МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____ 2020 г.

Разработка приложения «Частный курьер» на платформе Android

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____И.Л. Надточий
«___»_____2020 г.

Автор работы,
студент группы КЭ-405
_____В.Е. Лященко
«___»_____2020 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____2020 г.

Челябинск-2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____2020 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-405
Лященко Владиславе Евгеньевне
обучающейся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы: «**Разработка приложения «Частный курьер» на платформе Android» утверждена приказом по университету от 24 апреля 2020 г. №627
2. Срок сдачи студентом законченной работы: 1 июня 2020 г.
3. **Исходные данные к работе:**

– Гриффитс, Д. Head First. Программирование для Android / Дон Гриффитс, Дэвид Гриффитс; пер. с англ. Е. Матвеев. — СПб.: Питер, 2016. – 704 с.: ил. – (Head first O’Reilly). – Библиография в тексте.

– Филлипс, Б. Android. Программирование для профессионалов/Б. Филлипс, К. Стюарт, К. Марсикано, 3-е изд.— СПб.: Питер, 2017. — 688 с.: ил. — (Серия «Для профессионалов»).

– МакГрат, М. Программирование на Java для начинающих /М. Мак-Грат; [пер. с англ. М.А. Райтмана]. – Москва: «Э», 2016. – 192 с. – (Программирование для начинающих).

– Блох, Д. Java: эффективное программирование/Д. Блох, 3-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 464 с.: ил. — Парал. тит. англ.

– Firebase guides. https://firebase.google.com/docs/android/setup.

4. Перечень подлежащих разработке вопросов:

– анализ актуальности мобильных приложений;

– рассмотрение предметной области;

– выявление проблемы в предметной области;

– обзор и сравнение существующих решений;

– проектирование собственного решения в виде мобильного приложения на платформе Android;

– реализация собственного решения;

– тестирование разработанного приложения.

5. Дата выдачи задания: 1 декабря 2020 г.

Руководитель работы _____/ *И.Л. Надточий* /

Студент _____/ *В.Е. Лященко* /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	01.03.2020	
Разработка модели, проектирование	01.04.2020	
Реализация системы	01.05.2020	
Тестирование, отладка, эксперименты	15.05.2020	
Компоновка текста работы и сдача на нормоконтроль	24.05.2020	
Подготовка презентации и доклада	30.05.2020	

Руководитель работы _____/ *И.Л. Надточий* /

Студент _____/ *В.Е. Лященко* /

Аннотация

В.Е. Лященко. Разработка приложения «Частный курьер» на платформе Android. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2020, 70 с., 35 ил., библиогр. список – 25 наим.

В рамках выпускной квалификационной работы рассматривается разработка мобильных программных продуктов и ее актуальность на текущий момент времени. Рассматривается предметная область доставки посылок и проблема организации частной доставки. Производится анализ существующих решений в виде мобильных приложений на платформе Android.

Целью работы является разработка собственного на платформе Android-приложения для организации частной доставки, учитывающее недостатки существующих решений. Для этого рассматриваются инструменты создания мобильных приложений. Производится подбор оптимальных технологических решений для решения поставленной задачи.

Вслед за этапами проектирования и выбора технологических решений начинается разработка Android-приложения на языке программирования Java. При разработке используется база данных класса NoSQL Firebase Realtime Database и дополнительные сервисы Firebase.

По завершении этапа разработки, производится альфа-тестирование разработанного приложения при помощи: smoke-тестирования, UAT-тестирования и функционального тестирования.

Результатом работы является Android-приложение, отличающееся от представленных ранее удобством использования и направленностью на потребности частных лиц. Приложение получило название «Pick And Go».

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

На сегодняшний день мобильные устройства стали неотъемлемой частью жизни человека. Данные о количестве активных смартфонов в мире из различных источников сильно отличаются так как довольно сложно учесть все нюансы подсчета. Однако, по некоторым оценкам, на 2020 год уже 75% жителей Земли используют смартфоны. На 2017 год количество смартфонов на платформе Android достигло двух миллиардов.

Мобильные устройства уже давно являются не только средством коммуникации, но и выполняют многие задачи, которые ранее выполнялись только персональными компьютерами. Возможности мобильных устройств постоянно расширяются. Огромное значение в этом процессе имеют мобильные приложения, позволяющие пользователю расширить функционал устройства по своему усмотрению. Значительная часть существующих ныне приложений написана на языке Java. Приложения имеют самые разные направленности и решают проблемы из множества предметных областей. Одним из самых распространенных ныне видов мобильных приложений, являются приложения, решающие практические повседневные задачи, позволяющие: найти способы передвижения, просмотреть ассортимент товаров магазина, оформить заказ на доставку и многое другое. Для решения повседневных задач, возникающих у человека везде и всегда, мобильные приложения подходят как нельзя лучше, так как смартфон почти всегда бывает «под рукой».

У каждого человека возникала необходимость что-то перевезти или передать. И очень часто в таком случае возникает вопрос о том, как удовлетворить такую потребность. Современному человеку наверняка придет на ум идея найти мобильное приложение, позволяющее решить его вопрос. Однако приложения, позволяющие оформить доставку, чаще всего создаются для компаний, чья деятельность связана с

доставкой посылок. Это весьма удобно и выгодно для передачи посылок от интернет-магазинов и иных юридических лиц, часто использующих услуги доставки, до их клиентов. Подобная выгода обусловлена тем, что почти всегда юридические лица работают со службами доставки на особых условиях, так как регулярно пользуются данным видом услуг. Службы доставки не предлагают особые условия частным лицам, а потому использование их услуг зачастую бывает невыгодно. Немаловажным фактором является и время доставки, которое может значительно растянуться в связи с выходными днями, попадающими в период доставки, и периодичностью доставки. Также следует отметить, что в некоторые населенные пункты невозможно отправить посылку в связи с отсутствием филиала компании по доставке. По вышеперечисленным причинам возникает потребность в передаче посылок по договоренности с частными лицами, намеревающимися посетить населенный пункт, в который требуется доставка. Таким образом, возникает необходимость в мобильном приложении для организации частной доставки.

С учетом того факта, что за срочную доставку компаниями взымается повышенная плата, приложение могло бы решить не только проблему планирования практических задач, но и представляло бы экономический интерес для пользователей.

На данный момент уже существуют некоторые программные решения для оформления запросов и предложений на доставку. Однако, они не являются удобными для частной доставки и, по большей части, ориентированы на сбор предложений от различных компаний доставки и/или неудобны в использовании.

Таким образом, целью представленной выпускной квалификационной работы является разработка и реализация удобного и функционального приложения на платформе Android, предоставляющего возможность поиска частных лиц, с целью достижения договоренности о перевозке посылок из одного населенного пункта в другой.

Для достижения поставленной цели, необходимо решить следующие задачи:

1. Анализ существующих аналогов, посредством самостоятельного тестирования аналогичных программных решений и просмотра отзывов пользователей;
2. Выявление необходимого для удобства пользователей и выполнения задач функционала;
3. Формирование архитектуры проекта;
4. Формирование принципов заполнения базы данных для удобства выполнения запросов;
5. Анализ средств разработки и выбор оптимально подходящих средств для быстрой и качественной реализации проекта, позволяющих обеспечить гибкость решения;
6. Реализация приложения с учетом недостатков рассмотренных приложений-аналогов;
7. Определение подходящих методик тестирования и тестирование разработанного приложения на отказоустойчивость, корректность выполнения задач, удобства интерфейса.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор аналогов

Как было сказано ранее, на сегодняшний день уже существуют мобильные приложения для организации частной доставки. Для выявления преимуществ и недостатков приложений-аналогов, было найдено три подобных приложения, а именно: «Travel Post» [1] и «BlaBlaCar» [2], размещенные в магазине приложений Google Play Store [3], и «Getberry» [4].

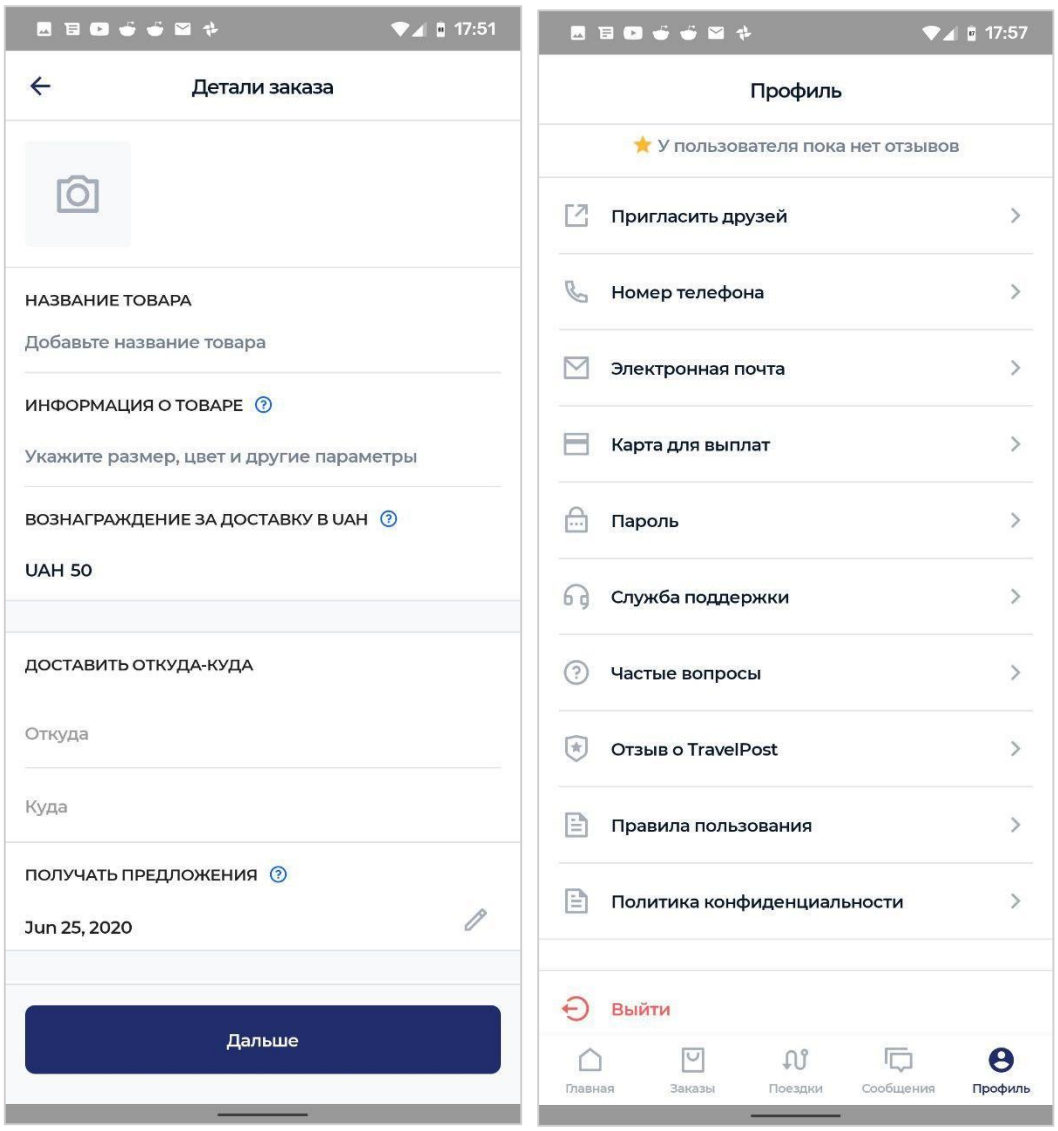
Следует отметить, что основной специализацией «BlaBlaCar» (версия 5.52.0) являются пассажирские перевозки. Оно не предоставляет удобных инструментов для оформления частной доставки и лишь позволяет найти потенциального доставщика и связаться с ним. Создатели приложения официально заявляют, что приложение не имеет и не будет иметь в дальнейшем подобного функционала [5].

