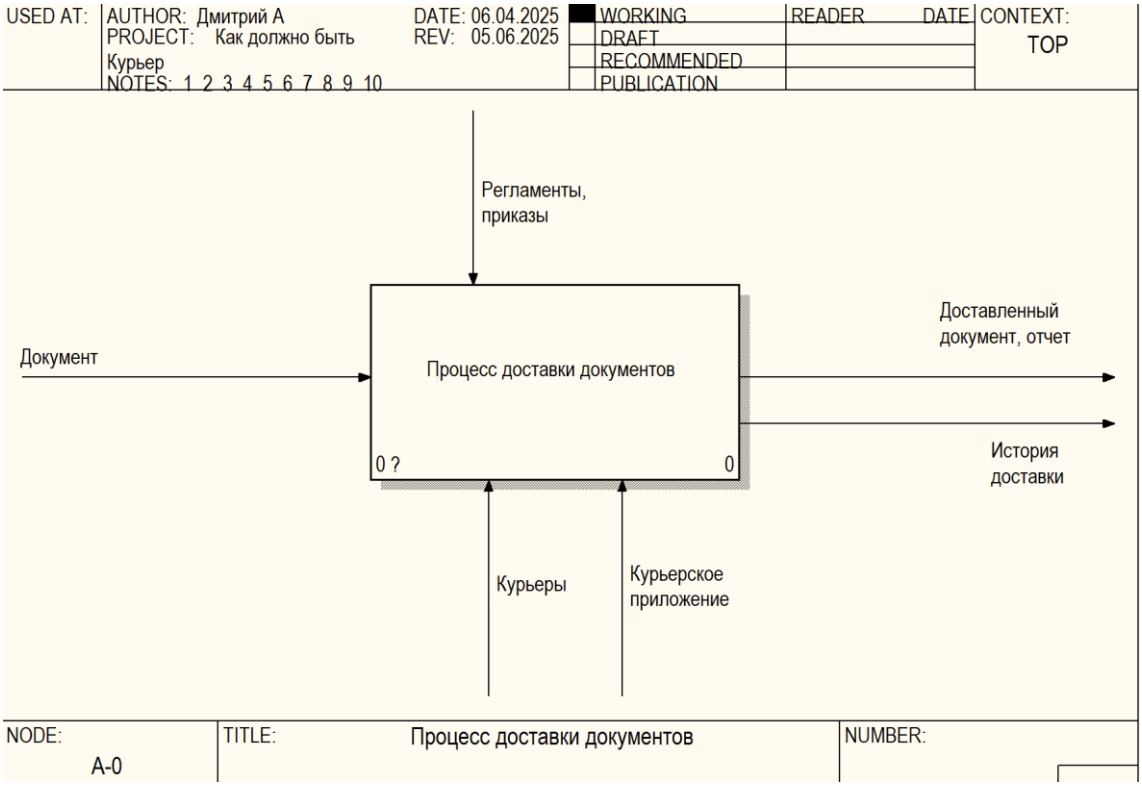

Рисунок 1 – Диаграмма процесса доставки документа «как должно быть»

Согласно контекстной диаграмме входными данными является:

– документ.

Механизмы:

– курьеры;
– курьерское приложение.

Управляющие воздействия:

– регламенты, приказы.

Выходные данные:

– доставленный документ, отчет;
– история доставки.

Декомпозиция контекстной диаграммы приведена на рисунке 2.

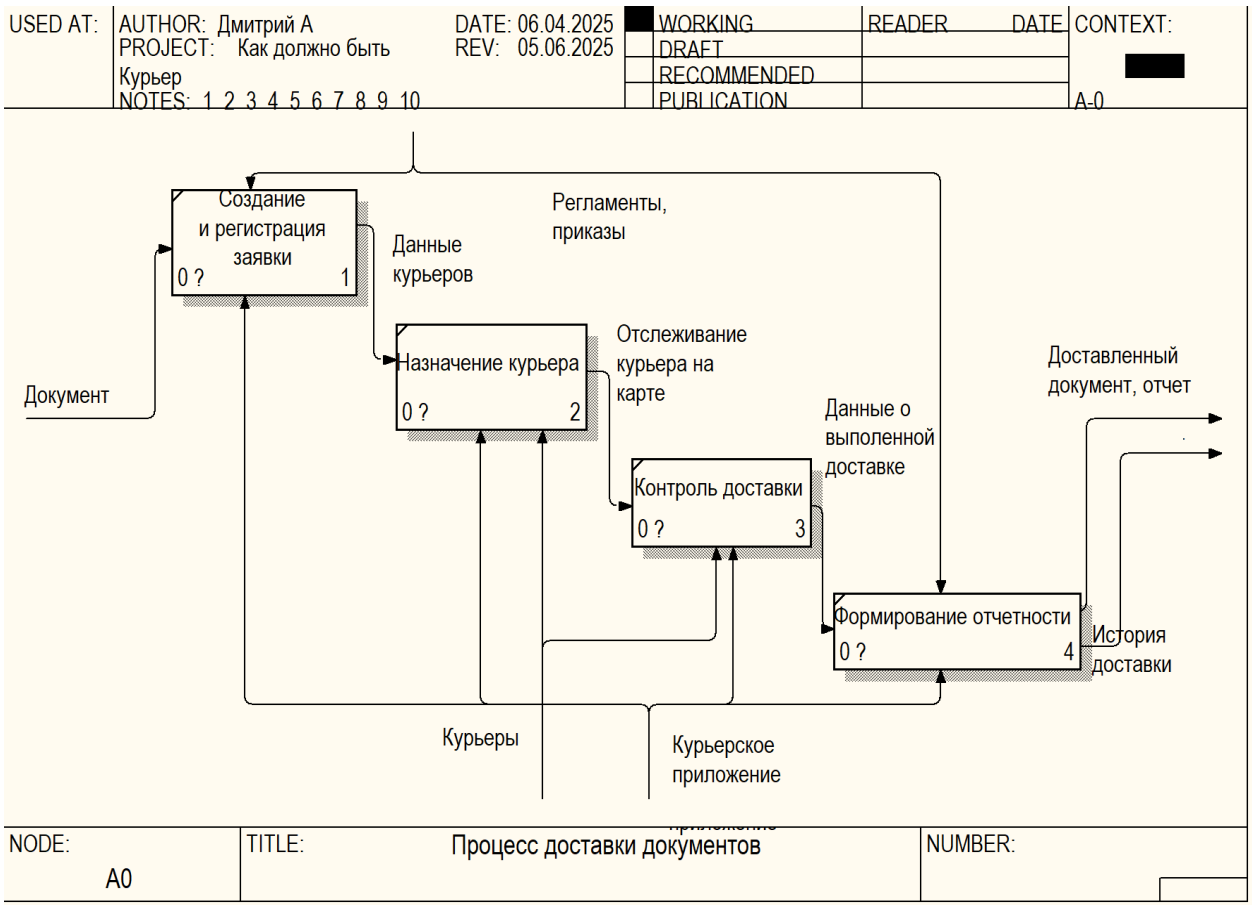

Рисунок 2 – Декомпозиция автоматизированного процесса

Согласно диаграмме декомпозиции, процесс начинается с создания и регистрации заявки на доставку документа. На этом этапе используются регламенты и приказы, а также данные о документе, который необходимо доставить.

После этого система назначает курьера, используя доступные данные о курьерах и их текущей загрузке. Назначенный курьер получает задание через курьерское приложение, а его перемещение отслеживается на карте в режиме реального времени.

Контроль доставки осуществляется на протяжении всего процесса, что позволяет своевременно выявлять и устранять возможные проблемы.

После успешного выполнения доставки система фиксирует данные о выполненной задаче, включая доставленный документ и отчет.

На завершающем этапе формируется отчетность на основе истории доставок, что позволяет анализировать эффективность работы курьеров и оптимизировать процесс в будущем.

1.3 Обзор аналогичных систем

На рынке существует достаточно большое количество приложений, близких по функционалу к разрабатываемому.

Проведем обзор аналогичных разрабатываемому приложению программ, а именно:

- «OKdesk»;
- «HubEx»;
- «Naumen».

Приложения будут сравниваться по функциональности управления заказами, включая создание, редактирование и отмену заказов, а также назначение курьеров вручную или автоматически с возможностью изменения статусов («В обработке», «В пути», «Доставлено»).

Важным аспектом является мониторинг и уведомления: приложения должны предоставлять список заказов с фильтрами и поддерживать push-уведомления для курьеров и пользователей.

Также будет оцениваться система отчетности, позволяющая анализировать статистику по выполненным заказам и экспортировать данные в форматы Excel или PDF.

Дополнительно учитывается возможность корректировки данных, например, редактирование заказов администратором в случае ошибок в адресе или других деталях.

В таблице 1 представлен сравнительный анализ существующих систем и сравнение с разрабатываемой.

Таблица 1 – Сравнительный анализ аналогичных систем

Функция	Okdesk	HubEx	Naumen	Разрабатываемая система
Создание заказов	+	+	+	+
Назначение курьеров	+	+	+	+
Отслеживание статусов	+	-	+	+
Мобильное приложение	+	+	-	+
Гибкая отчетность	-	-	+	+
Стоимость	Высокая	Средняя	Высокая	Оптимальная

В приложении Okdesk при входе встречает окно авторизации (Рисунок 3).

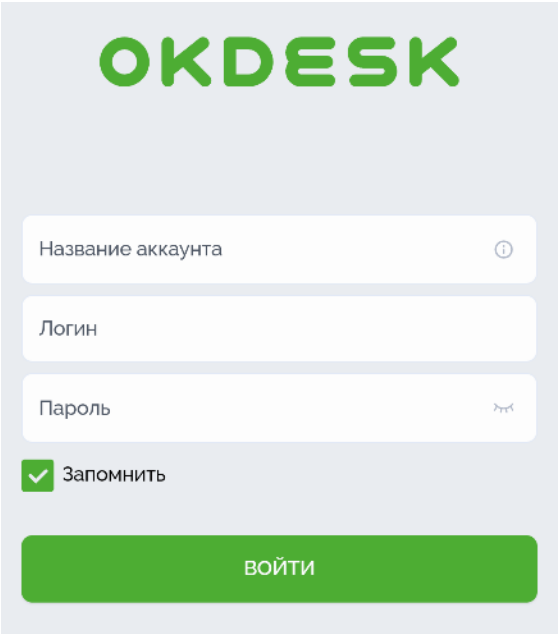

Рисунок 3 – Окно авторизации Okdesk

Название аккаунта, логин и пароль выдает администратор системы.

Далее после авторизации появляется окно заявок. Окно заявок представлено на рисунке 4.

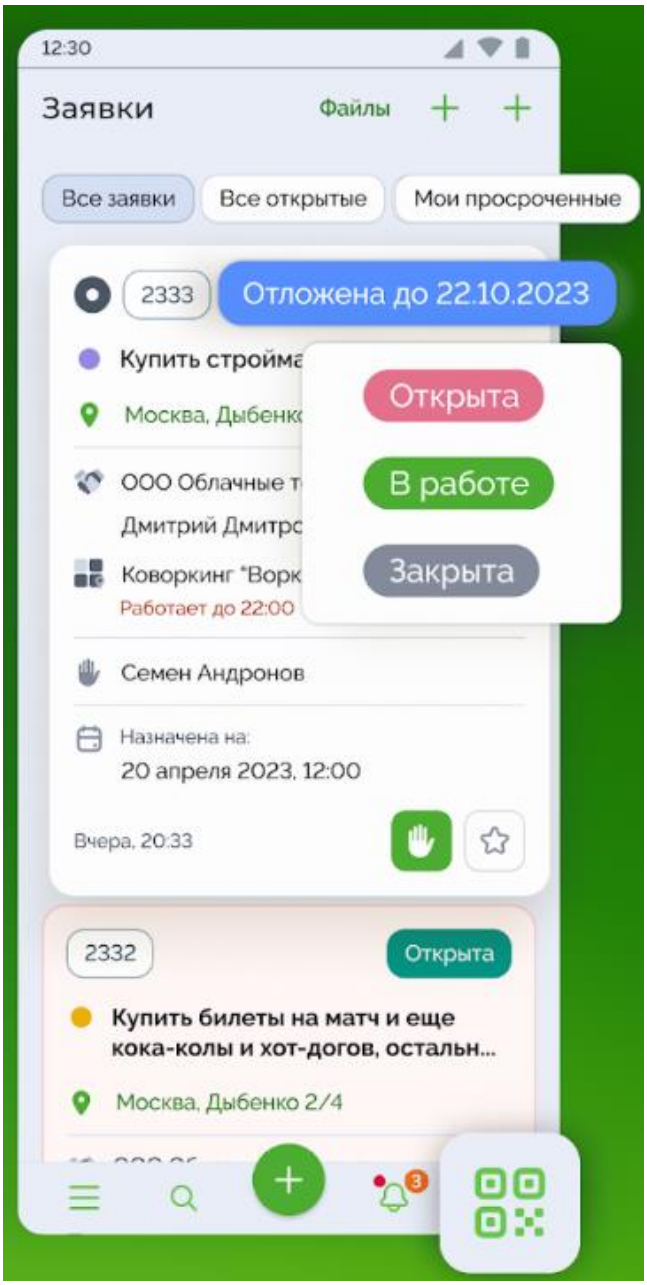

Рисунок 4 – Окно заявок Okdesk

В данном окне есть возможность просмотреть номер заявки, ее статус, адрес доставки, ответственного за выполнение работы и срок выполнения заявки, а также добавить их в избранное.