Приложение «TravelPost» (версия 3.1.0) обладает необходимым функционалом для оформления запросов и предложений на частную доставку, но не имеет удобного интерфейса и не соответствует ожиданиям пользователей, согласно отзывам в Google Play. Несмотря на поддержку всех городов мира, приложение не позволяет указать валюту для вознаграждения. В «TravelPost» поддерживается только валюта UAH - Украинская гривна (рис 1.1, а). Настройки приложения также не подразумевают выбора валюты (рис 1.1, б). Таким образом, если пользователь не проживает в Украине, расчеты в непривычной валюте являются большим неудобством для него.

Важно и то, что за размещение запроса на доставку взымается комиссионный сбор, начисляемый поверх величины указанного вознаграждения (рис 1.2), что также является неприятностью для пользователя.

Еще одной особенностью оформления запроса на доставку является требование ввода несущественных параметров доставки (например, цвет товара) и точных параметров высоты, длины и ширины, что также затрудняет заполнение

формы и отнимает много времени. Такая подробность имеет место при заполнении форм для отправки посылок компаниями, ориентированными на доставку. Частная же доставка, должна сделать процесс отправки посылок легче и удобнее - без необходимости описания мелких подробностей.



а) б)

Рисунок 1.1 – Снимки экрана из приложения «Travel Post»

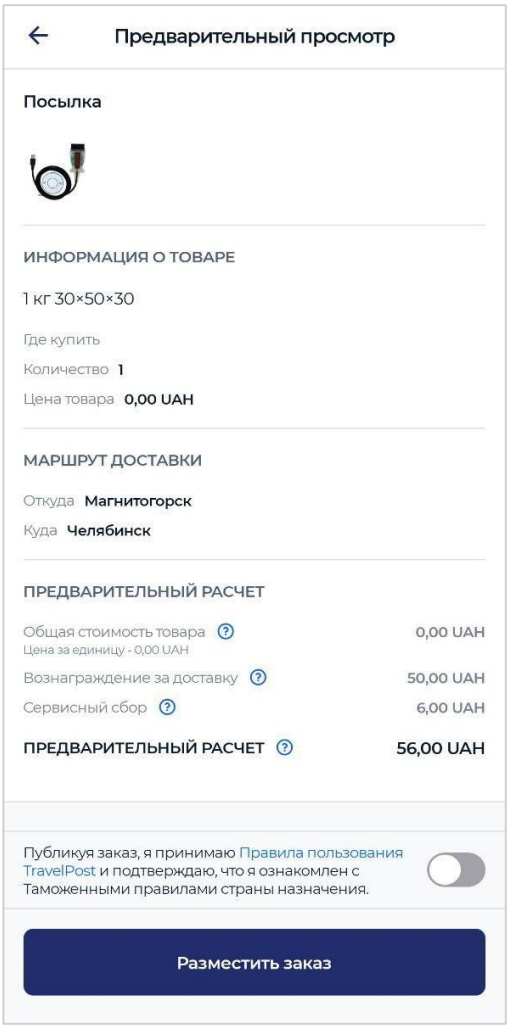


Рисунок 1.2 – Расчет стоимости доставки в приложении «Travel Post»

Также при поиске и исследовании аналогов было найдено приложение «Getberry», которое, согласно описанию, реализует нужный для частной доставки функционал. Однако по ссылкам для загрузки оно оказалось недоступно. Таким образом приложение «Getberry» протестировать не удалось.

Согласно результатам обзора аналогов, на данный момент не существует достаточно удобного инструмента оформления частной доставки. Для решения этой проблемы требуется создать новое приложение, достаточно гибкое, чтобы учесть интересы большинства пользователей. А так как доставка всегда сопряжена с мобильностью, создание приложения на платформе Android будет очень уместно в данном случае.

1.2 Анализ основных технологических решений

Среда разработки

Android Studio

Официальным средством разработки под операционную систему (ОС) Android является Android Studio, разработанная компанией Google [6].

Android Studio – интегрированная среда разработки, основанная на программном обеспечении IntelliJ IDEA компании JetBrains. IntelliJ IDEA в первую очередь направлена на ускорение разработки ПО, Android Studio, основанная на нем, также позволяет создавать программное обеспечение быстро и просто.

Разработка приложений в Android Studio может производиться на языках программирования Java и Kotlin. Для разметки используется язык XML.

Android Studio имеет большое количество инструментов для отладки и тестирования приложений и является свободно распространяемой. К используемым во время разработки инструментам Android Studio можно отнести:

1. Система автоматической сборки Gradle [7];
2. Удобный многофункциональный эмулятор мобильных устройств;
3. Интеграция с GitHub [8];
4. Интеграция с сервисами Firebase;
5. Инструменты анализа производительности, совместимости версий и иных особенностей;
6. Шаблоны основных макетов и компонентов.

Eclipse

Eclipse – это бесплатная среда разработки от организации Eclipse Foundation [9]. Благодаря активному развитию, а также поддержке со стороны компании и сторонних разработчиков, на данный момент у этой IDE имеются следующие достоинства:

1. Доступность интерфейса и документации на Русском языке;
2. Достойная производительность на маломощных персональных компьютерах (ПК);
3. Большое количество дополнений, доступных к установке;
4. Возможность подключения модулей;
5. Возможность групповой разработки.

Eclipse имела большую популярность несколько лет назад и считалась монополистом на рынке IDE для Android. Однако в связи с выходом Android Studio, в 2014 г. Google приостановила поддержку Eclipse в качестве основной среды для разработки приложений под ОС Android.

Классы баз данных

Реляционные базы данных

Реляционная база данных — это совокупность взаимосвязанных таблиц, каждая из которых хранит информацию об объектах некоторого типа. Строка таблицы содержит данные об одном объекте (например, номере гостиницы, постояльце), а столбцы таблицы описывают различные характеристики этих объектов — атрибутов (например, сведения о постояльце). Записи, т. е. строки таблицы, имеют одинаковую структуру, — они состоят из полей, хранящих атрибуты объекта. Каждое поле, (столбец), всегда описывает одну характеристику объекта и всегда имеет строго определенный тип данных. Все записи в таблице имеют одинаковые поля и их количество, различие зависит только от свойств описанного объекта.

В реляционной базе данных каждая таблица имеет первичный ключ — поле или комбинацию полей, которые однозначно идентифицируют запись. Если ключ включает нескольких полей, он называется составным. Ключ должен обладать свойством уникальности, чтобы по значению ключа имелась возможность отыскать единственную запись.

Таблицы реляционной базы данных (БД) должны отвечать требованиям нормализации отношений. Нормализация отношений — это формальный аппарат ограничений на формирование таблиц, который устраняет возможность дублирования, обеспечивает непротиворечивость дынных и уменьшает трудоемкость ведения базы данных.

Реляционные БД находят применение повсеместно: в организациях для учёта персонала, ведения бухгалтерии, учёта постояльцев гостиницы, поставщиков, партнёров, клиентов, в адресных и телефонных книгах, словарях, справочниках. Но все эти варианты использования объединяет одно свойство – строгая типизация данных, что не свойственно приложениям, осуществляющим планирование практических задач и имеющим гибкую структуру данных.

Нереляционные базы данных

Базы данных NoSQL хорошо подходят для многих современных мобильных приложений, как автономных, так и требующих подключения к интернету, когда требуются гибкие масштабируемые базы данных с высокой производительностью и широкими функциональными возможностями, способные обеспечивать максимальное удобство использования.

Как правило, базы данных NoSQL предлагают гибкие схемы, что позволяет осуществлять разработку быстрее и обеспечивает возможность поэтапной реализации. Благодаря использованию гибких моделей данных БД NoSQL хорошо подходят для частично структурированных и неструктурированных данных.

Базы данных NoSQL оптимизированы для конкретных моделей данных и шаблонов доступа, что позволяет достичь более высокой производительности по сравнению с реляционными базами данных.

Базы данных NoSQL предоставляют API и типы данных с широкой функциональностью, которые специально разработаны для соответствующих моделей данных.

СУБД

Firebase

Сервис Firebase предоставляет разработчику целый пакет облачных услуг [10]. Но его основой является облачная система управления базой данных (СУБД) класса NoSQL, позволяющая не только хранить, но и синхронизировать данные между клиентами. В сервисе поддержаны особенности интеграции с приложениями под операционные системы Android и iOS, реализовано API для приложений на JavaScript, Java, Objective-C и Node.js. Предусмотрено API для шифрования данных. Для переноса данных предусмотрено автоматическое преобразование базы данных в JSON объект.

Для хранения данных приложения в Firebase используется база данных типа NoSQL Realtime Database. Данные хранятся и передаются в текстовом формате JSON. JSON-текст представляет собой (в закодированном виде) одну из двух структур:

1. Набор пар ключ-значение. Ключом может быть только регистрозависимая строка, значением – любой объект;
2. Упорядоченный набор значений.

Листинг 1.1 демонстрирует пример JSON-представления объекта, загруженный из БД Firebase Realtime Database. Согласно листингу, в объекте содержатся значения различных типов, а также вложенные объекты.

Листинг 1.1 - Пример объекта в формате JSON

```
"Users" : {
  "A26t6m4izlZNw3oDLHPL7iofaM93" : {
    "Messages" : {
      "-M8PafvFKYW0dlIzE6-v" : {
        "sendDate" : 1590657991088,
        "status" : false,
        "text" : "Добро пожаловать, Евгений! Спасибо за выбор нашего приложения."
      }
    },
    "Requests" : {
      "-M8Pcb1JWBJAuyXMfwjk" : {
        "award" : "600₽",
```

Окончание листинга 1.1


```
        "capacity" : 1,
        "clippedTo" : "-M8PdGqYdEGkDEyNexx7",
        "comment" : "Небольшой пакет",
        "deliverDeadline" : 1591358400000,
        "destPoint" : "Уфа",
        "fromHouse" : false,
        "inTime" : false,
        "pickPoint" : "Челябинск",
        "recipientName" : "Ирина",
        "recipientNumber" : "9532417568",
        "sendDeadline" : 1591297200000,
        "status" : 5,
        "toHouse" : false,
        "weight" : 2
      }
    },
    "age" : 137669129919,
    "emailVerified" : true,
    "gender" : true,
    "incomingReqNumber" : 0,
    "joined" : 1590657991086,
    "messageNumber" : 0,
    "name" : "Евгений",
    "numVerified" : false,
    "phone" : "9536241789",
    "photo" : "https://firebasestorage.googleapis.com/v0/b/pickandgo-
fb204.appspot.com/o/man.png?alt=media&token=d5694b86-19e4-4c04-8d32-32daa8054eee",
    "ratingSum" : 0,
    "reviewsNumber" : 0
  }
```


Рисунок 1.3 демонстрирует представление JSON - объекта, описанного выше, в консоли Firebase.

Рисунок 1.3 - Представление объекта в консоли Firebase

Firebase Realtime Database поддерживает следующие типы данных: Null, Boolean, Integer, floating-point, Date, Text string, Byte, Cloud Firestore references, Geographical point, Array, Map.

Объекты в Firebase Realtime Database представляются как документы, расположенные в каталогах. Каталог может содержать несколько документов и подкаталогов.

Ключевой возможностью базы данных Realtime Database облачной платформы Firebase является работа в реальном времени. Она не использует HTTP-запросы, а использует синхронизацию данных, что позволяет всем подключенным клиентам получить измененную информацию в считанные секунды.

Firebase также имеет собственную систему регистрации и аутентификации, позволяющую проводить аутентификацию несколькими способами, например, при помощи адреса электронной почты, номера телефона, аккаунтов Google, Facebook и Twitter.

Для хранения пользовательских файлов, таких как изображение профиля, используется облачное хранилище данных Firebase Cloud Storage. Файлы также сохраняются на устройстве, что избавляет пользователя от необходимости загружать одни и те же документы слишком часто, растрачивая мобильный трафик. Естественно, данные не превышают объем памяти, выделенный приложению. Помимо экономии мобильного трафика, это позволяет более рационально использовать оперативную память устройства, выгружая неиспользуемые файлы. При необходимости, файлы могут быть загружены в ОЗУ с устройства, даже при отсутствии подключения.

В случае использования Firebase Realtime Dtabase для поддержки PUSH-уведомлений потребуется использовать Firebase Cloud Messaging (FCM). В совокупности с сервисом Firebase Clud Functions, призванным реагировать на те или иные события в БД, FCM позволит оповещать пользователя при появлении интересной для него информации.

Firebase Cloud Messaging (FCM) – кроссплатформенное решение, позволяющее добавить в мобильное приложение поддержку PUSH уведомлений.

FCM позволяет отправлять PUSH уведомления на все устройства разных платформ с использованием API или же панели инструментов.

В добавок ко всему FCM имеет огромное количество инструментов для отладки приложения и аналитики, например, позволяет отследить, на каком экране пользователи проводят больше всего времени или же из какой страны, города и даже улицы пользователи запускают приложение.

MongoDB

MongoDB разработан и управляется MongoDB Inc [11]. Это база данных NoSQL с открытым исходным кодом. В отличие от Firebase, который предлагает полную экосистему сервисов, MongoDB является только (очень мощной) базой документов.

Масштабируемость и гибкость - два фактора, которые учитывались при разработке MongoDB. Он предлагает очень мощные запросы и индексацию. Несмотря на то, что MongoDB предлагает только целенаправленный сервис для хранения данных, он по-прежнему широко применяется благодаря мощным возможностям хранения, которые он предлагает. С MongoDB разработчики получают больше возможностей в разработке приложений. MongoDB учитывает их потребности в разработке, поэтому приложение эффективно хранит данные.

Язык программирования

Двумя самыми распространенными языками программирования для платформы Android являются Java и Kotlin.

Java

Java является универсальным языком на котором можно писать совершенно разные приложения. К нему уже существует множество библиотек и наработок, он отлично поддерживается средой разработки Android Studio. Также, следует отметить, что на сегодняшний день существует уже огромное сообщество Java-разработчиков, постоянно публикующих свои решения для различных вопросов по разработке.

Kotlin

Kotlin является более молодым языком и хорошо подходит для бытрой разработки за счет своих коротких, по сравнению с Java, конструкций. Он прекрасно подходит для небольших приложений.

Актуальность компонентов для поставленной задачи

Исходя из анализа существующих решений и технических возможностей, для реализации мобильного приложения на платформе Android следует выбирать среду разработки Android Studio если мощность ПК достаточно высока для работы с ресурсоемкой средой разработки. Встроенные инструменты позволят быстро и с удобством реализовать мобильное приложение, протестировать его работу и подготовить к запуску на рынок.

Мобильные приложения, создаваемые для планирования какого-либо типа практических задач, даже если они изначально спланированы до мелочей, не могут оставаться неизменными на протяжении всего периода своего существования. Зачастую они требуют доработок и внесения изменений по мере появления новых потребностей пользователей и их оценки тех или иных возможностей приложения. Целевая аудитория может быть очень широкой, что требует большой гибкости решений. В связи с тем, что требования к приложению могут внезапно измениться, следует выбирать технологические решения, позволяющие с наименьшими

трудозатратами вносить изменения в проект. Реализуемое в рамках выпускной квалификационной работы приложение не является исключением. А потому, рациональным решением будет использовать для разработки базу данных класса NoSql. Также, в связи с тем, что в процессе развития приложение может стать достаточно сложным следует выбирать универсальный язык программирования.

Так как связи между данными в разрабатываемом приложении предполагается довольно простыми, для разработки не требуется высокая мощность и функциональность СУБД. А вот использование дополнительных сервисов может значительно упростить разработку и сделать мобильное приложение удобным и функциональным. Таким образом, в роли СУБД следует выбрать платформу облачных услуг Firebase от компании Google. Она предоставляет широкие возможности облачных вычислений для разработки и бесплатный стартовый пакет. Условия этого пакета обеспечивают разработчика всем необходимым для создания приложения. Немаловажным преимуществом Firebase является то, что платформа принадлежит компании Google, владеющей и операционной системой Android. Это обеспечивает простоту интеграции и может значительно упростить процесс разработки Android-приложения за счет наличия официальной документации.

1.3 Вывод

В качестве среды разработки была выбрана среда Android Studio 3.5.2. Языком программирования стал Java. В качестве СУБД была выбрана платформа облачных услуг Firebase, предоставляющая возможность работы с БД Firebase Realtime Database класса NoSQL и сопутствующими сервисами.

2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

Для реализации приложения необходим следующий набор подсистем:

1. База данных класса NoSQL Firebase Realtime Database, обеспечивающая хранение данных о пользователях и всей информации, связанной с доставкой посылок.
2. Firebase Storage обеспечивающее хранение изображений.
3. Сервисы Firebase (Firebase Authentication, Firebase Cloud Messaging, Firebase Cloud Functions), обеспечивающих авторизацию, регистрацию, а также оповещение пользователей путем отправки PUSH-уведомлений.
4. Бизнес-логика приложения – организация обработки данных и их размещение в базе данных.
5. Графический интерфейс клиентского приложения.

2.1 Функциональные требования

В ходе проектирования, разработки и анализа технического задания были определены следующие функциональные требования.

Для получения доступа к функционалу пользователя требуется иметь аккаунт в системе, т. е. пройти процедуру регистрации. Пользователь должен иметь возможность выступать в двух ролях: отправитель и доставщик, используя для этого один аккаунт. Аккаунты не должны делиться на типы доставщик и отправитель.

Независимо от роли пользователь должен иметь возможность:

1. Зарегистрироваться;
2. Авторизоваться;
3. Добавить и изменить информацию профиля;
4. Перейти к набору номера или сообщения при просмотре профиля другого пользователя;

5. Оценить взаимодействие с пользователем и оставить отзыв с указанием роли в которой выступил пользователь (доставщик/отправитель).

В качестве доставщика пользователь должен иметь возможность:

1. Предложить доставку;
2. Просмотреть список исходящих предложений;
3. Редактировать предложение о доставке (только если отсутствуют запросы на доставку);
4. Просмотреть список входящих запросов, закрепленных за предложением;
5. Просмотреть детали входящего запроса;
6. Просмотреть профиль пользователя, отправившего запрос на доставку;
7. Перейти к набору номера получателя посылки, указанного в запросе на доставку отправителем;
8. Принять/отклонить входящий запрос;
9. Получить уведомление об отмене принятого запроса;
10. Скрыть предложение после достижения договоренности с одним/несколькими пользователями с целью не получать новых договоренностей о доставке;
11. Удалить предложение о доставке.

В качестве отправителя пользователь должен иметь возможность:

1. Создать запрос на доставку;
2. Найти предложение о доставке (сразу после ввода всех данных по запросу на доставку должен осуществляться поиск подходящих предложений) и отправить запрос автору одного из предложений;
3. Получить уведомление о том, что запрос принят/отклонен;
4. Получить уведомление в случае отмены доставки;

5. Сохранить запрос на доставку (если не нашлось вариантов доставки сразу после оформления запроса);
6. Просмотреть список исходящих запросов;
7. Просмотреть детали исходящего запроса;
8. Просмотреть предложение, за которым закреплен запрос;
9. Редактировать запрос на доставку;
10. Удалить запрос на доставку (если запрос был закреплен за предложением, автор предложения должен быть уведомлен);
11. Осуществить поиск предложений для запроса в статусе «создан» через некоторое время после его создания;
12. Просмотреть профиль пользователя, предлагающего доставку.

2.2 Нефункциональные требования

1. Интерфейс приложения должен быть интуитивно понятным;
2. Мобильное приложение должно функционировать на устройствах с операционной системой Android версии не ниже 9.0;
3. Мобильное приложение должно быть доступно только в портретной ориентации.