После выбора заявки открывается более подробная информация о ней, представленная на рисунке 5.

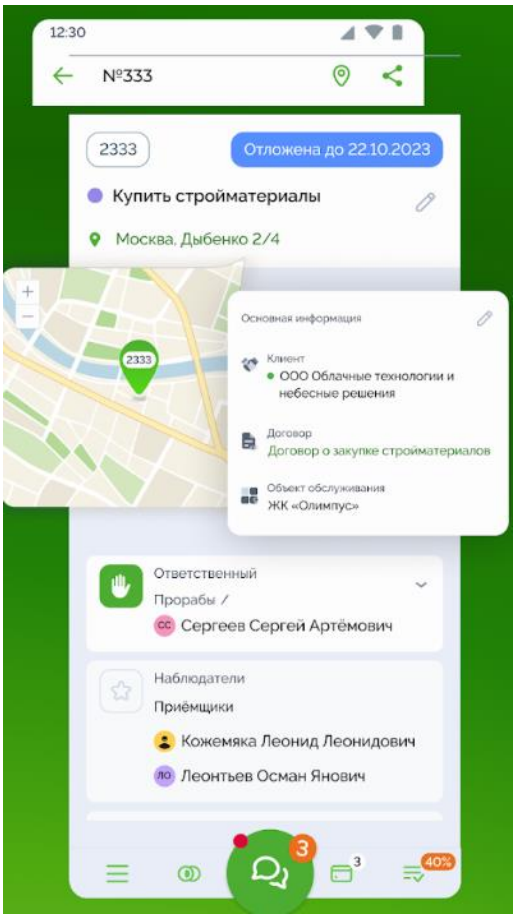

Рисунок 5 – Больше подробностей о заявке

При переходе в заявку пользователь имеет возможность просмотреть ответственного за заявку, номер заявки и просмотреть точку на карте куда следует доставить заказ.

Достоинствами приложения являются:

– удобный интерфейс;

– простота навигации, интуитивно понятное управление;

– просмотр задач, контактов, сделок и проектов в режиме реального времени;

– фильтрация по статусам, приоритетам, исполнителям;

– push-уведомления;

– поддержка двухфакторной аутентификации (2FA).

К недостаткам можно отнести:

– перегруженность интерфейса большим количеством элементов;

– зависания при плохом интернете;

– не всегда быстро обновляются данные.

В приложении HubEx при входе встречает окно авторизации. Есть 3 различных способа авторизации: через SMS-уведомление, с помощью логина и пароля и с помощью email, логина или домена организации (Рисунки 6-8).

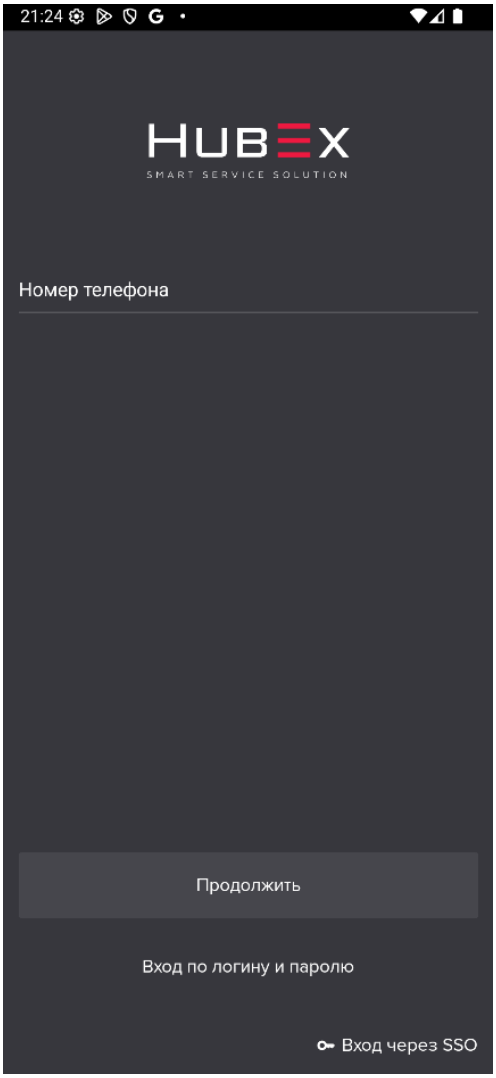

Рисунок 6 – Окно авторизации через SMS-уведомление

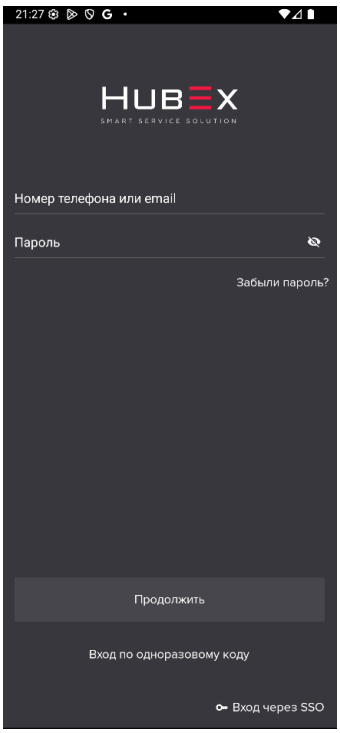

Рисунок 7 – Авторизация с помощью логина и пароля.

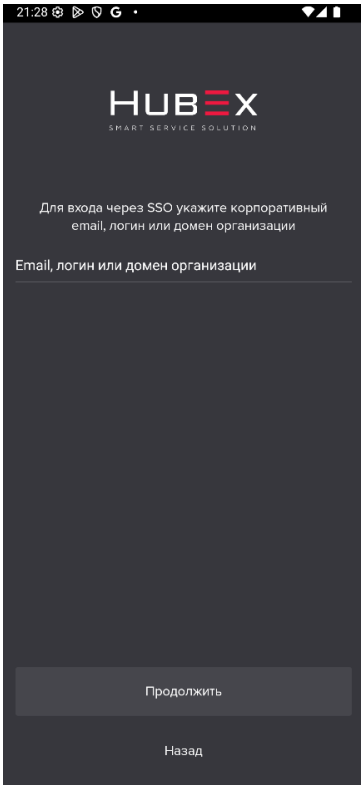

Рисунок 8 - Авторизация с помощью email, логина или домена организации

После авторизации можно увидеть окно с заявками (Рисунок 9).

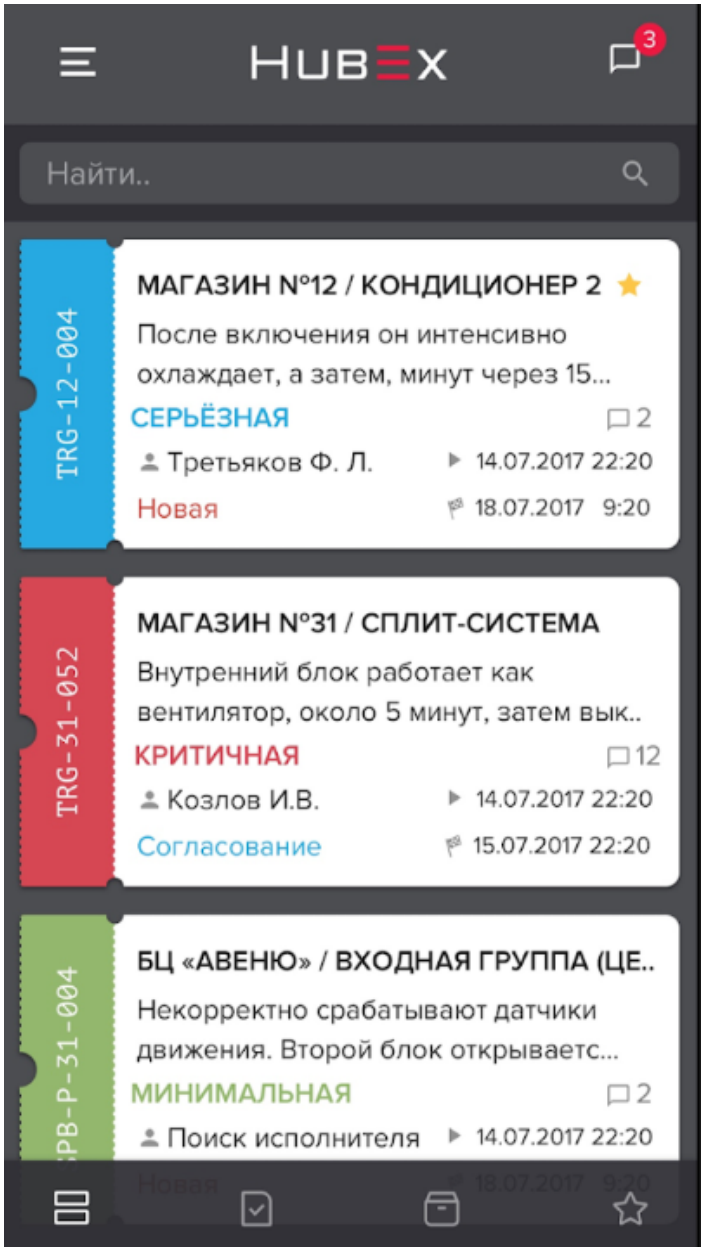

Рисунок 9 – Окно с заявками

При нажатии на заявку можно увидеть ее подробности (Рисунок 10).

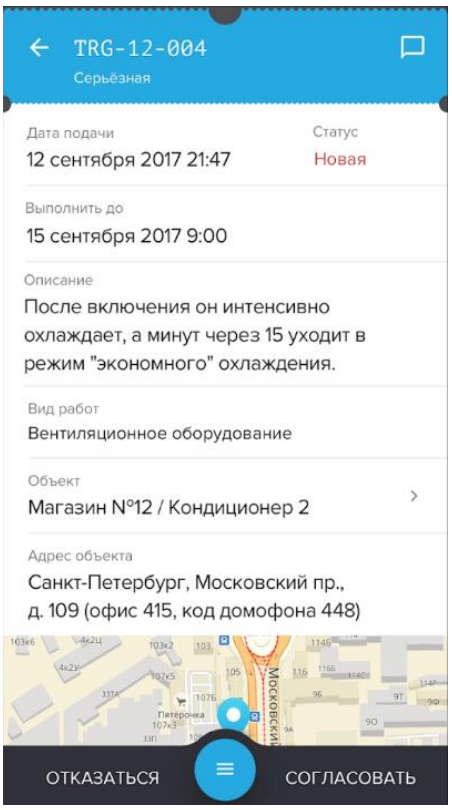

Рисунок 10 – Подробности заявки

Достоинствами приложения являются:

- минималистичный интерфейс;
- простота освоения;
- добавление, редактирование, фильтрация контактов.

К недостаткам можно отнести:

- нельзя создать полноценную заявку с этапами, статусами, историей изменений;
- не подходит для задач, где нужно контролировать этапы работы;
- при плохом соединении возможны зависания.

В приложении Naumen нет стандартного окна авторизации. Вместо этого вход в систему происходит автоматически, используя корпоративную учетную запись на рабочем компьютере. Затем открывается окно с заявками, представленное на рисунке 11.

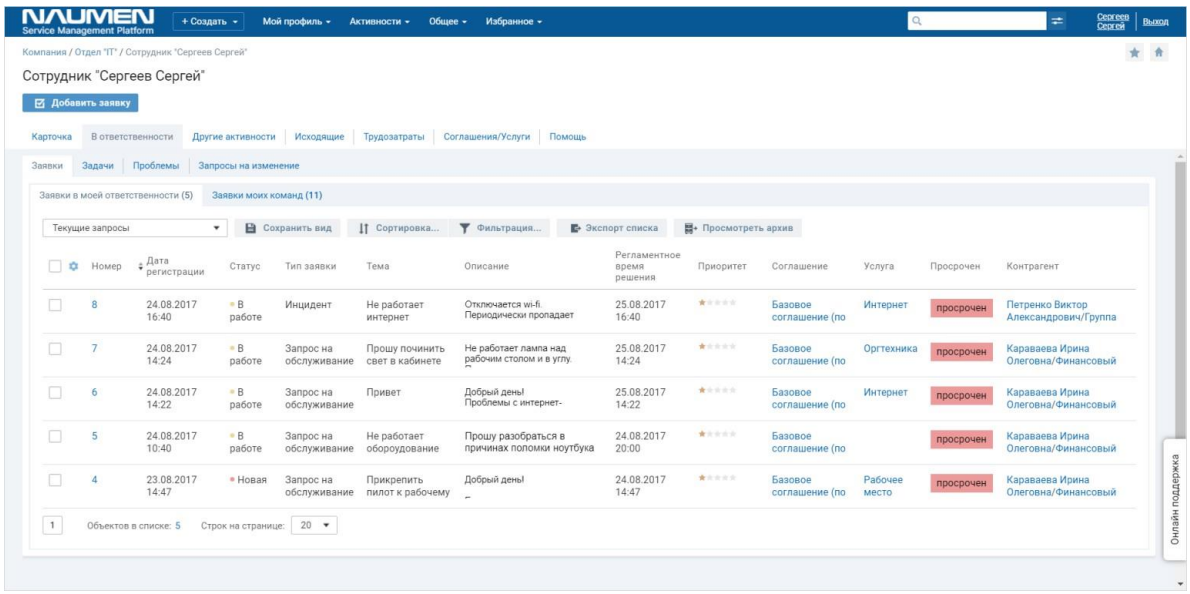

Рисунок 11 – Окно заявок Naumen

Также в Naumen можно просмотреть подробности заявки (Рисунок 12).

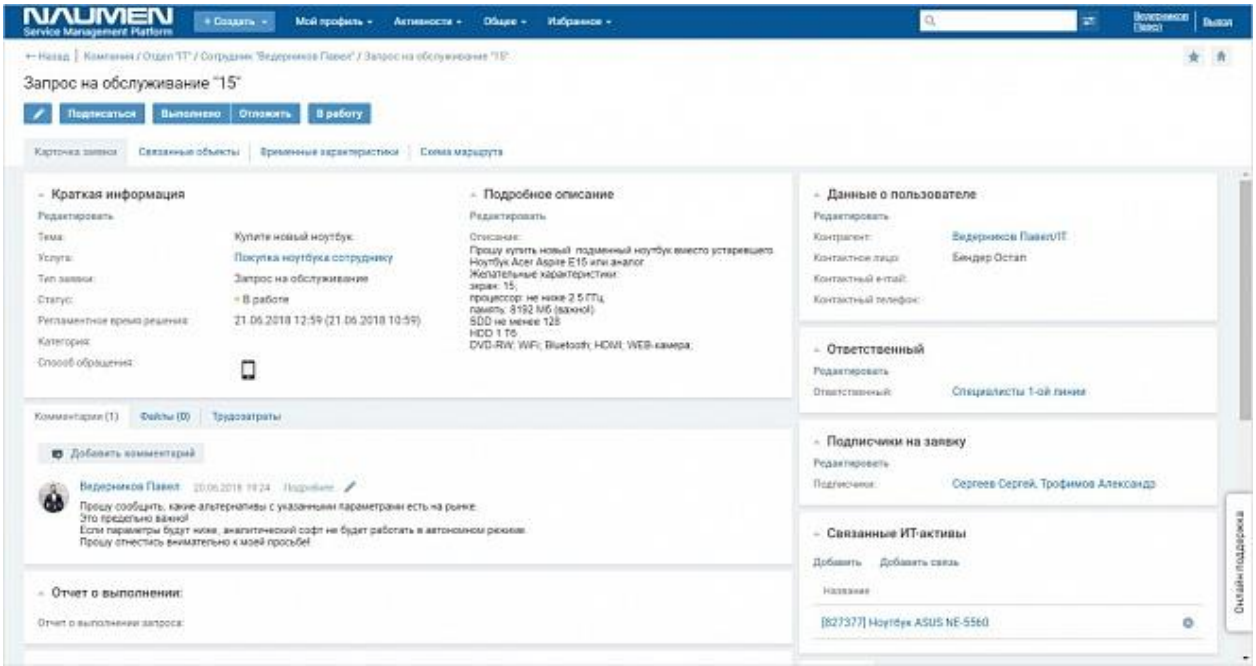

Рисунок 12 – Подробности заявки

Достоинствами приложения являются:

– подходит для крупного бизнеса, госструктур, телекома;

– поддержка SAP, 1С, Oracle, Active Directory и др.;
– готовые и настраиваемые отчеты;
– разграничение прав доступа.

К недостаткам можно отнести:

– отсутствие мобильного приложения;
– требуются время и ресурсы на настройку. Зачастую требуется помощь от сотрудников организации Naumen;
– устаревший дизайн.

При создании приложения необходимо четко сфокусироваться на реализации ключевых функций, одновременно минимизируя типичные недостатки аналогов. В отличие от существующих решений, разрабатываемое приложение будет предлагать полноценную офлайн-работу и узкоспециализированный функционал, ориентированный исключительно на курьерские услуги. Это позволит обеспечить более надежную и удобную платформу для пользователей в данной конкретной сфере.

1.4 Выбор средств проектирования и разработки

Android Studio представляет собой мощную интегрированную среду разработки (IDE), созданную Google специально для построения нативных приложений под операционную систему Android. Этот инструмент занимает центральное место в экосистеме Android-разработки, предоставляя программистам все необходимые средства для создания, тестирования и публикации высокопроизводительных мобильных приложений. По своей роли в разработке под Android он сопоставим с такими фреймворками, как Laravel для PHP или Ruby on Rails для Ruby, однако в отличие от них ориентирован исключительно на мобильную платформу [25].

Одним из ключевых преимуществ Android Studio является предоставление полного доступа ко всем возможностям Android-платформы. В отличие от кроссплатформенных решений вроде Expo, которые

ограничивают разработчиков набором API, доступных через свой SDK, Android Studio позволяет использовать любые нативные функции операционной системы без каких-либо ограничений.

Это включает работу с аппаратными сенсорами, фоновыми сервисами, кастомными драйверами, системными API и другими низкоуровневыми возможностями, что особенно важно для сложных проектов, требующих глубокой интеграции с платформой.

Среда разработки тесно интегрирована с официальным Android SDK и поддерживает современные языки программирования - Kotlin и Java. [26] Kotlin, ставший предпочтительным языком для Android-разработки, предлагает более лаконичный и безопасный синтаксис по сравнению с Java, при этом сохраняя полную совместимость с существующим кодом.

Android Studio включает мощные инструменты для работы с пользовательским интерфейсом, такие как визуальный редактор макетов (Layout Editor), который позволяет создавать интерфейсы как графически, так и через XML-разметку. Система сборки на основе Gradle предоставляет гибкие возможности для управления зависимостями и настройки процесса компиляции.

Для отладки и тестирования приложений Android Studio предлагает комплекс профессиональных инструментов. Встроенный эмулятор поддерживает различные версии Android и конфигурации устройств, позволяя тестировать приложения в разных условиях.

Профилировщики производительности дают детальную информацию о потреблении ресурсов процессора, памяти и батареи, помогая оптимизировать работу приложения.

Инструменты тестирования включают поддержку модульных (JUnit) и UI-тестов (Espresso), что способствует созданию надежного и стабильного кода.

Однако использование Android Studio связано с определенными сложностями. Основной барьер - достаточно высокий порог входа для начинающих разработчиков.

В отличие от более простых решений вроде Expo, здесь требуется глубокое понимание архитектуры Android, принципов работы SDK и особенностей выбранного языка программирования.

Процесс настройки среды разработки также более сложен - необходимо правильно установить и настроить Android SDK, эмулятор, систему сборки и другие компоненты.

Еще одним ограничением является платформенная специфичность - Android Studio предназначена исключительно для разработки под Android, поэтому создание кроссплатформенных приложений потребует использования дополнительных инструментов или фреймворков, таких как Flutter или React Native.

Также стоит отметить значительные требования к аппаратным ресурсам - для комфортной работы с Android Studio, особенно при использовании эмулятора, требуется достаточно мощный компьютер.

Несмотря на эти ограничения, Android Studio остается незаменимым инструментом для профессиональной разработки под Android. Его мощь и гибкость особенно востребованы в проектах, где критически важны производительность, доступ ко всем функциям платформы и возможность тонкой настройки всех аспектов работы приложения.

Для команд, ориентированных исключительно на Android-платформу и нуждающихся в максимальном контроле над процессом разработки, Android Studio продолжает быть оптимальным выбором.

Сравнение Android Studio с другими популярными фреймворками для разработки мобильных приложений представлено в таблице 2.

Таблица 2 – Сравнение популярных фреймворков мобильной разработки

Критерий	Android Studio	Expo (React Native)	Flutter	Xcode (iOS)
Тип разработки	Нативная (Android)	Кроссплатформенная (JS)	Кроссплатформенная (Dart)	Нативная (iOS/macOS)
Доступ к API	Полный (все Android-функции)	Ограничен (только Expo SDK)	Широкий (но не все нативные возможности)	Полный (все iOS-функции)
Языки	Kotlin, Java	JavaScript/TypeScript	Dart	Swift, Objective-C
Производительность	Высокая (нативный код)	Средняя (JS-мост)	Высокая (компиляция в нативный код)	Высокая (нативный код)
Сборка и публикация	Локальная (ручная настройка)	Облачная (Expo Build Service)	Локальная/Облачная	Локальная (Xcode Cloud)
Поддержка	Официальная (Google)	Сообщество + Expo Inc.	Официальная (Google)	Официальная (Apple)

Основными критериями Android Studio стали следующие:

- прямой доступ ко всем API и функциям Android без ограничений;
- возможность работы с низкоуровневыми системными компонентами;
- встроенные инструменты для работы с интерфейсами (Layout Editor);
- поддержка современных языков (Kotlin, Java, C++);
- продвинутый эмулятор с поддержкой разных версий Android;
- настройка через Gradle с поддержкой различных конфигураций;
- возможность кастомизации процесса сборки;
- умный встроенный редактор кода с автодополнением и рефакторингом.

Android Studio Editor является встроенным редактором кода Android Studio и представляет собой мощный инструмент, разработанный специально для профессиональной Android-разработки. Основанный на платформе IntelliJ IDEA, он сочетает в себе все преимущества современной IDE с уникальными функциями, ориентированными именно на создание мобильных приложений.

Главной особенностью редактора является его глубокая интеграция с экосистемой Android. При работе с кодом разработчик получает интеллектуальные подсказки, учитывающие специфику Android API. Редактор

автоматически предлагает соответствующие методы и константы из SDK, что значительно ускоряет процесс написания кода. Особенно это заметно при работе с такими компонентами, как Activity, Fragment или ViewModel.

Для языковой поддержки редактор предлагает полный набор возможностей работы с Kotlin и Java. В случае с Kotlin, который стал предпочтительным языком для Android-разработки, реализованы все современные функции: от умного автодополнения до сложного рефакторинга. При этом сохраняется полная совместимость с Java-кодом, что важно для работы с унаследованными проектами.

Работа с XML-ресурсами - еще одна сильная сторона редактора. При редактировании файлов макетов или манифеста разработчик получает контекстно-зависимую помощь. Например, при работе с AndroidManifest.xml редактор автоматически предлагает доступные разрешения, а при редактировании строковых ресурсов - поддерживает мультиязычность.

Отладка кода реализована на высоком уровне. Разработчик может устанавливать точки останова, проверять значения переменных в реальном времени и даже изменять некоторые параметры прямо во время выполнения приложения. Особенно полезной оказывается функция "Evaluate Expression", позволяющая проверить работу отдельных участков кода без необходимости перезапуска приложения.

Производительность редактора оптимизирована для работы с большими проектами. Благодаря системе индексации навигация по коду остается быстрой даже в сложных кодовых базах. При этом редактор потребляет значительно меньше ресурсов, чем многие аналоги, что делает комфортной работу даже на компьютерах средней мощности.

Для командной работы предусмотрены удобные инструменты интеграции с системами контроля версий. Разработчик может просматривать изменения, создавать коммиты и разрешать конфликты прямо в интерфейсе редактора. Поддержка популярных VCS, включая Git, SVN и Mercurial, реализована на уровне базовой функциональности.

Кастомизация интерфейса позволяет каждому разработчику настроить рабочее пространство под свои предпочтения. Доступны различные цветовые схемы, настройки шрифтов и даже режимы отображения. Например, можно включить "режим презентации" с увеличенным текстом или "ночной режим" для работы в темное время суток. Ниже на рисунке 13 представлено окно с редактором кода.

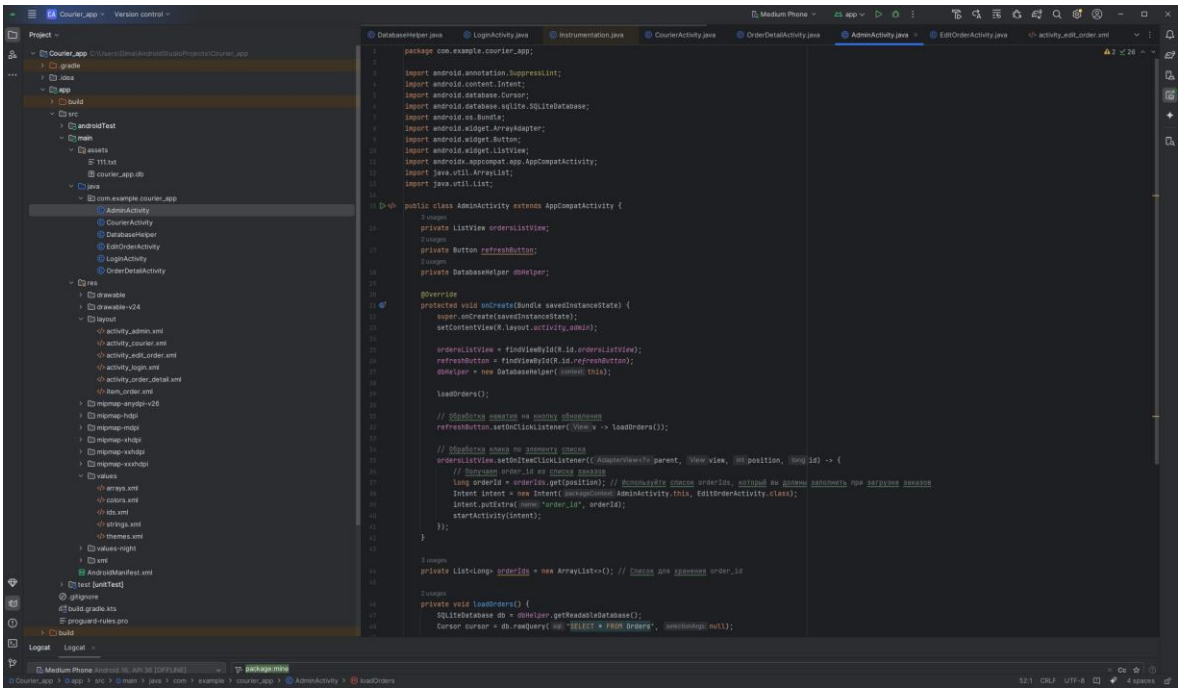

Рисунок 13 – Редактор кода Android Studio Editor

Android Studio Editor предлагает гибкие возможности персонализации рабочей среды. Пользователи могут адаптировать интерфейс под свои нужды - изменять расположение элементов, подключать или отключать дополнительные модули, а также настраивать инструментальную панель. Эти параметры доступны как для отдельных проектов, так и для глобальных настроек. Редактор поддерживает многозадачность с возможностью разделения экрана на рабочие области и использования внешних дисплеев для повышения эффективности разработки [24].