3 ПРОЕКТИРОВАНИЕ

3.1 Архитектура приложения «Pick And Go»

Рисунок 3.1 изображает схему модулей, запланированных для реализации в приложении со схематичным изображением их взаимодействия.

Приложение выстраивается по архитектуре «клиент-сервер». Основой работы приложения является БД Firebase Realtime Database, к которой происходит обращение для получения данных. Доступ к базе разрешен только авторизованным пользователям. Дополнительным компонентом хранения является Firebase Storage, позволяющее загружать изображения и файлы. Для отправки PUSH-увдомлений предполагается использование сервисов Firebase Cloud Functions и Cloud Messaging. Логика приложения обеспечивает обработку и отображения полученных данных.

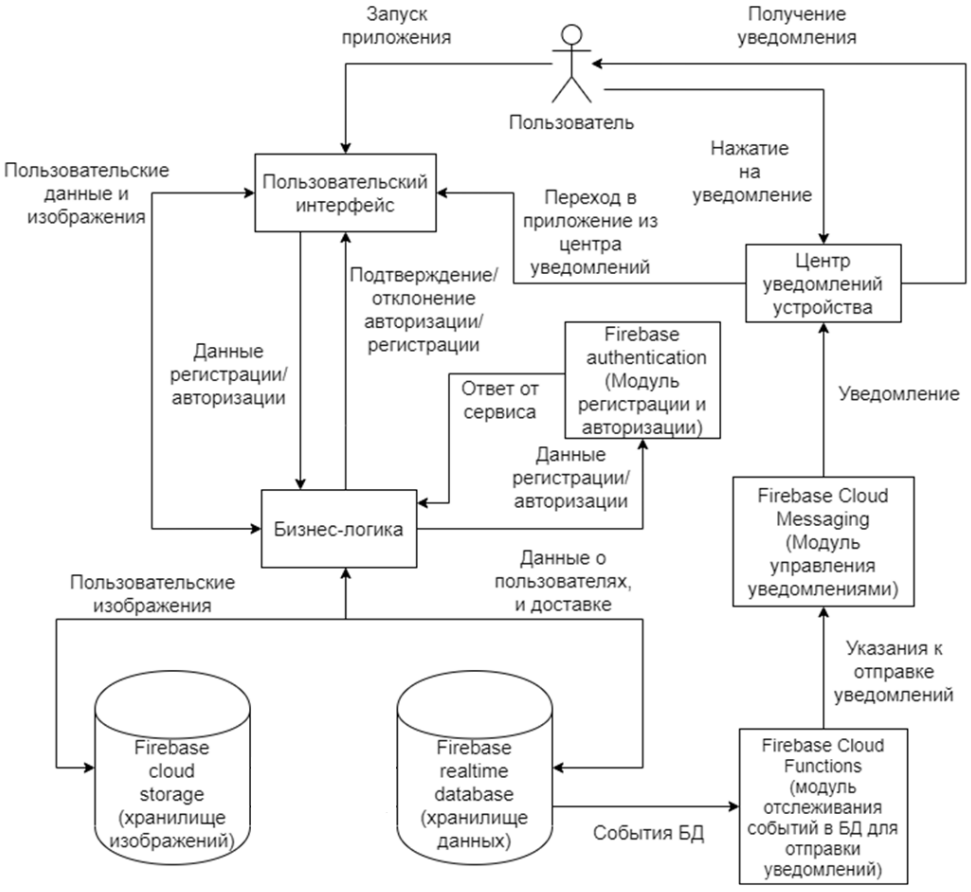


Рисунок 3.1 – Архитектура приложения

На момент представления выпускной квалификационной работы реализованы и корректно работают все модули, кроме модуля рассылки PUSH-уведомлений, который в дальнейшем будет реализован на основе сервисов Firebase Cloud Functions и Firebase Cloud Messaging.

Принцип формирования базы данных

Основной особенностью баз данных класса NoSQL является свободный принцип расположения данных. Главное, чем следует руководствоваться – удобство выполнения запросов к БД. Если встает вопрос о том куда лучше поместить данные, к которым осуществляется доступ двумя типами запросов, в приоритете должны быть те, которые выполняются чаще.

На основании данного принципа, формирование базы данных для разработанного приложения организовано следующим образом: все данные, исходящие от пользователя, являются вложенными в объект «Пользователь». Исключение составляют исходящие от пользователя предложения на доставку. Они хранятся вне объекта, содержащего данные пользователя. Такая реализация позволяет с легкостью осуществлять поиск предложений по запросу. На рис 3.2 приведено схематичное представление расположения данных в БД.

Следует отметить, что использование сервисов Firebase позволяет обеспечить полную изолированность данных авторизации от данных профиля пользователя и иной информации, вносимой пользователем при работе с приложением. Данные авторизации при этом хранятся вне базы данных. Так как вся информация, хранимая в БД может просматриваться пользователями при взаимодействии, появляется возможность не вводить дополнительных условий доступа и разрешить доступ к базе данных (в пределах функционала приложения) всем авторизованным пользователям.

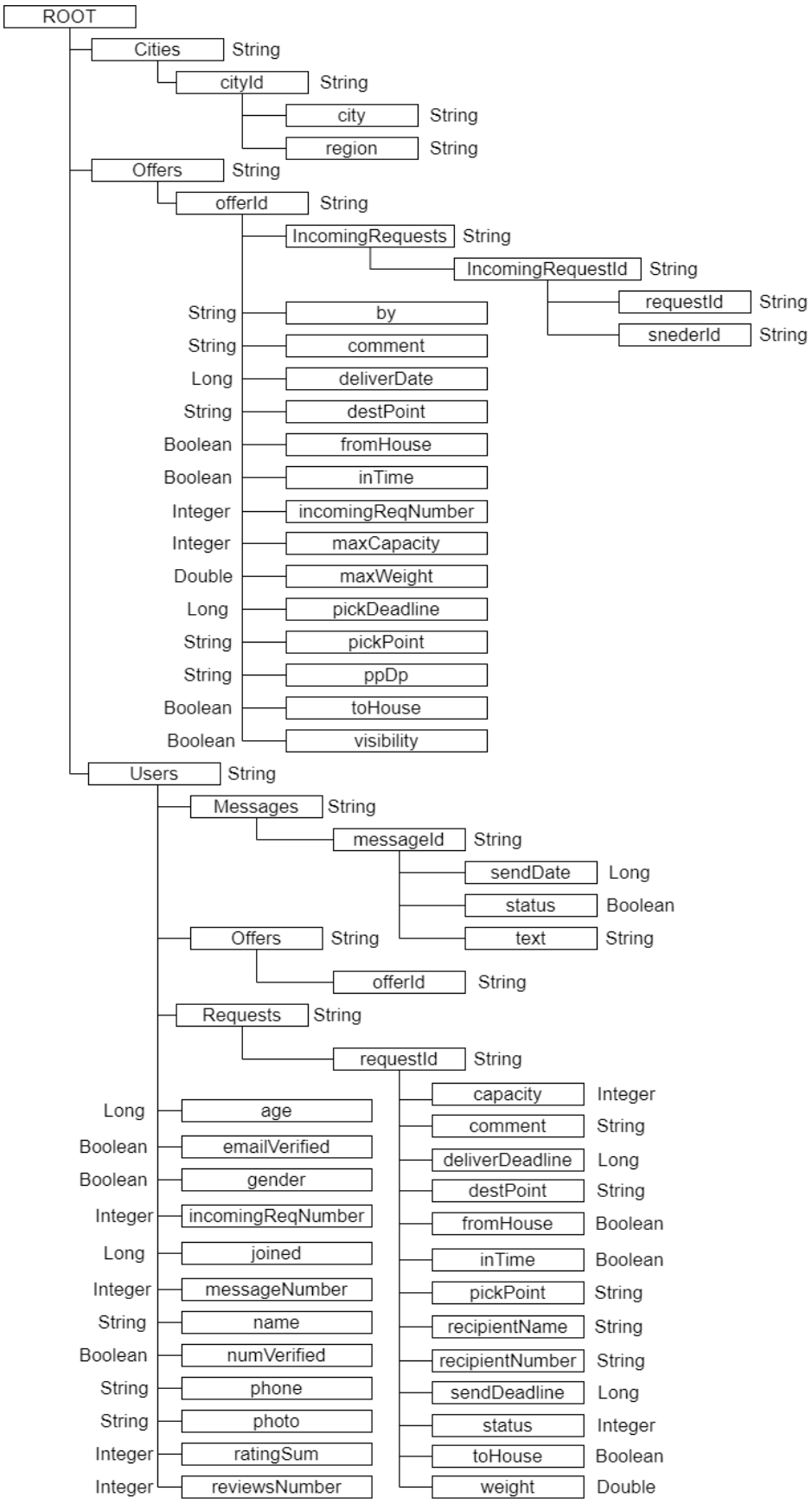


Рисунок 3.2 - Расположение данных в базе

3.2 Алгоритмы решения задачи

Основной задачей приложения является организация взаимодействия между отправителем посылки и доставщиком.

Следовательно, требуется:

1. Предоставить возможность размещать предложения и запросы на доставку, а также удалять и изменять их по необходимости;
2. Предоставить возможность поиска предложений по запросу и отправки запроса автору предложения, а также возможность ответа на запрос автором предложения;
3. Уведомлять пользователей обо всех изменениях, касающихся размещенных запросов и предложений.

Далее будут представлены алгоритмы выполнения основных бизнес-процессов. Алгоритм создания предложения представлен на рисунке 3.3. Алгоритм создания запроса и поиска предложений - на рисунке 3.4. Алгоритм удаления запроса - на рисунке 3.5. Алгоритм удаления предложения - на рисунке 3.6.