SQLite - это легковесная реляционная СУБД с открытым кодом, использующая стандартный SQL-синтаксис в противовес NoSQL-системам [30]. Её ключевая особенность заключается в автономной файловой архитектуре, где вся база данных хранится в едином файле, что особенно востребовано в мобильных и встраиваемых системах. В отличие от клиент-серверных СУБД, SQLite функционирует как встраиваемая библиотека, работающая в адресном пространстве основного приложения без необходимости отдельного серверного процесса [4],[6],[18].

SQLite поддерживает стандартный синтаксис SQL (Structured Query Language) и предлагает все основные возможности реляционных баз данных, включая транзакции, триггеры, представления и вложенные запросы. В отличие от клиент-серверных СУБД, SQLite не требует отдельного серверного процесса и работает непосредственно с файлом базы данных [7],[9].

Данные в SQLite организованы в виде традиционных таблиц со строгим соблюдением схемы данных [11],[12]. Каждая таблица содержит заранее определённые столбцы с конкретными типами данных, что обеспечивает целостность информации. При этом SQLite использует динамическую типизацию - значения могут храниться в любом типе столбца, независимо от объявленного типа данных [13],[14],[18].

Основные преимущества выбора данной СУБД:

- вся база данных содержится в одном файле на диске;
- не требует серверного процесса (работает как библиотека);
- вся СУБД занимает несколько сотен килобайт;
- быстрый доступ к данным благодаря локальному хранению.

1.5 Постановка задачи на разработку

Основные функциональные требования к приложению следующие:

- просмотр списка заказов;
- добавление/редактирование заказов (тип документа, статус, адрес);

- назначение заказов курьерам;
- генерация отчетов по заказам в PDF-формате;
- просмотр списка доступных курьеров;
- назначение заказов конкретным курьерам;
- просмотр заказов, назначенных курьеру;
- обновление статуса заказа курьером;
- управление справочниками;
- добавление/редактирование данных о пользователях;
- привязка заказов к маршрутам (точки отправления/доставки);
- разделение доступа по ролям (Администратор, диспетчер, курьер);
- возможность работы в оффлайн-режиме.

После описания предметной области, выбора средств разработки, описания аналогичных систем и выбора функций, которые необходимо реализовать в приложении, опишем процесс разработки мобильного приложения.

Выводы по первой главе

В первой главе представлено описание предметной области, моделирование бизнес-процессов предметной области. Также проводится обзор аналогов и выбор средств проектирования и разработки, в результате чего для разработки приложения был выбран фреймворк Android Studio, а в качестве субд SQLite. Также проведена постановка задачи на разработку системы, описаны ее основные функции.

Глава 2 Реализация программного обеспечения

2.1 UML моделирование системы

Проведем моделирование разрабатываемой системы с использованием нотации UML. Диаграмма вариантов использования (use-case diagram) позволяет понять, какие основные функции будет реализованы в системе, определить круг пользователей, доступные им функции, и зависимости между функциями [5]. На рисунке 14 приведена диаграмма вариантов использования (прецедентов) для разрабатываемого мобильного приложения курьерской службы.

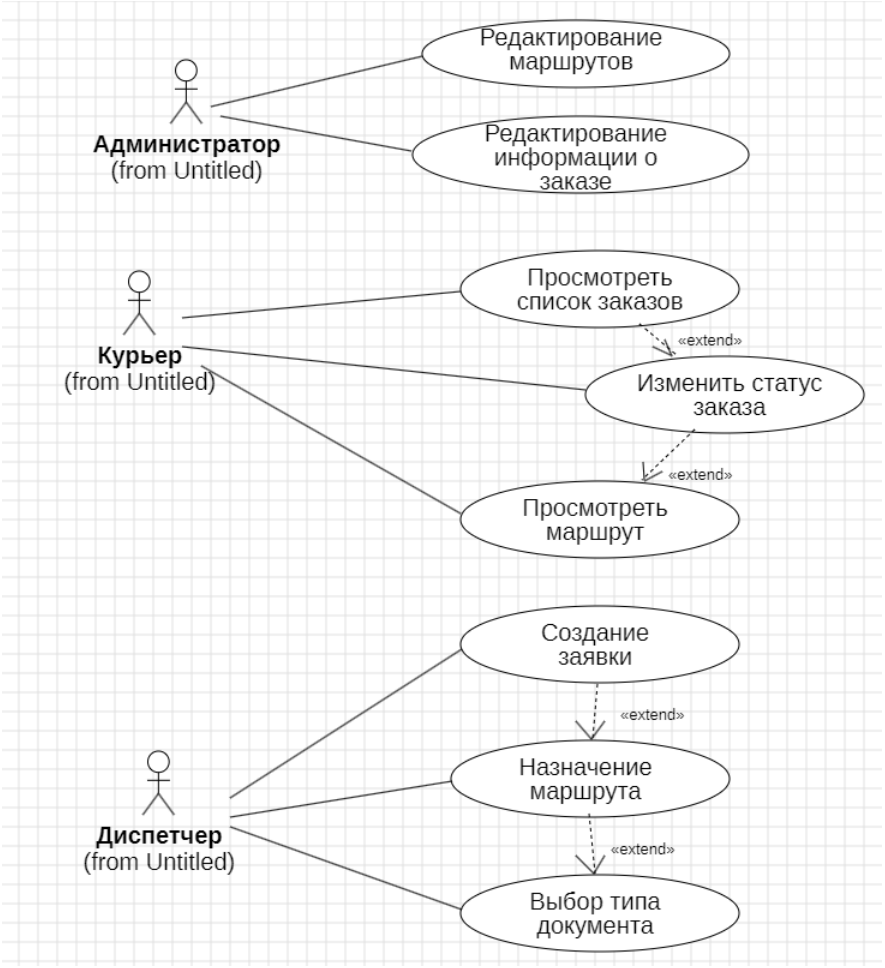

Рисунок 14 – Диаграмма вариантов использования

Диаграмма включает в себя три роли: курьер, диспетчер и администратор.

Диспетчер, после входа в приложение, может создать заявку на транспортировку документа, а также указать тип передаваемого документа. Назначить адрес откуда забрать и куда доставить документ. После назначения адреса диспетчер имеет возможность завершить создание заявки и направить ее на обработку курьерам.

Диаграмма последовательности процесса оформления заказа показана на рисунке 15.

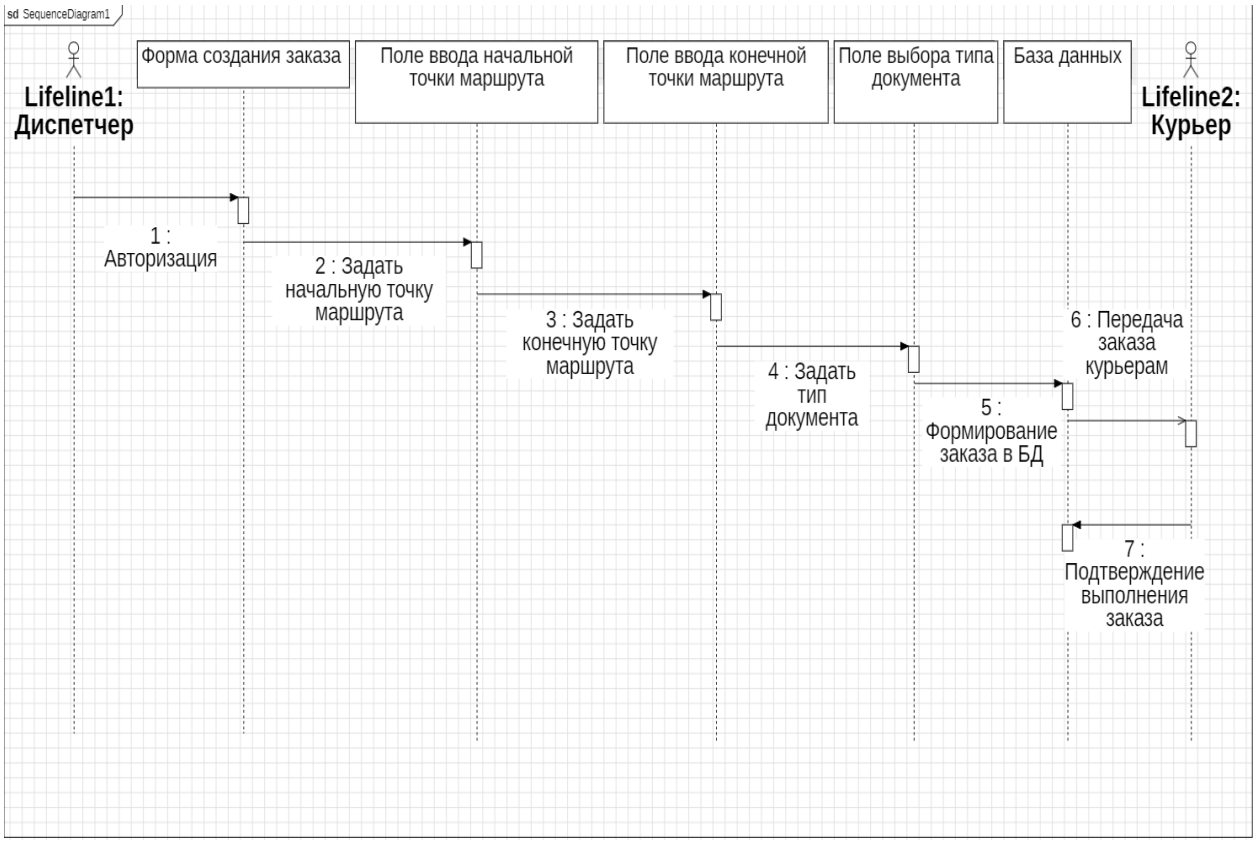

Рисунок 15 – Диаграмма последовательности

Курьер при авторизации в приложении видит доступный список заказов доступных в БД. После выполнения заказа курьер подтверждает заказ и статус заказа обновляется на выполненный.

2.1.1 Диаграмма классов курьерского приложения

При создании программных систем особое значение приобретает этап проектирования архитектуры, где ключевую роль играет построение диаграмм классов UML. Эти диаграммы служат графическим представлением структуры разрабатываемого приложения, отображая основные элементы системы и отношения между ними. Такой подход к проектированию позволяет наглядно представить сложную структуру данных и взаимодействие компонентов.

Основным строительным блоком диаграммы выступает класс - абстракция, описывающая конкретную сущность предметной области. В каждом классе выделяют два принципиальных аспекта: характеристики объекта (атрибуты) и его поведение (методы). Особую ценность представляют связи между классами, демонстрирующие взаимодействие различных компонентов системы и формирующие целостное представление об архитектуре[10].

Практическая значимость диаграмм классов проявляется в нескольких аспектах. Они позволяют заблаговременно обнаруживать архитектурные проблемы, такие как чрезмерная связанность компонентов или нарушение принципов инкапсуляции. Кроме того, эти диаграммы выполняют важную документационную функцию, облегчая процесс вхождения новых разработчиков в проект и координацию работы команды.

С точки зрения жизненного цикла разработки, диаграммы классов формируют фундамент для последующей реализации системы. Они не только фиксируют текущее состояние проекта, но и открывают возможности для архитектурных улучшений. Тщательный анализ диаграммы может выявить участки, где применение определенных шаблонов проектирования позволит повысить гибкость и масштабируемость системы.

На рисунке 16 представлена диаграмма классов курьерского приложения.