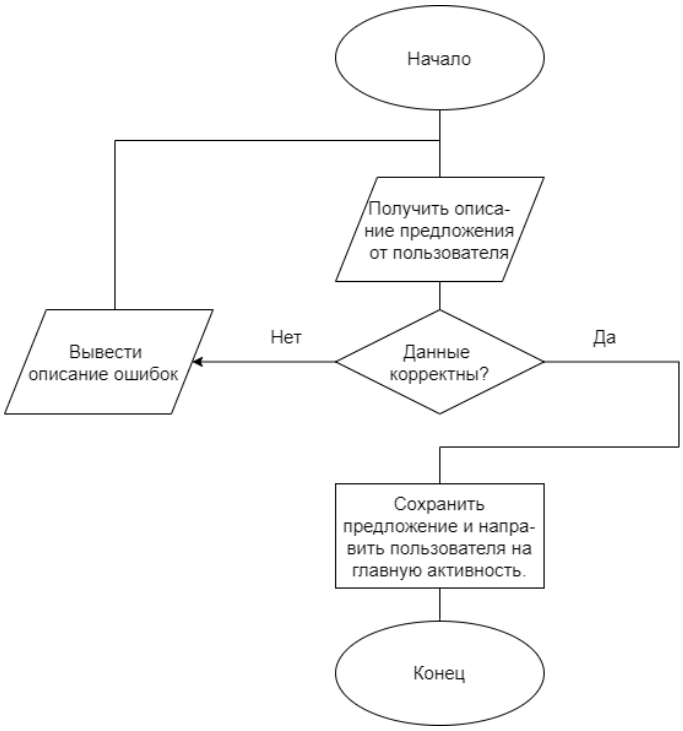


Рисунок 3.3 - Алгоритм создания предложения

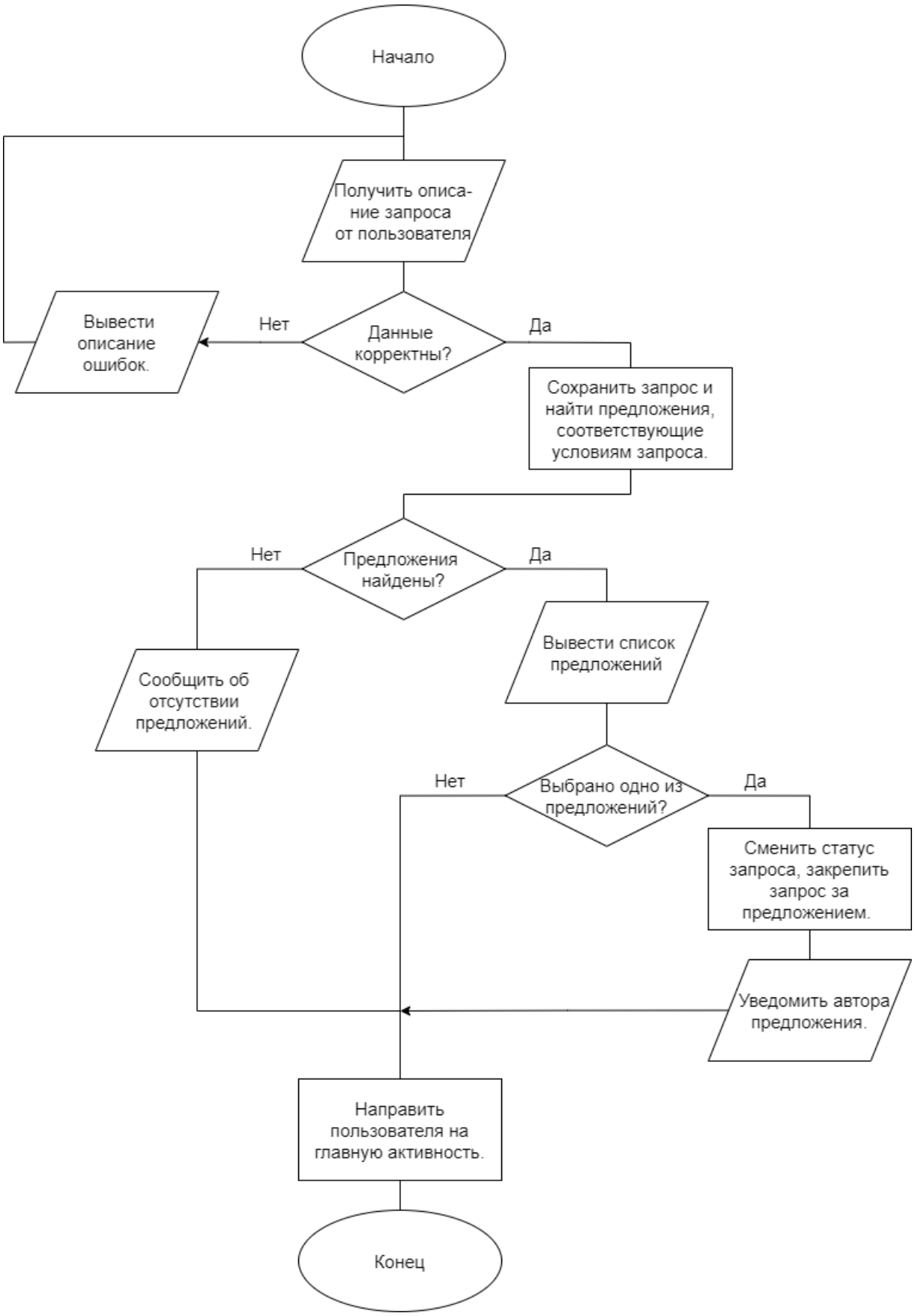


Рисунок 3.4 – Алгоритм создания запроса и поиска предложений

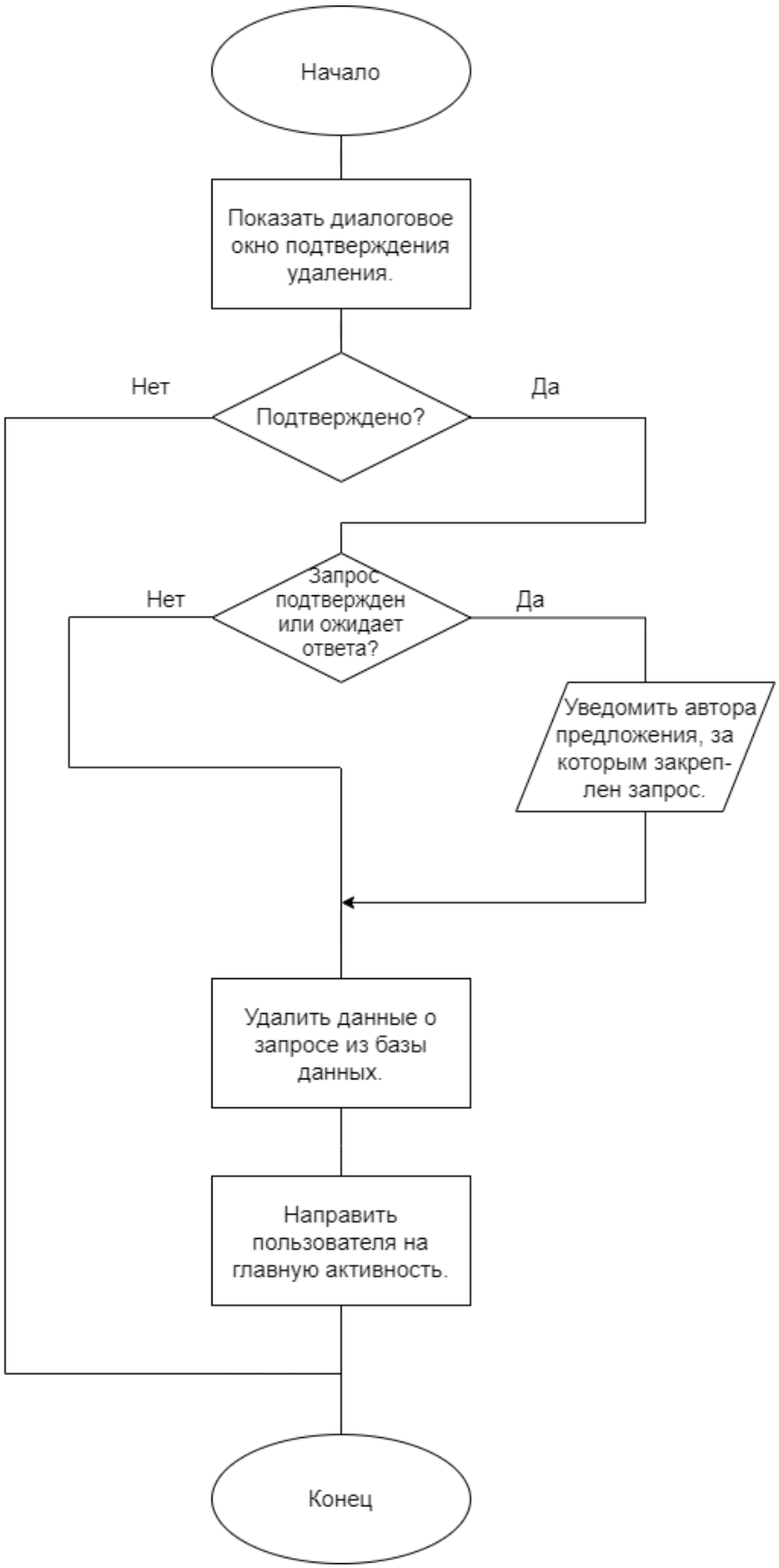


Рисунок 3.5 – Алгоритм удаления запроса

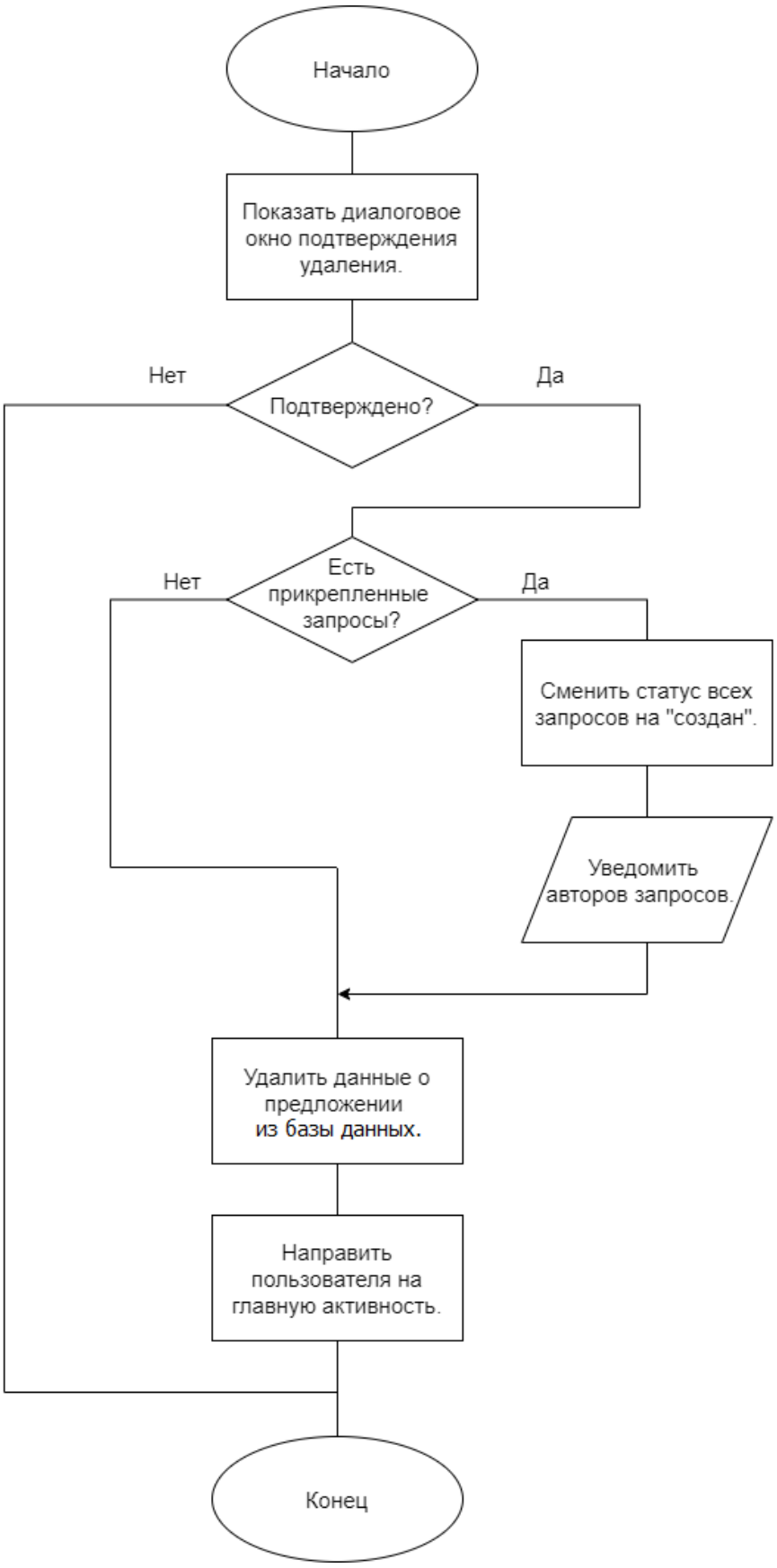


Рисунок 3.6 – Алгоритм удаления предложения

3.3 Описание данных

Входные данные

В качестве входных данных от пользователя в момент регистрации используются:

– адрес электронной почты;

– пароль;

– имя;

– дата рождения;

– номер телефона;

– пол.

В качестве входных данных от пользователя в момент авторизации используются

– адрес электронной почты;

– пароль.

В качестве входных данных в момент создания и редактирования предложения на перевозку используется:

– пункт отправления;

– пункт назначения;

– максимальная масса предметов, которые возможно перевезти;

– максимальные габариты предметов, которые возможно перевезти;

– крайние дата и время для передачи посылки доставщику отправителем;

– дата и время доставки посылки в пункт назначения;

– возможности по виду доставки (от дома отправителя до дома получателя, от точки отправления доставщика, до точки его прибытия в пункт назначения, от дома отправителя до точки прибытия доставщика в пункт назначения или от точки отправления доставщика до дома получателя);

– возможности времени доставки (по прибытию или к назначенному времени);

– комментарий (опционально).

В качестве входных данных в момент создания и редактирования запроса на перевозку используется:

– пункт отправления;

– пункт назначения;

– масса предметов, которые нужно перевезти;

– габариты предметов, которые возможно перевезти;

– крайние дата и время доставки посылки в пункт назначения;

– предпочтения по виду доставки (от дома отправителя до дома получателя, от точки отправления доставщика, до точки его прибытия в пункт назначения, от дома отправителя до точки прибытия доставщика в пункт назначения или от точки отправления доставщика до дома получателя);

– предпочтения по скорости доставки (по прибытию или к назначенному времени);

– контактные данные получателя (имя и номер телефона);

– описание вознаграждения;

– комментарий (опционально).

Выходные данные

Выходными данными мобильного приложения являются активности. Для дальнейшего понимания следует объяснить, что это такое.

«Активность — одна четко определенная операция, которую может выполнить пользователь. Например, в приложении могут присутствовать активности для составления сообщения электронной почты, поиска контакта или создания снимка. Активности обычно ассоциируются с одним экраном и программируются на Java» [12, с. 44].

Во время работы с приложением, пользователь может перемещаться и взаимодействовать с 18 активностями, предназначенными для:

1. Авторизации/регистрации (Auth);
2. Заполнения данных профиля (RegInfo);
3. Выбора действия (MainActivity);
4. Просмотра входящих сообщений (MessageCenter);
5. Просмотра данных своего профиля (Profile);
6. Изменения данных профиля (UpdateProfile);
7. Выбора и установки аватара (AvatarPicker);
8. Добавления запроса/предложения на доставку (Addition);
9. Просмотра результатов поиска по запросу (OfferList);
10. Просмотра подробностей найденного предложения (OfferDetails);
11. Просмотра профиля другого пользователя (AnotherProfile);
12. Просмотра списка исходящих предложений (MyOffersList);
13. Редактирования исходящего предложения (UpdateOffer);
14. Редактирования исходящего запроса (UpdateRequest);
15. Просмотра списка исходящих запросов (MyRequests);
16. Просмотра одного из исходящих предложений (MyOffer);
17. Просмотра одного из исходящих запросов (ViewMyRequest);
18. Просмотра входящего запроса (ViewIncomingRequest).

Внешний вид активностей будет представлен на рисунках далее с описанием особенностей их отображения и взаимодействия с ними в разделе 4.1.

4 РЕАЛИЗАЦИЯ

4.1 Реализация интерфейсов приложения «Pick And Go»

В «Pick And Go» большое внимание уделяется удобству использования. Для этого используются различные визуальные компоненты и специальные возможности.

Визуальные компоненты сопровождают пользователя на всех этапах использования приложения, начиная с главного экрана. Например, при регистрации пользователю поступает стартовое сообщение, что немедленно отображается на главной странице приложения, появлением яркой точки на кнопке перехода к входящим сообщениям. Внешний вид главного экрана при наличии новых сообщений изображен на рисунке 4.1, а. После прочтения сообщений отметка исчезает (рисунок 4.1, б).

Важным визуальным компонентом также являются статусы объектов. Например, на экране со списком входящих сообщений, новые – отмечаются специальным значком, который исчезает после их прочтения (старта активности «MessageCenter»). Пример отображения сообщения при первом посещении активности со списком сообщений после поступления сообщения и последующем отображен на рисунке 4.2, а и б - соответственно.

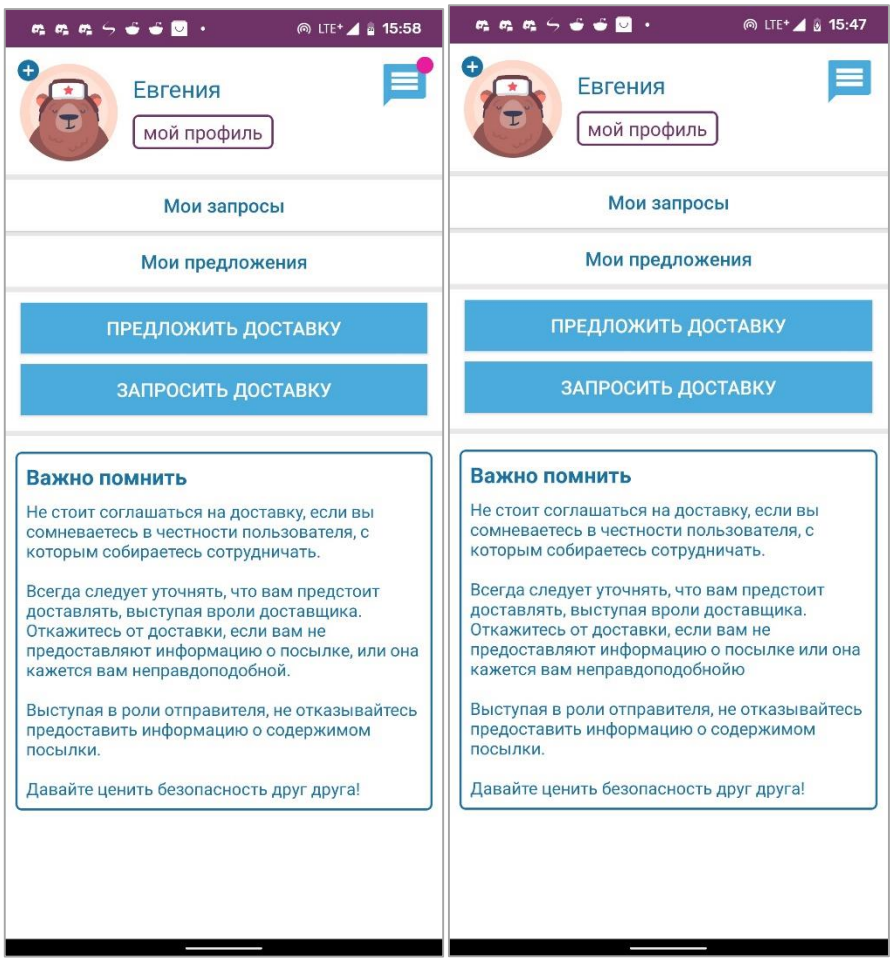
Нельзя не отметить, что выбранный способ реализации отображения статуса сообщений является весьма интересным, т. к. напрямую связан с жизненным циклом активности и особенностями работы базы данных. Реализация метода представлена в листинге 5.1.

В методе onStart() класса MessageCenter все сообщения из базы данных передаются в список, объявленный как List<Message> messages, для последующей передачи в адаптер. Так как метод onChildAdded, используемый для синхронизации данных с базой имеет свойство обновлять данные объектов сразу после их изменения и добавлять, как новые элементы списка, требуется менять статус в тот момент, когда

пользователь покидает активность. Для этого изменение статуса сообщений происходит в методе onStop() класса MessageCenter, листинг которого представлен ниже (листинг 1). Важно отметить, что метод onStop() сработает даже в случае уничтожения активности, а потому, смена статуса сообщений в нем оправдана. «Метод onStop() вызывается, когда активность перестает быть видимой для пользователя. Это может произойти из-за того, что она полностью закрывается другой активностью, отображаемой поверх нее, или из-за того, что активность готовится к уничтожению. Если метод onStop() вызывается из-за того, что активность готовится к уничтожению, перед onStop() вызывается метод onSaveInstanceState()» [12, с. 174].