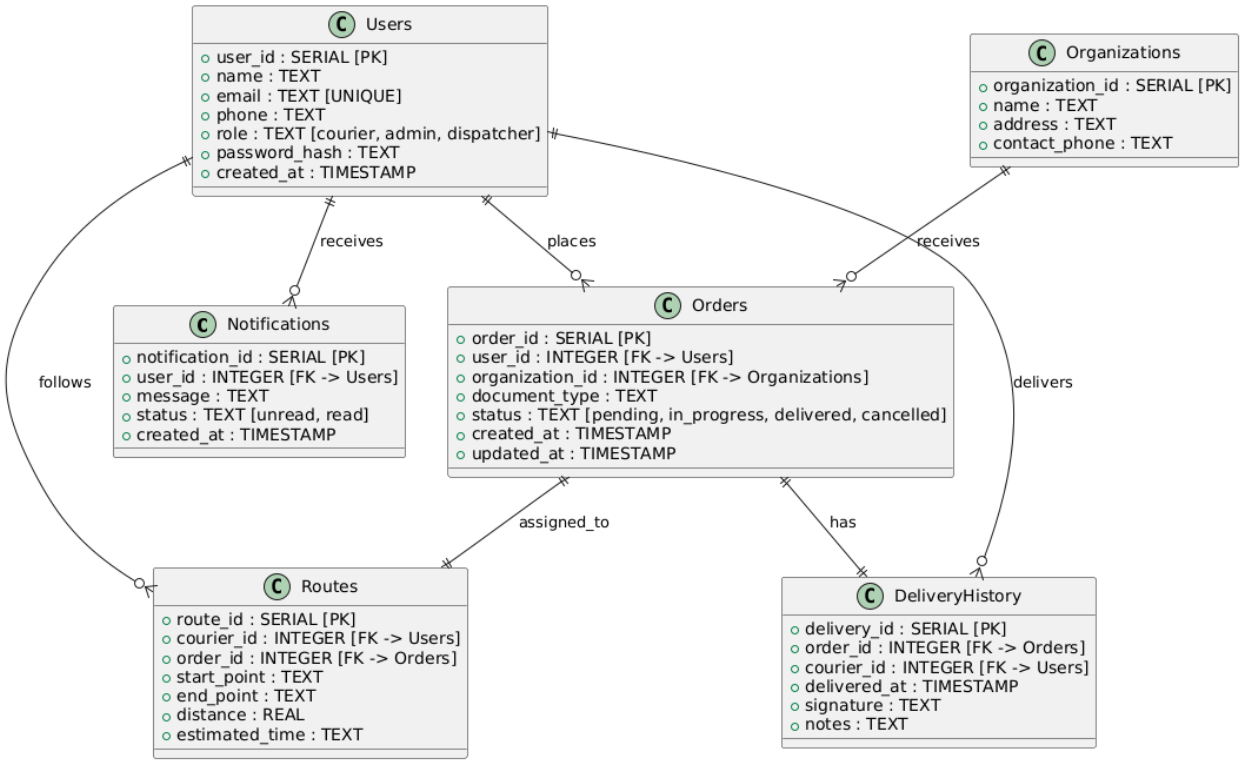

Рисунок 16 – Диаграмма классов курьерского приложения

Грамотно построенная диаграмма классов существенно упрощает процесс разработки сложных систем, способствует созданию продуманной архитектуры и минимизирует вероятность появления ошибок на этапе кодирования. Именно поэтому данный этап считается одним из наиболее важных в процессе создания качественного программного обеспечения.

2.1.2 Диаграмма компонентов курьерского приложения

Диаграмма компонентов представляет собой важный инструмент визуального моделирования в UML, который отображает структурную организацию программной системы на уровне ее составных частей. В отличие от диаграммы классов, которая фокусируется на логической структуре, диаграмма компонентов отражает физическое построение системы из отдельных модулей [2]. На рисунке 17 представлена диаграмма компонентов курьерского приложения.

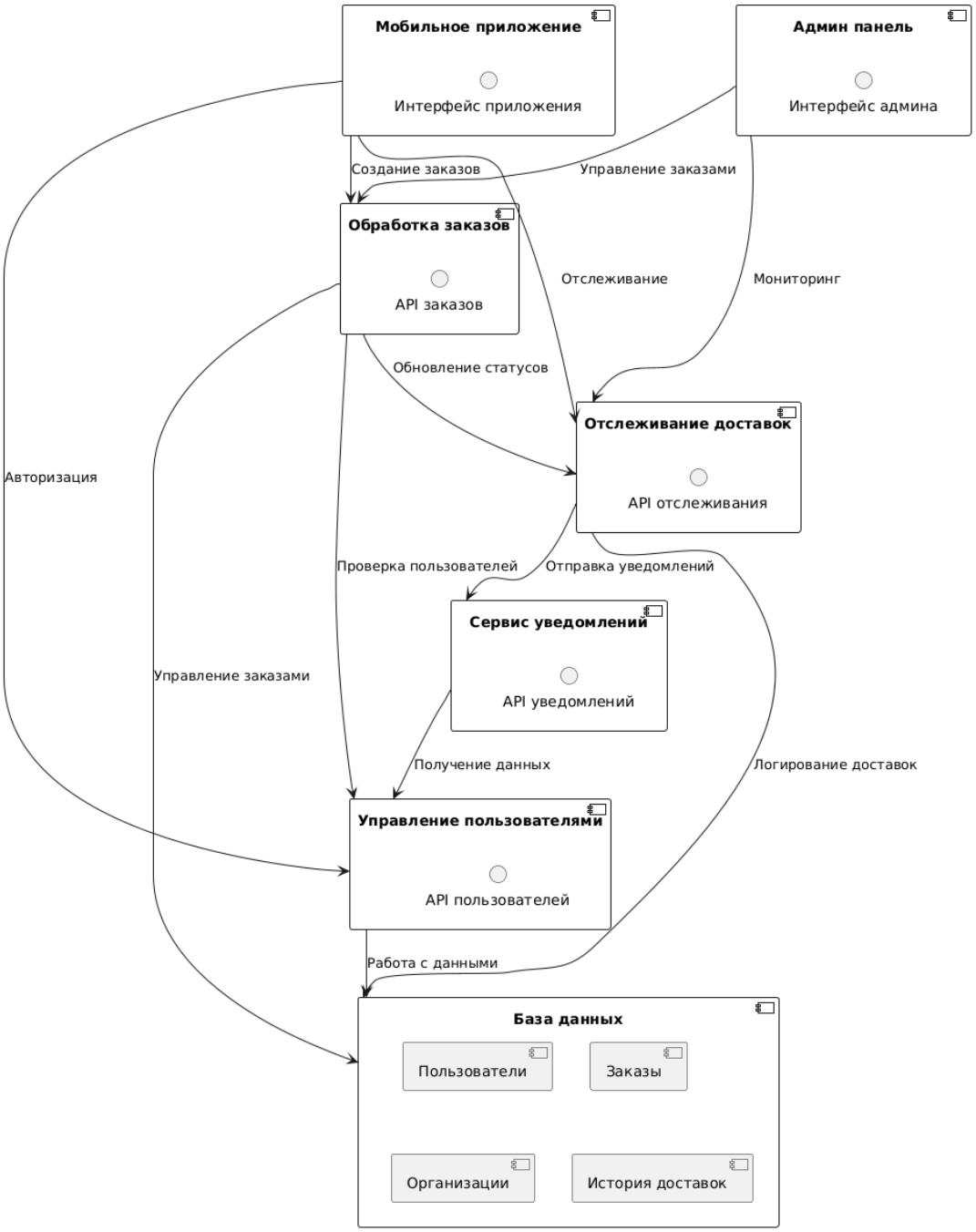

Рисунок 17 – Диаграмма компонентов

Использование диаграмм компонентов в процессе проектирования способствует созданию более продуманной и гибкой архитектуры. Они позволяют заранее спланировать процесс интеграции модулей и минимизировать риски, связанные с зависимостями между компонентами системы [8].

2.2 Реализация мобильного приложения

Общая архитектура реализуемого мобильного приложения схематично представлена ниже.

```
courier_app/
├── src/
│   ├── main/
│   │   ├── assets/
│   │   │   └── courier_app.db
│   │   └── java/
│   │       └── com/example/courier_app/
│   │           ├── AdminActivity.java
│   │           ├── CourierActivity.java
│   │           ├── DispatcherActivity.java
│   │           ├── DatabaseHelper.java
│   │           ├── EditOrderActivity.java
│   │           ├── EditUserActivity.java
│   │           ├── LoginActivity.java
│   │           ├── OrderDetailActivity.java
│   │           └── UserManagmentActivity.java
├── build.gradle
└── AndroidManifest.xml
```

Структура проекта включает в себя следующие основные компоненты.

Административный модуль (AdminActivity.java).

Центральный компонент для управления системой. Позволяет просматривать все заказы, назначать их курьерам, генерировать отчеты и управлять справочными данными (типы документов, статусы заказов). Интегрирован с DatabaseHelper для работы с локальной базой данных. Модуль включает в себя EditUserActivity(Модуль просмотра данных о пользователях) и UserManagmentActivity(Модуль добавления и редактирования

пользователей) [15],[16],[19]. Далее представлен код реализации загрузки информации о заказах, хранящейся в базе данных.

```
private void loadOrders() {
    SQLiteDatabase db = dbHelper.getReadableDatabase();
    Cursor cursor = db.rawQuery("SELECT * FROM Orders ORDER BY updated_at DESC", null);
    List<String> orders = new ArrayList<>();
    orderIds.clear();
    while (cursor.moveToNext()) {
      @SuppressLint("Range") long orderId = cursor.getLong(cursor.getColumnIndex("order_id"));
      @SuppressLint("Range") String status = cursor.getString(cursor.getColumnIndex("status"));
      @SuppressLint("Range") String updatedAt = cursor.getString(cursor.getColumnIndex("updated_at"));
      String orderInfo = "Order ID: " + orderId +
          "\nStatus: " + status +
          "\nLast updated: " + updatedAt;
      orders.add(orderInfo);
      orderIds.add(orderId);
    }
    cursor.close();
    ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple_list_item_1, orders);
    ordersListView.setAdapter(adapter);
  }
```

Модуль курьера (CourierActivity.java).

Основной рабочий интерфейс для курьеров. Обеспечивает просмотр назначенных заказов с детализацией маршрутов, позволяет обновлять статусы доставки в реальном времени и формировать цифровые акты приема-передачи

[17],[20],[21]. Интегрирован с OrderDetailActivity.java(Модуль промотра информации о заявке), с помощью которого есть возможность сгенерировать отчет. Ниже представлен код для генерации отчета.

```
// Формируем строку с информацией о заказе
        @SuppressLint("Range") String orderInfo = "Детали заказа\n\n" +
"Заказ ID: " + cursor.getInt(orderIdIndex) + "\n" +
"Откуда: " + cursor.getString(startPointIndex) + "\n" +
"Куда: " + cursor.getString(endPointIndex) + "\n" +
"Тип документа: " + cursor.getString(documentTypeIndex) + "\n" +
"Статус: " + cursor.getString(statusIndex) + "\n" +
"Курьер: " + cursor.getString(courierNameIndex);
// Создаем PDF-документ
PdfDocument pdfDocument = new PdfDocument();
// Создаем страницу
PdfDocument.PageInfo pageInfo = new PdfDocument.PageInfo.Builder(612,
792, 1).create(); // Размер страницы A4 (612x792 точек)
PdfDocument.Page page = pdfDocument.startPage(pageInfo);
// Рисуем текст на странице
Canvas canvas = page.getCanvas();
Paint paint = new Paint();
// Заголовок
paint.setColor(Color.BLACK);
paint.setTextSize(18);
paint.setFakeBoldText(true); // Жирный текст
canvas.drawText("Отчет о заказе", 50, 50, paint);
// Разделительная линия
paint.setStrokeWidth(2);
canvas.drawLine(50, 60, 562, 60, paint);
// Основной текст
paint.setTextSize(14);
```

```
paint.setFakeBoldText(false); // Обычный текст
String[] lines = orderInfo.split("\n");
int y = 100; // Начальная позиция по Y
for (String line : lines) {
canvas.drawText(line, 50, y, paint);
y += 30; // Отступ между строками
}
// Завершаем страницу
pdfDocument.finishPage(page);
// Сохраняем PDF в папку "Загрузки"
File downloadsDir = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS);
File file = new File(downloadsDir, "OrderReport_" + orderId + ".pdf");
try {
// Записываем PDF в файл
pdfDocument.writeTo(new FileOutputStream(file));
Toast.makeText(this, "Отчет сохранен в " + file.getAbsolutePath(), Toast.LENGTH_LONG).show();
// Показываем диалог с выбором: открыть файл или нет
showOpenFileDialog(file);
} catch (IOException e) {
Toast.makeText(this, "Ошибка при сохранении отчета: " + e.getMessage(), Toast.LENGTH_SHORT).show();
} finally {
pdfDocument.close();
}
```

Модуль диспетчера (DispatcherActivity.java).

Основной рабочий интерфейс для диспетчеров. Позволяет создавать заявки, указывать тип документов, указывать начальный и конечный адреса доставки, назначать курьеров.

Система авторизации (LoginActivity.java).

Реализует безопасный вход в систему с разделением прав доступа. Поддерживает две роли пользователей: администратор (полный доступ) и курьер (ограниченный функционал). Учетные данные проверяются против локальной SQLite базы через DatabaseHelper [22],[27],[28]. Ниже приведен код, после получения роли пользователя приложением, для открытия соответствующей начальной формы.

```
Intent intent;
        switch (role) {
            case "courier":
                intent = new Intent(LoginActivity.this, CourierActivity.class);
                break;
            case "admin":
                intent = new Intent(LoginActivity.this, AdminActivity.class);
                break;
            case "dispatcher":
                intent = new Intent(LoginActivity.this, DispatcherActivity.class);
                intent.putExtra("dispatcher_id", userId); // Передаем ID диспетчера
                break;
            default:
                throw new IllegalStateException("Unexpected value: " + role);
        }
        startActivity(intent);
    } else {
```

Toast.makeText(this, "Неверные учетные данные", Toast.LENGTH_SHORT).show();}

Ядро данных (DatabaseHelper.java).