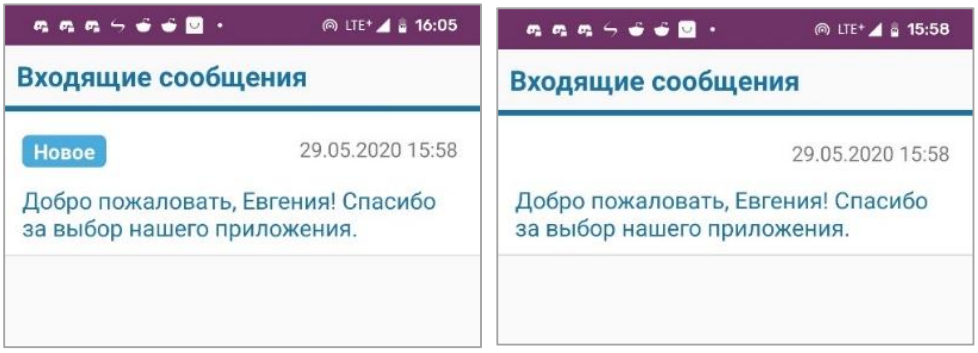
Листинг 5.1 – Изменение статуса сообщений

```
@Override
public void onStop(){
    super.onStop();
    messages.clear();
    mAuth = FirebaseAuth.getInstance();
    final FirebaseUser user = mAuth.getCurrentUser();
    if (user!=null) {

myRef.child("Users").child(user.getUid()).child("Messages").addChildEventListener(new
ChildEventListener() {
            @Override
            public void onChildAdded(@NonNull DataSnapshot dataSnapshot, @Nullable
String s) {

myRef.child("Users").child(user.getUid()).child("Messages").child(dataSnapshot.getKey
()).child("status").setValue(false);
            }

            @Override
            public void onChildChanged(@NonNull DataSnapshot dataSnapshot, @Nullable
String s)  {
            }

            @Override
            public void onChildRemoved(@NonNull DataSnapshot dataSnapshot) {}

            @Override
            public void onChildMoved(@NonNull DataSnapshot dataSnapshot, @Nullable
String s) {}

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

            }
        });
    }
}
```



а) б)

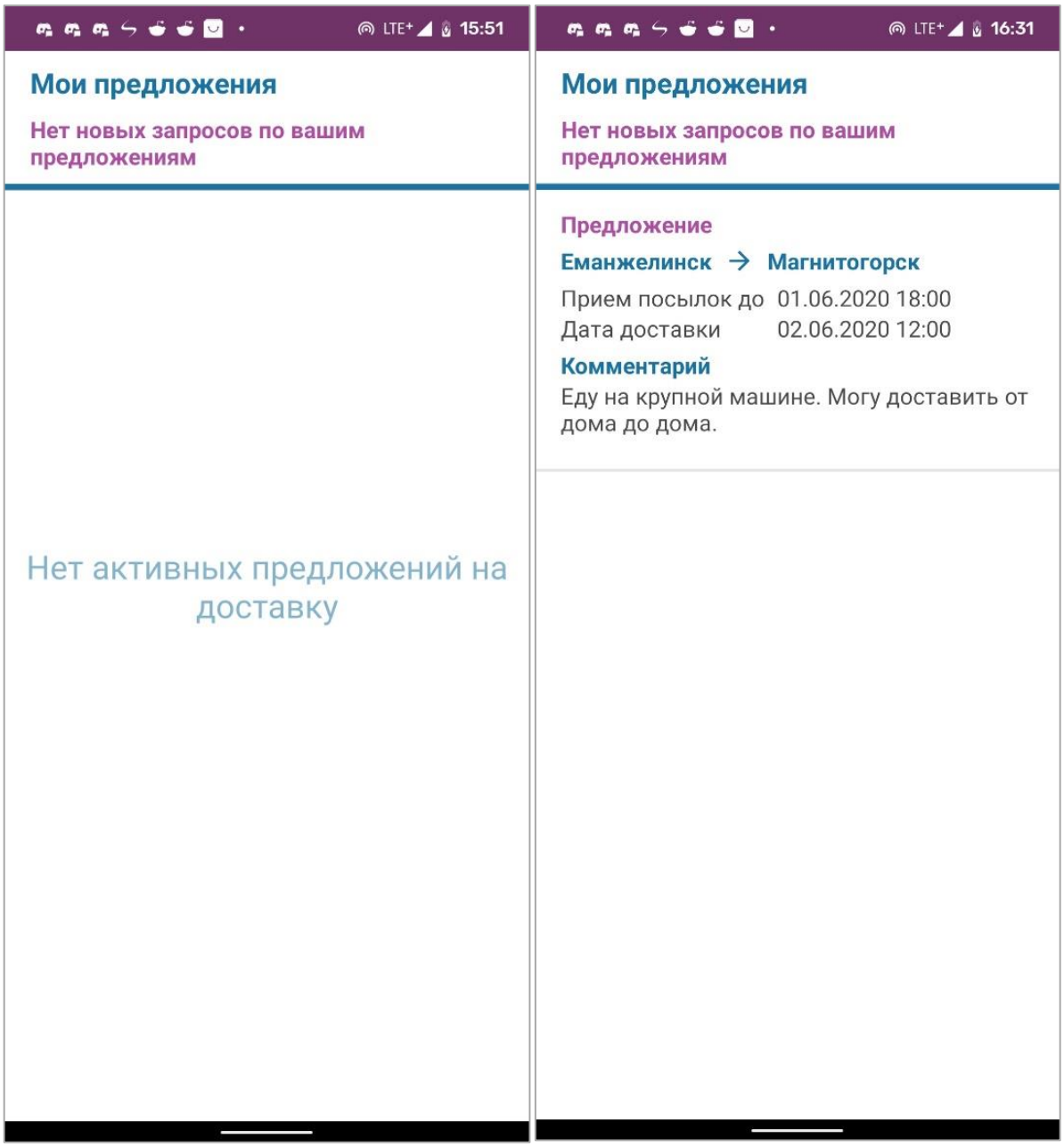
Рисунок 4.1 - Внешний вид главной страницы (MainActivity) при наличии (а) и отсутствии (б) новых сообщений



а) б)

Рисунок 4.2 - Внешний вид нового (а) и прочитанного сообщений (б) в активности MessageCenter

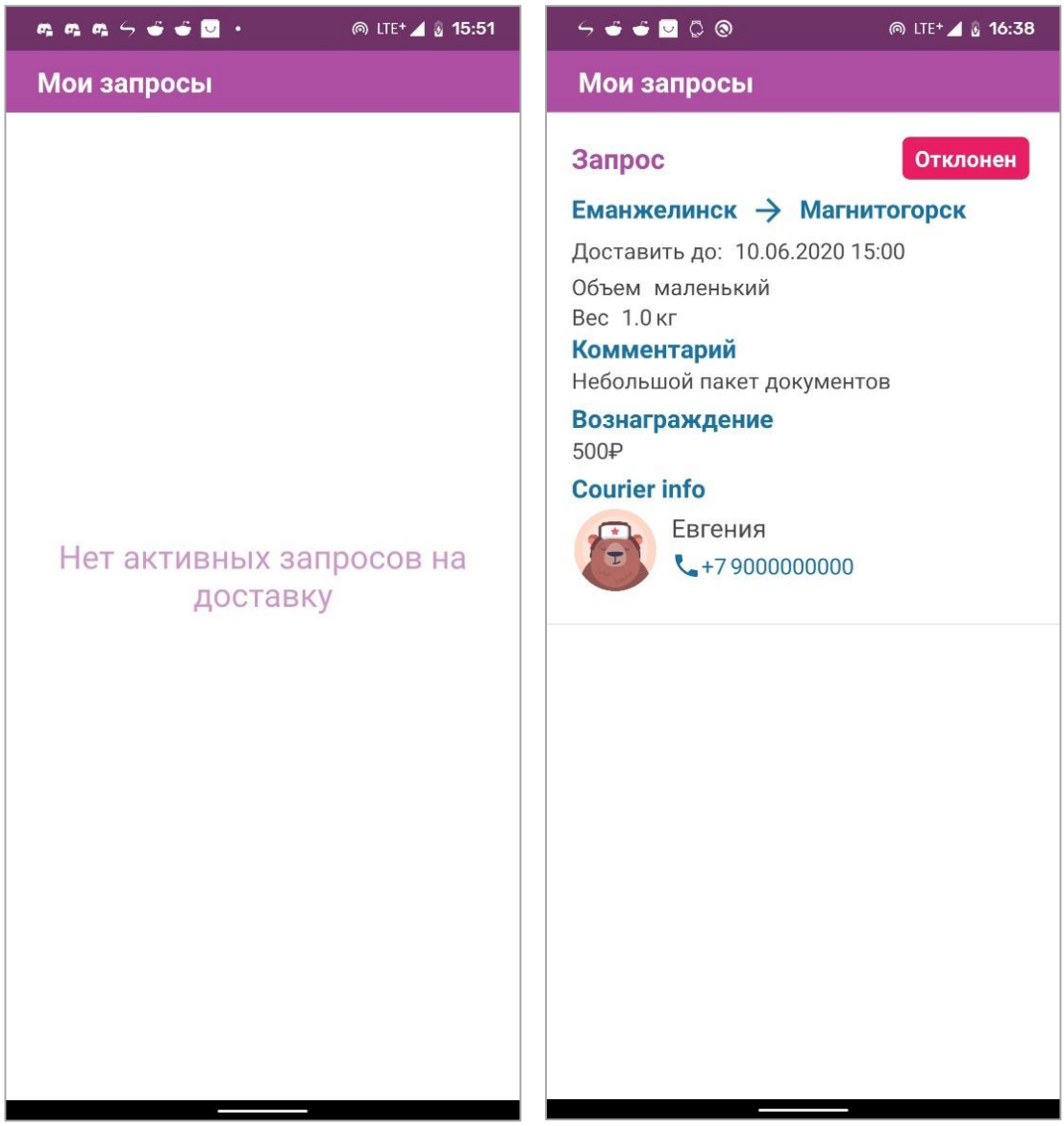
Внешний вид активности MyOffersList представлен на рисунке 4.3. При отсутствии исходящих предложений на доставку в нем отображается соответствующее сообщение (рисунок 4.3, а), при наличии – список предложений (рисунок 4.3, б).