Централизованный модуль работы с данными. Инкапсулирует всю логику работы с SQLite базой: создание/обновление схемы данных, выполнение запросов к таблицам заказов и маршрутов, обеспечение целостности данных. Поддерживает оффлайн-режим работы.

Модуль работы с заказами.

Включает OrderDetailActivity.java (просмотр деталей заказа) и EditOrderActivity.java (редактирование параметров). Обеспечивает полный цикл обработки заказа - от создания до завершения, включая генерацию PDF-отчетов и обновление статусов.

База данных (courier_app.db).

Содержит в себе внутреннюю базу данных приложения.

Основной файл конфигурации модуля приложения (build.gradle).

Подключает необходимые библиотеки, задаёт минимальную и целевую версии Android, на которых будет работать приложение, а также настраивает оптимизацию и защиту кода перед выпуском финальной версии. Без него сборка APK-файла невозможна [29].

«Паспорт» приложения (AndroidManifest.xml).

Объявляет системе Android все ключевые компоненты (активности, сервисы), запрашивает разрешения и задаёт основные настройки. Содержит в себе информацию о всех окнах приложения, запрашивает разрешения, задает название приложения, задает стартовое окно. Ниже представлен код содержащий в себе информацию о всех окнах приложения.

```
<!-- LoginActivity (Главный экран) -->
    <activity
        android:name=".LoginActivity"
        android:exported="true"
        android:label="Вход в систему">
```

```
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category    android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
        <!-- CourierActivity (Экран курьера) -->
        <activity
            android:name=".CourierActivity"
            android:label="Курьер" />
        <!-- OrderDetailActivity (Экран деталей заказа) -->
        <activity
            android:name=".OrderDetailActivity"
            android:label="Детали заказа" />
        <activity
            android:name=".AdminActivity"
            android:label="Детали заказа" />
        <activity
            android:name=".EditOrderActivity"
            android:label="Детали заказа" />
        <activity
            android:name=".DispatcherActivity"
            android:label="Детали заказа" />
        <activity android:name=".UserManagementActivity" />
        <activity android:name=".EditUserActivity" />
```

Выводы по второй главе

Во второй главе подробно рассмотрены этапы проектирования и реализации программного обеспечения для мобильного приложения курьерской службы. На основе UML-моделирования были разработаны ключевые диаграммы, включая диаграмму вариантов использования,

диаграмму последовательностей, диаграмму классов и компонентов. Эти модели позволили визуализировать структуру системы, определить роли пользователей, их взаимодействие с функционалом приложения, а также спроектировать архитектуру данных и взаимосвязи между компонентами.

Особое внимание уделено реализации модулей приложения, таких как административный интерфейс, функционал курьера и диспетчера, система авторизации и ядро работы с данными. Для каждого модуля приведены фрагменты кода, демонстрирующие ключевые функции, включая загрузку заказов, генерацию отчетов и управление ролями пользователей. Структура проекта и его конфигурационные файлы отражают продуманную организацию компонентов и их интеграцию.

Проведенное моделирование и реализация позволили создать четкую архитектуру приложения, обеспечивающую стабильную работу всех модулей, гибкость в управлении заказами и безопасность данных. Основной вывод главы заключается в успешном переходе от проектирования к реализации, что подтверждается детальным описанием логики системы и ее компонентов.

Глава 3 Тестирование ИС

3.1 Описание процесса тестирования

Оценка качества программного обеспечения представляет собой комплексный процесс, основанный на многоуровневой системе тестирования. Основная цель этого процесса - подтверждение соответствия разработанного продукта исходным техническим требованиям и бизнес-правилам.

Современные методики контроля качества предполагают последовательное применение различных видов тестирования, каждый из которых решает конкретные задачи на разных этапах жизненного цикла разработки.

На начальных стадиях разработки особое внимание уделяется модульному тестированию, которое позволяет проверить корректность работы отдельных компонентов системы. Такой подход дает возможность выявлять и устранять ошибки на ранних этапах, что существенно снижает стоимость их исправления в дальнейшем.

По мере развития проекта важную роль начинает играть интеграционное тестирование, направленное на проверку взаимодействия между различными модулями приложения.

После внесения значительных изменений в код обязательно проводится регрессионное тестирование. Его основная задача - убедиться, что новые функциональные возможности не нарушили работу существующих компонентов системы. Этот вид тестирования особенно важен в условиях непрерывной интеграции и частых обновлений продукта.

Завершающим этапом внутренней проверки является системное тестирование, в ходе которого полностью собранный продукт тестируется на соответствие всем заявленным требованиям. На этом этапе проверяются не только функциональные характеристики, но и нефункциональные параметры, такие как производительность, безопасность и удобство использования.

Особое место в процессе контроля качества занимают пользовательские испытания, которые включают альфа- и бета-тестирование. Эти методы позволяют получить ценную обратную связь от реальных пользователей и выявить потенциальные проблемы, которые могли быть упущены на предыдущих этапах проверки. Результаты таких испытаний часто становятся основой для дальнейшего совершенствования продукта и разработки новых функциональных возможностей.

Эффективная система контроля качества программного обеспечения должна сочетать различные методы тестирования, адаптированные под конкретный проект и его особенности.

Такой комплексный подход позволяет не только выявлять существующие проблемы, но и предотвращать потенциальные ошибки, что в конечном итоге приводит к созданию надежного и качественного программного продукта, соответствующего ожиданиям пользователей.

3.2 Представление и тестирование работы приложения

Перед началом тестирования необходимо выполнить следующие шаги:

- установить приложение на устройство с Android 8.0 или новее;
- проверить, что приложение корректно работает без интернета;
- авторизоваться под одной из ролей (администратор, диспетчер, курьер).

Приложение должно соответствовать следующим требованиям:

- функциональность: все операции выполняются без ошибок, доступ ограничен в зависимости от роли пользователя;
- удобство: интерфейс интуитивно понятен;
- надежность: данные сохраняются при перезапуске приложения, работа в оффлайн-режиме стабильна;
- безопасность: нет доступа к функциям без авторизации.

Таблица 3 – Тестовые сценарии

ID	Роль	Действие	Ожидаемый результат
T1	Все	Запуск без интернета	Данные загружаются из локальной базы
T2	Все	Ввод неверных учетных данных	Появляется сообщение об ошибке
T3	Администратор	Просмотр списка пользователей	Отображаются все пользователи с их ролями
T4	Администратор	Изменение роли пользователя	Новые данные сохраняются в системе
T5	Администратор	Изменение данных заказа	Данные изменяются, время обновления фиксируется.
T6	Диспетчер	Создание заказа	Заказ появляется в списке со статусом "В обработке"
T7	Диспетчер	Назначение курьера	В карточке заказа отображается выбранный курьер
T8	Курьер	Смена статуса на "В пути"	Статус и время обновления фиксируются
T9	Курьер	Генерация PDF-отчета	Файл сохраняется с корректными данными
T10	Диспетчер	Попытка редактирования пользователей	Функция недоступна
T11	Курьер	Попытка переназначить заказ	Функция недоступна
T12	Все	Перезапуск после изменений	Данные не теряются

Система блокирует сохранение при незаполненных обязательных полях.

Ввод неверных учетных данных (T2):

- открыть приложение;
- ввести неверные данные.

Ожидаем: На экране появляется уведомление об ошибке.

На рисунке 18 представлено уведомление об ошибке.

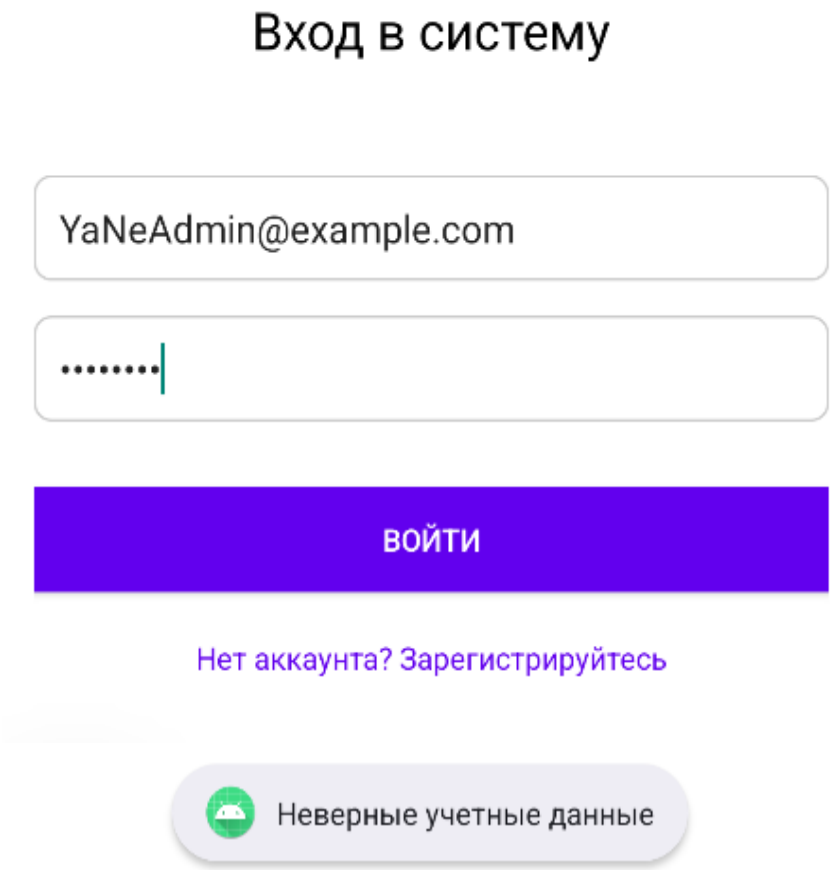

Рисунок 18 – Уведомление об ошибке

Изменение статуса (Т8):

– войти как курьер;
– выбрать заказ и изменить статус.

Ожидаем: Система фиксирует новое состояние и время.

На рисунке 19 и 20 представлены результаты тестирования изменения статуса.

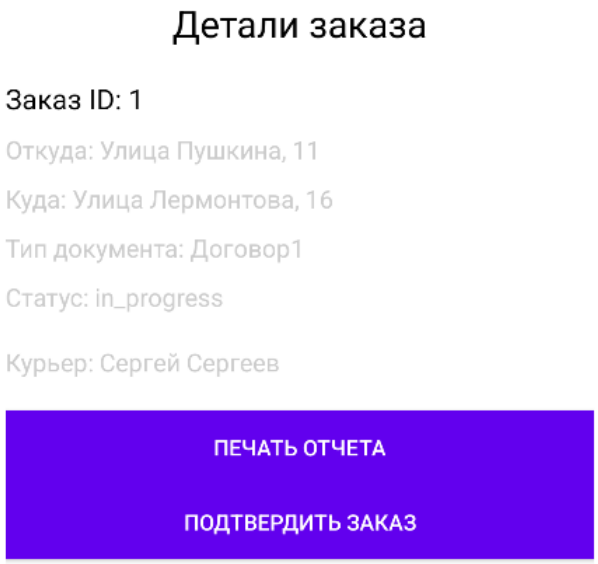

Рисунок 19 – Детали заказа до нажатия кнопки «Подтвердить заказ»

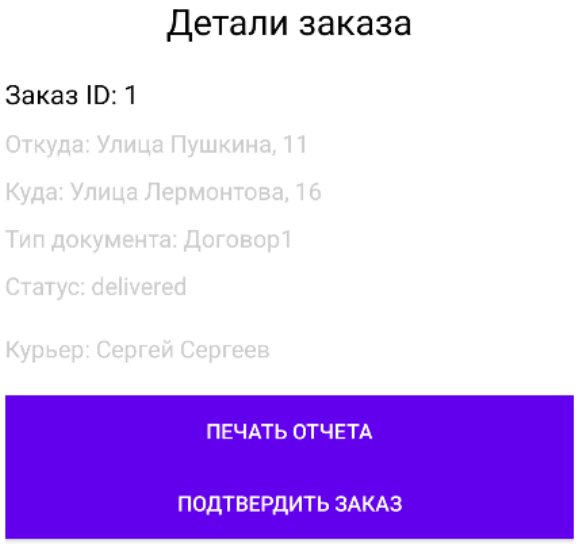

Рисунок 20 – Детали заказа после нажатия кнопки «Подтвердить заказ»

Статус доставки изменился на «delivered».

Генерация отчета (Т9)

– войти как курьер;
– открыть заказ и создать PDF.

Ожидаем: Файл сохраняется в папке "Загрузки" с полными данными.

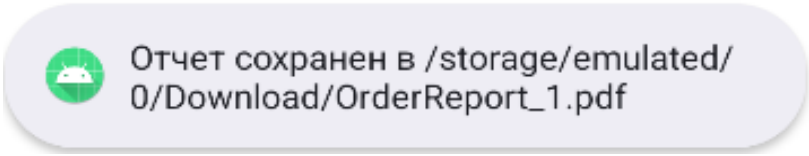

Рисунок 21 – Уведомление о расположении сохраненного файла

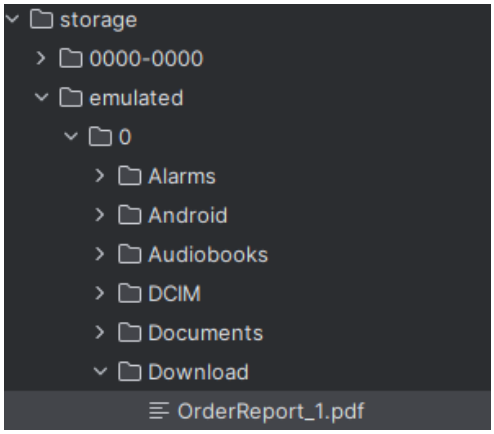

Рисунок 22 – Расположение файла в системе

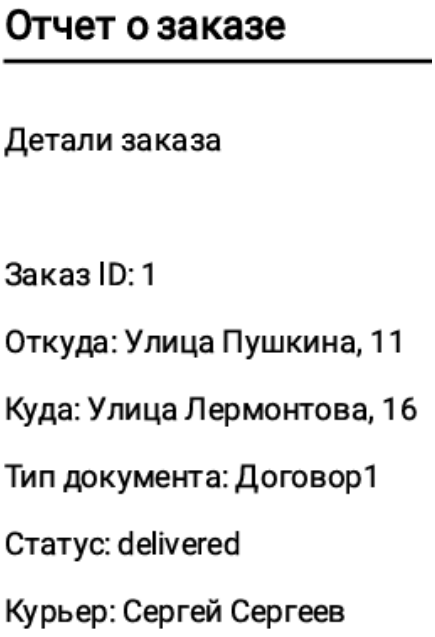

Рисунок 23 – Сгенерированный отчет

Структура окна курьеров сделана с использованием компонента ListView, что позволяет прокручивать список при наличии большого списка записей. На рисунке 24 представлена структура окна курьеров.

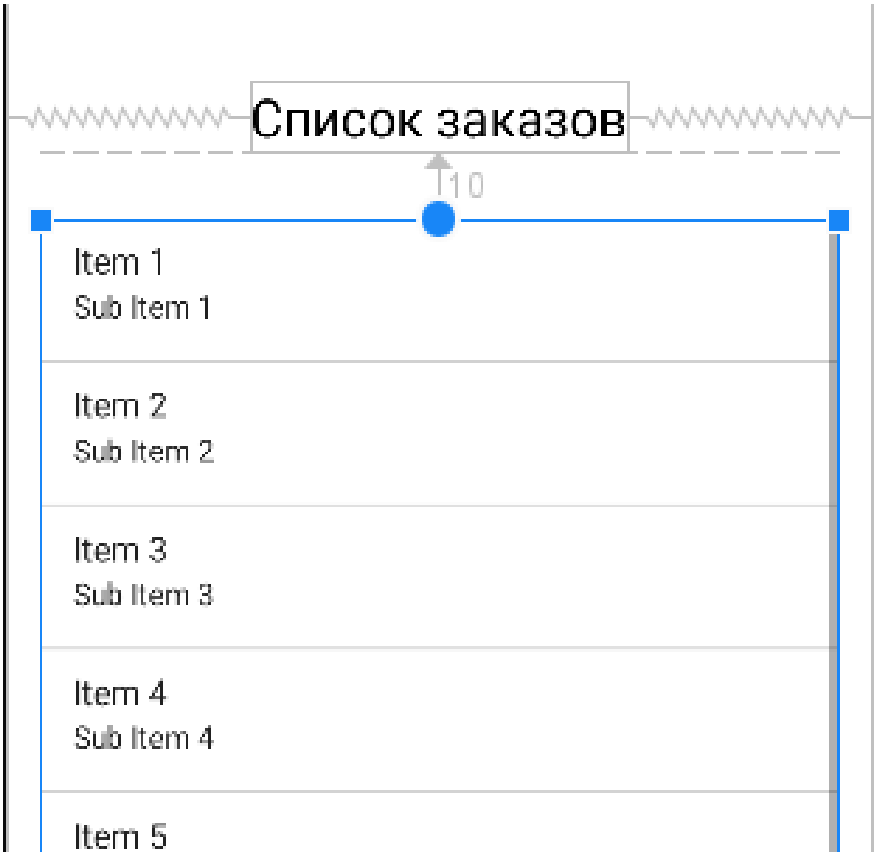

Рисунок 24 – Структура окна курьеров

В системе реализован функционал просмотра заказов для курьеров.

На рисунке 25 представлено окно курьеров с загруженной из базы данных информацией.

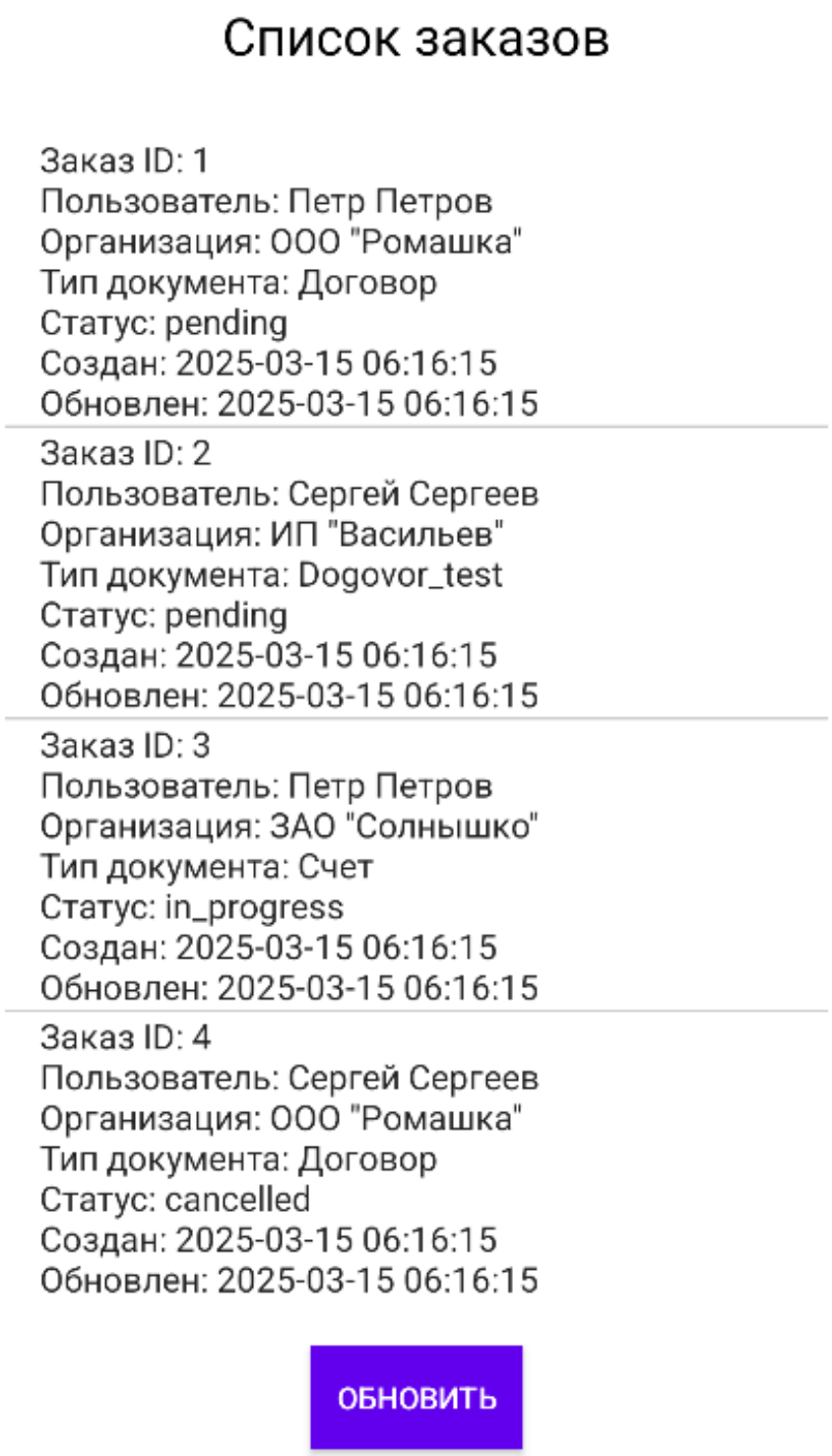

Рисунок 25 – Окно с информацией для курьеров

При нажатии на позицию – можно посмотреть адрес отправки, адрес доставки, тип документа, статус заказа и курьера, который взял на себя заказ.

Также можно сгенерировать файл отчета. На рисунке 26 показано окно «Детали заказа».

Для каждого заказа отображаются:

- номер заказа;
- курьер принявший заказ;
- организация куда необходимо доставить заказ;
- тип документа;
- статус заказа;
- дата создания заказа;
- дата обновления заказа;
- адрес отправки и адрес доставки заказа.

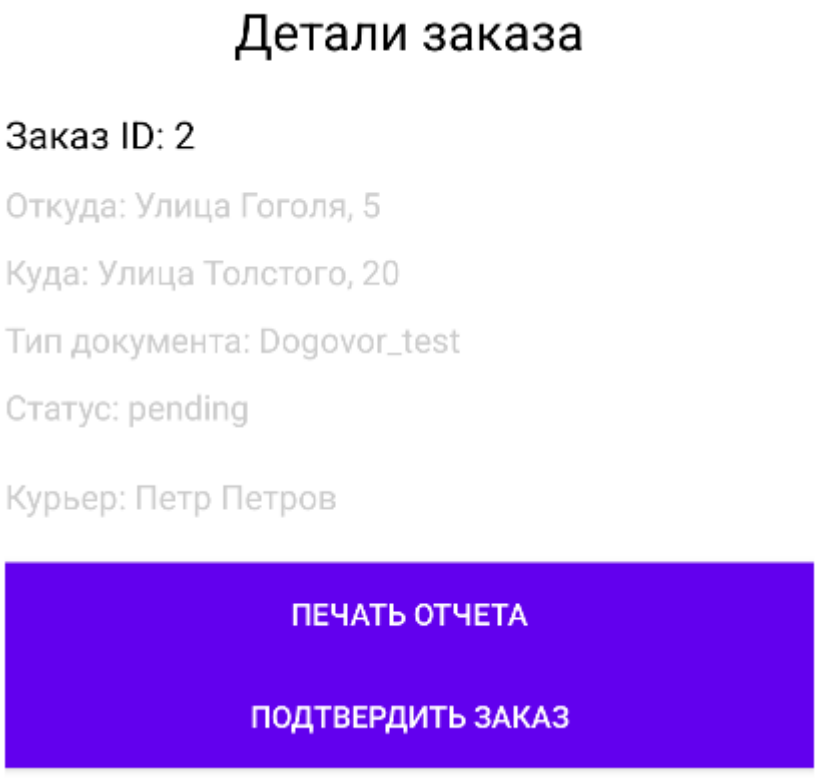

Рисунок 26 – Окно «Детали заказа»

При нажатии на кнопку «Печать отчета» создается PDF-файл в котором выведена информация с формы. Приложение предлагает сразу открыть файл или сделать это позже (Рисунок 27).

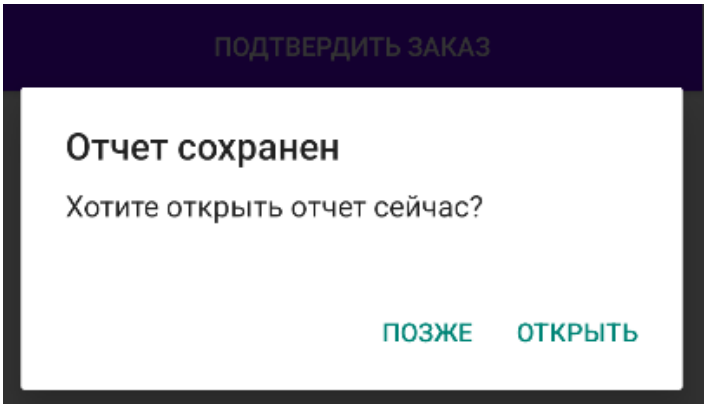

Рисунок 27 – Подтверждение открытия

Стандартным местом сохранения файла отчета является папка загрузки в основной памяти телефона.

На рисунке 28 представлен сгенерированный отчет.

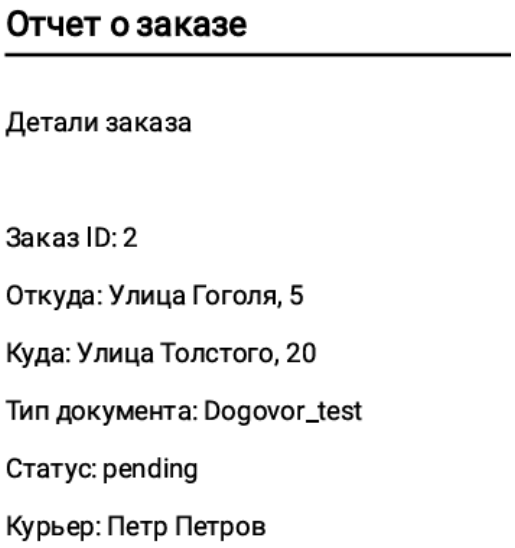

Рисунок 28 – Отчет

Реализована форма администратора, через которую происходит редактирование заказов, в частности изменения типа документа, статус, диспетчера, создавшего заявку, статус заявки, курьера, и адреса доставки.

Формы для администратора представлены на рисунках 29-31.

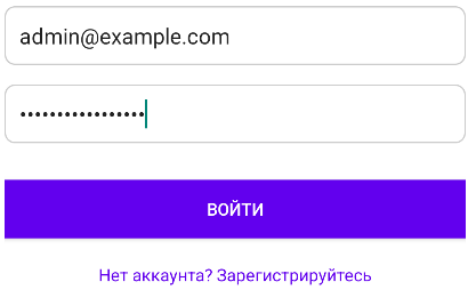

Рисунок 29 – Авторизация в качестве администратора

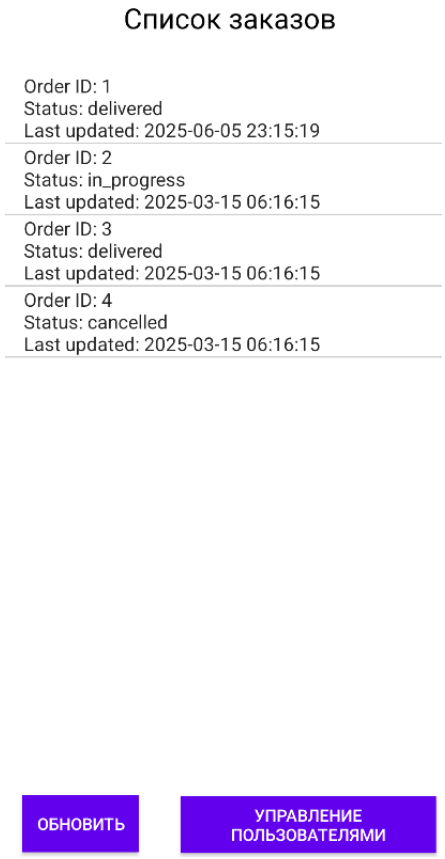

Рисунок 30 – Главный экран формы администратора

Также через эту форму имеется возможность создать нового пользователя или изменить информацию о существующем.

При нажатии на заказ открывается форма редактирования заказа, представленная на рисунке 31.

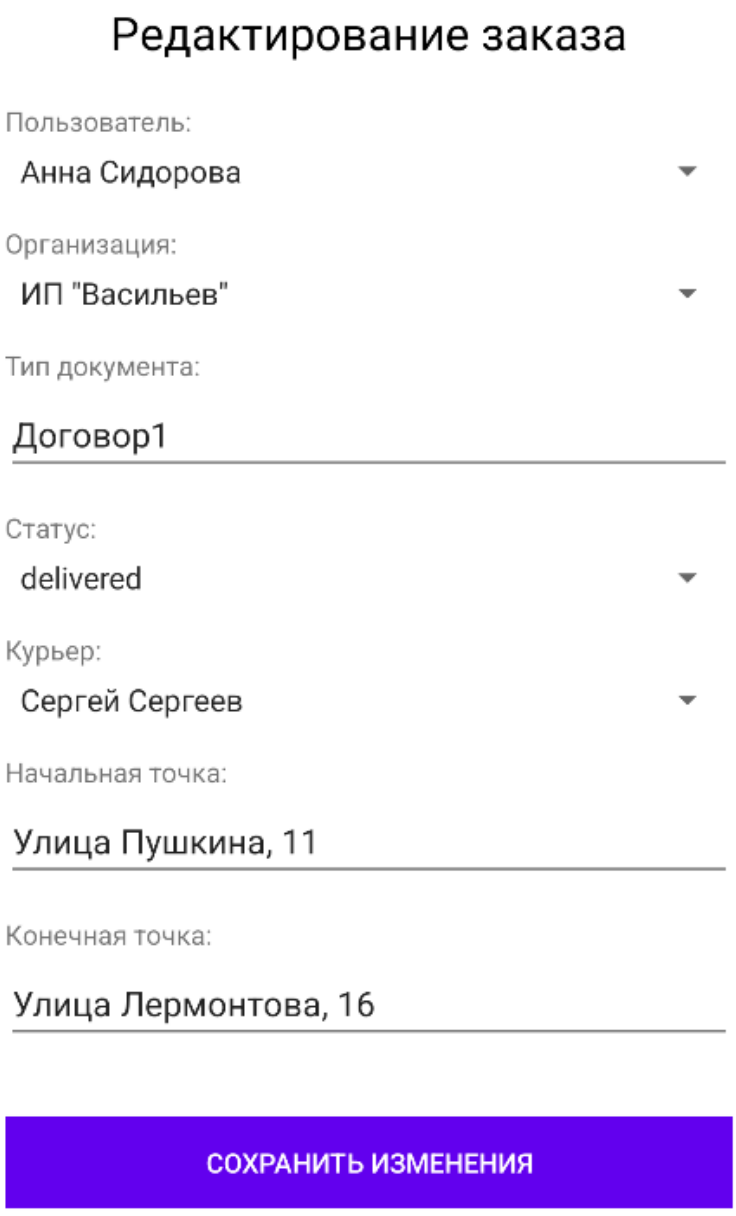

Рисунок 31 – Изменение данных о заказе

Изменение данных о заказе является важной функциональной возможностью, которая требуется в различных рабочих ситуациях. Чаще всего необходимость внесения правок возникает при обнаружении ошибок,

допущенных на этапе оформления заявки. Типичные случаи включают некорректно указанные адресные данные или тип документа.

Экспериментальная проверка работоспособности разработанного программного комплекса продемонстрировала его полное соответствие установленным техническим условиям и эксплуатационным нормативам. В процессе всестороннего анализа были детально исследованы основные бизнес-процессы приложения: процедура аутентификации, механизмы формирования и корректировки заказов, функционал распределения задач между курьерами, формирование документальных отчетов, а также особенности автономной работы без сетевого подключения.

Приоритетным направлением испытаний стала верификация системы разграничения доступа для различных категорий пользователей (администраторы, диспетчеры, курьеры). Полученные результаты убедительно доказали эффективность реализованной модели авторизации и контроля полномочий. Замеры быстродействия подтвердили соблюдение требований к скорости отклика интерфейсных элементов, тогда как использование локального хранилища информации гарантировало бесперебойную работу в условиях отсутствия интернет-соединения.

Практическая демонстрация функциональных возможностей системы с визуализацией рабочих экранов наглядно показала продуманность и логичность пользовательских интерфейсов. Каждый элемент управления и форма ввода данных демонстрировали предсказуемое поведение, соответствующее типовым сценариям взаимодействия. Особого внимания заслуживает тщательно проработанный механизм внесения корректировок в заказы, учитывающий специфику операционной деятельности сотрудников службы доставки.

Выводы по третьей главе

В данной главе представлены результаты тестирования разработанного приложения и описаны его основные функции.

Заключение

Выпускная квалификационная работа была посвящена комплексному анализу бизнес-процессов курьерской доставки документов в ООО «Газпромнефть Информационно-Технологический оператор» и разработке инновационного программного решения для их автоматизации. В ходе исследования были достигнуты значимые результаты, имеющие как теоретическую, так и практическую ценность для предприятия.

Проведенный анализ выявил ключевые проблемы существующей системы:

- отсутствие единой цифровой платформы управления заказами;
- необходимость ручного распределения заданий;
- использование устаревших методов подтверждения доставки;
- трудоемкий процесс формирования отчетности.

В ходе выполнения ВКР было разработано и реализовано:

- мобильное приложение для платформы Android на языке Java;
- механизм цифрового подтверждения доставки;
- инструмент автоматической генерации отчетов;
- концептуальная модель данных (ERD) с шестью ключевыми сущностями.

Основные достижения работы:

- создание полнофункционального программного решения;
- разработка алгоритмов автоматизации ключевых процессов;
- реализация системы защиты данных.

Перспективы развития:

- внедрение геолокационных сервисов;
- интеграция с корпоративными CRM и ERP-системами;
- масштабирование решения на другие предприятия отрасли.

Практическая значимость работы подтверждается:

- готовностью решения к пилотному внедрению;

- возможностью адаптации для других компаний;
- соответствием современным стандартам цифровизации.

Выполненная работа полностью соответствует требованиям образовательного стандарта и демонстрирует:

- глубокое понимание предметной области;
- владение современными технологиями разработки;
- способность решать сложные практические задачи;
- готовность к профессиональной деятельности в сфере ИТ.

Результаты исследования могут быть использованы как для дальнейшего развития системы в «Газпромнефть ИТО», так и в качестве основы для научных работ в области автоматизации бизнес-процессов.

Таким образом, результат выполнения ВКР демонстрирует успешное решение поставленных задач и открывают новые перспективы для дальнейшего совершенствования системы документооборота в компании.

Разработанное программное решение готово к пилотному внедрению в одном из региональных подразделений с последующим масштабированием на всю компанию.

Список используемой литературы и используемых источников

1. 10 принципов разработки мобильных интерфейсов. - CMS magazine [Электронный ресурс] - URL: https://cmsmagazine.ru/journal/items-10-principles-mobile-interface-design/, (дата обращения: 4.05.2025)
2. Блох, Дж. Java. Эффективное программирование / Дж. Блох. — Москва: Вильямс, 2021. — 464 с.
3. Васильев, А. Н. Программирование на Java для начинающих / А. Н. Васильев. — СПб.: Наука и Техника, 2022. — 480 с.
4. Гагарина, Л. Г. Разработка и эксплуатация баз данных: учебное пособие / Л. Г. Гагарина, В. В. Киселев. — Москва: Форум, 2021. — 320 с.
5. Гамма, Э. Приемы объектно-ориентированного проектирования / Э. Гамма и др. — СПб.: Питер, 2022. — 512 с.
6. Гриффитс, Д. Изучаем SQL / Д. Гриффитс. — Москва: Эксмо, 2022. — 352 с.
7. Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт. — Москва: Вильямс, 2022. — 1328 с.
8. Дронов, В. А. Android. Программирование для профессионалов / В. А. Дронов. — Москва: Питер, 2023. — 608 с.
9. Иванов, П. С. Оптимизация SQL-запросов в современных СУБД / П. С. Иванов // Информационные технологии. — 2022. — № 5. — С. 45–52.
10. Использование диаграмм вариантов использования UML при проектировании программного обеспечения. [Электронный ресурс]. https://habr.com/ru/articles/566218/, (дата обращения 4.05.2025)
11. Карпова, И. П. Базы данных: модели, разработка, реализация / И. П. Карпова. — Санкт-Петербург: Питер, 2022. — 416 с.
12. Коннолли, Т. Базы данных: проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли, К. Бегг. — Москва: Вильямс, 2021. — 1440 с.

13. Кузнецов, С. Д. Основы баз данных: учебник / С. Д. Кузнецов. — Москва: Интернет-университет информационных технологий, 2023. — 484 с.

14. Кузнецова, М. А. Современные подходы к проектированию баз данных / М. А. Кузнецова // Вестник компьютерных наук. — 2022. — № 4. — С. 12–20.

15. Лебедев, Д. В. Использование Room Persistence Library в Android / Д. В. Лебедев // Программирование и сети. — 2021. — № 6. — С. 55–62.

16. Макконнелл, С. Совершенный код / С. Макконнелл. — Москва: БХВ-Петербург, 2020. — 896 с.

17. Мартин, Р. Чистый код: создание, анализ и рефакторинг / Р. Мартин. — Москва: Диалектика, 2023. — 464 с.

18. Петрова, Е. В. Разработка мобильных приложений с использованием SQLite / Е. В. Петрова // Программная инженерия. — 2021. — № 3. — С. 28–35.

19. Роббинс, А. Linux. Командная строка / А. Роббинс. — Москва: БХВ-Петербург, 2020. — 720 с.

20. Сидоров, А. К. Безопасность данных в Android-приложениях / А. К. Сидоров // Кибербезопасность. — 2023. — № 2. — С. 67–74.

21. Таненбаум, Э. Современные операционные системы / Э. Таненбаум, Х. Бос. — СПб.: Питер, 2021. — 1120 с.

22. Фаулер, М. Рефакторинг: улучшение существующего кода / М. Фаулер. — СПб.: Питер, 2021. — 448 с.

23. Федотов, А. М. Android. Разработка мобильных приложений / А. М. Федотов. — СПб.: БХВ-Петербург, 2022. — 560 с.

24. Хорстманн, К. Java. Библиотека профессионала / К. Хорстманн. — Москва: Диалектика, 2023. — 864 с.

25. Android API Reference : справочник по API Android [Электронный ресурс] // Android Developers. – URL: https://developer.android.com/reference (дата обращения: 15.04.2024).

26. Android Developers : официальная документация для разработчиков Android [Электронный ресурс]. – URL: https://developer.android.com (дата обращения: 15.04.2025).

27. Material Design Guidelines : принципы Material Design [Электронный ресурс]. – URL: https://material.io (дата обращения: 15.04.2024).

28. Oracle Java Documentation : официальная документация Java [Электронный ресурс]. – URL: https://docs.oracle.com/javase/ (дата обращения: 15.04.2024).

29. Room Database Guide : руководство по Room Persistence Library [Электронный ресурс] // Android Developers. – URL: https://developer.android.com/training/data-storage/room (дата обращения: 15.05.2024).

30. SQLite Documentation : руководство по SQLite [Электронный ресурс]. – URL: https://www.sqlite.org/docs.html (дата обращения: 15.04.2025).