а) б)

Рисунок 4.3 – Внешний вид активити MyOffersList при отсутствии исходящих предложений (а) и при наличии одного предложения (б)

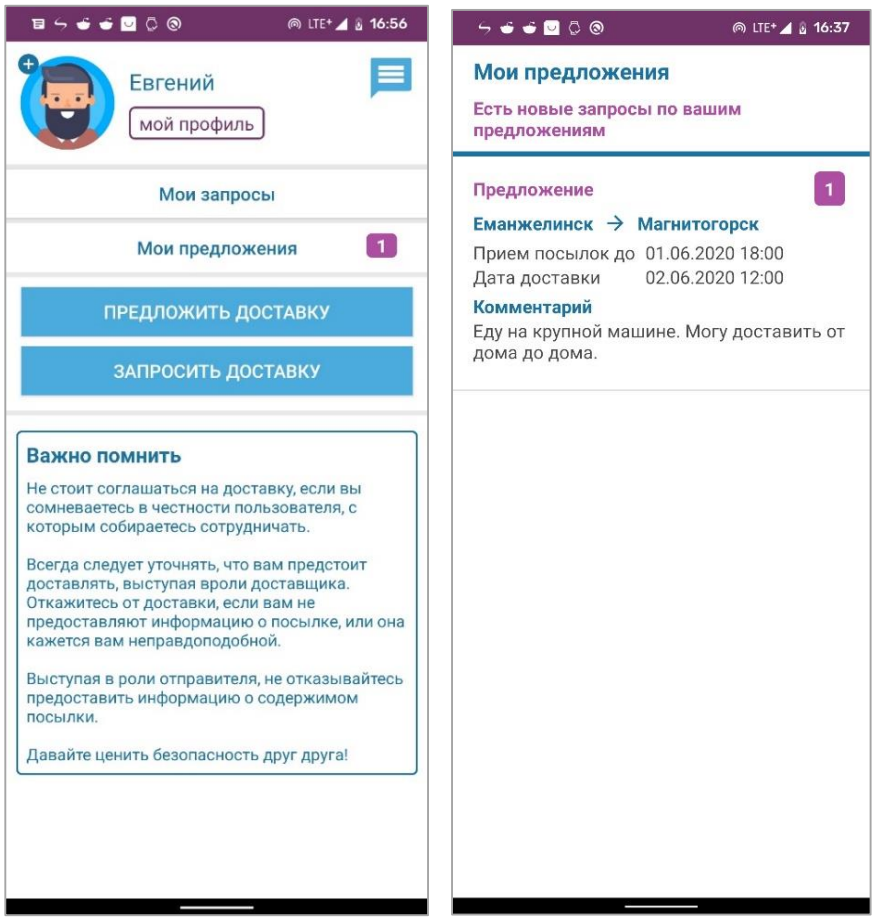
Внешний вид активности MyRequests представлен на рисунке 4.4. При отсутствии исходящих запросов на доставку в нем отображается соответствующее сообщение (рисунок 4.4, а), при наличии – список запросов (рисунок 4.4, б).



а) б)

Рисунок 4.4 – Внешний вид активити MyRequests при отсутствии исходящих предложений (а) и при наличии одного предложения (б)

Аналогично статусам сообщений и элементу, указывающему на наличие новых сообщений, отображаются элементы, уведомляющие о наличии новых входящих запросов. Они появляются в активностях MyOffer и MyOfferList, как продемонстрировано на рисунке 4.5. При этом в шапке активности MyOfferList также появляется сообщение, уведомляющее о наличии новых предложений по запросу. Важно отметить, что знаком наличия новых запросов, содержащим также их количество, отмечается именно то предложение, по которому поступили запросы. Это делает просмотр входящих запросов быстрым и удобным.



а) б)

Рисунок 4.5 – Внешний вид компонентов, уведомляющих о наличии новых входящих запросов в активностях MainActivity и MessageCenter

Активность Addition предусмотрена для добавления запросов и предложений. В зависимости от того, какой тип запроса указан во входящем Intent, оно принимает вид формы создания запроса или предложения.

«Интенты представляют собой многоцелевые средства передачи информации, а класс Intent предоставляет разные конструкторы в зависимости от того, для чего должен использоваться интент» [13, с. 121].

На рисунке 4.6 изображен внешний вид активности Addition, отображаемый при намерении создать предложение. На рисунке 4.7 – внешний вид при намерении создать запрос.

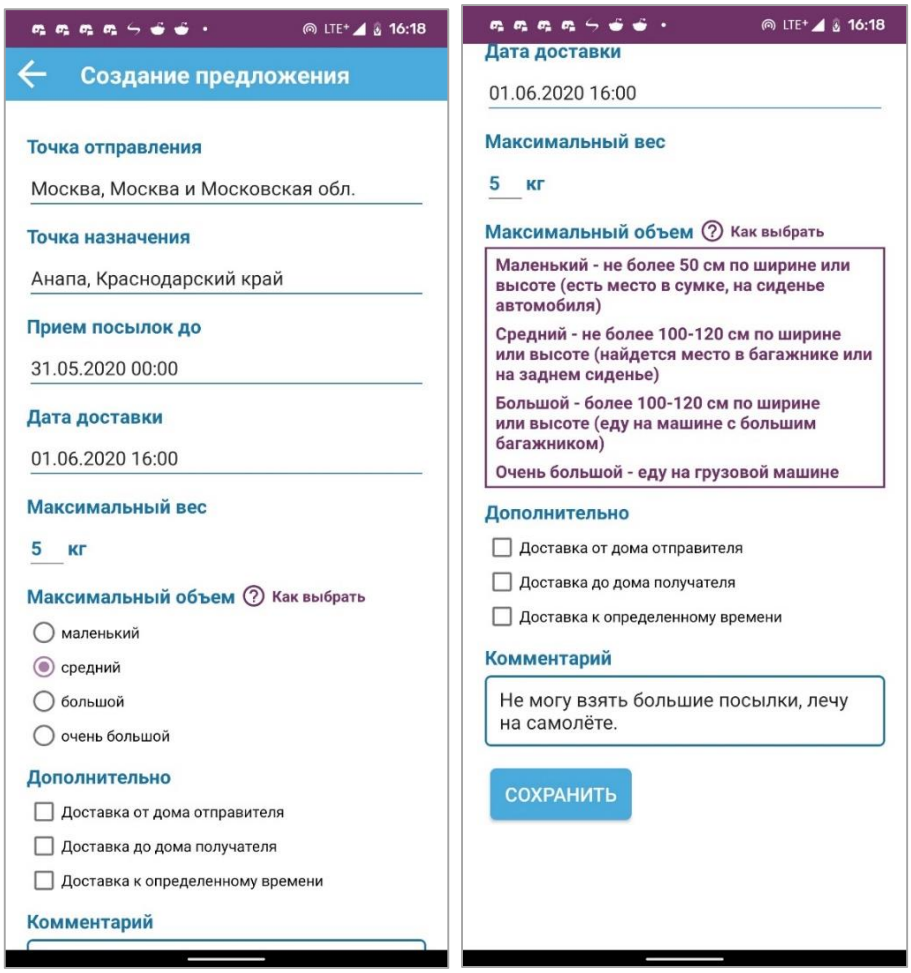


Рисунок 4.6 – Внешний вид активности Addition при получении из Intent указания типа формы «предложение» (подсказка демонстрируется)

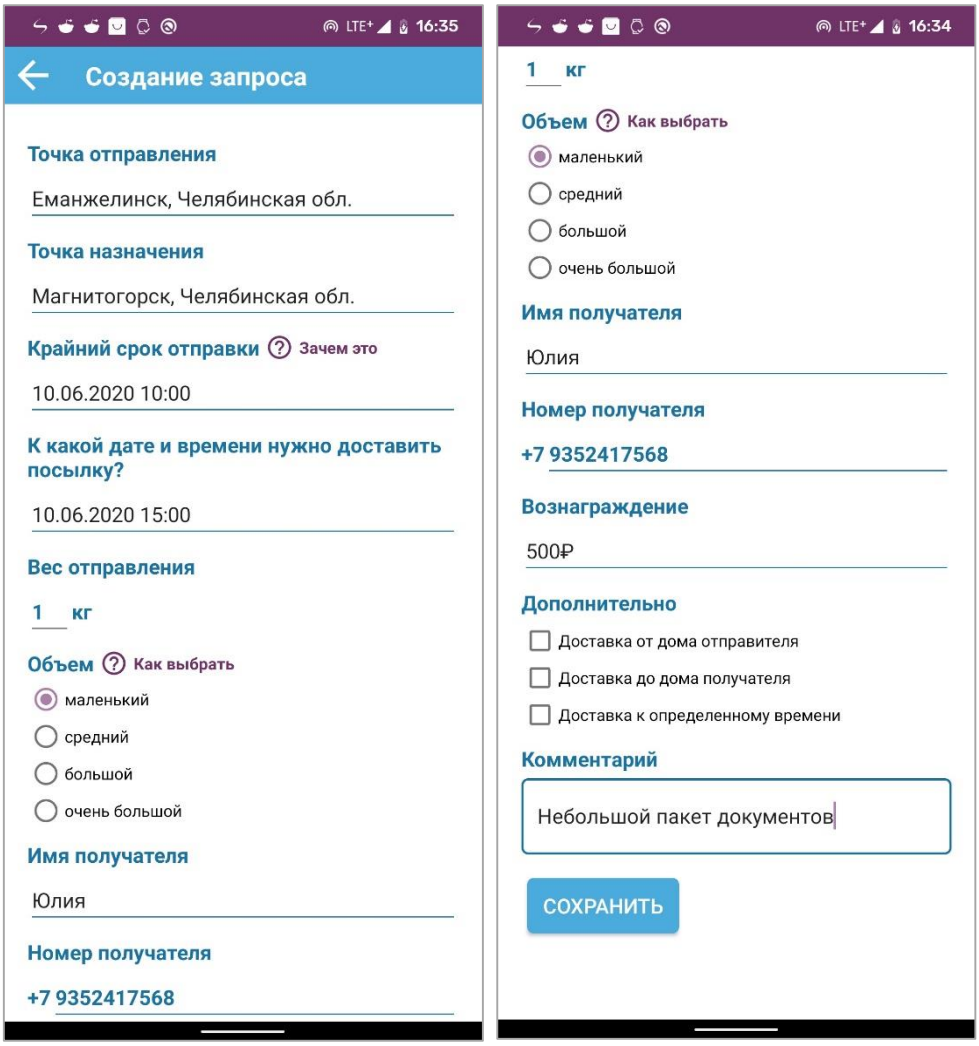


Рисунок 4.7 - Внешний вид активности Addition при получении из Intent указания типа формы «запрос» (подсказки скрыты)

Как на рисунке 4.6, так и на рисунке 4.7 можно заметить, что для однозначного понимания назначения полей и ввода верной информации пользователем предусмотрены подсказки, появляющиеся и исчезающие по нажатию на текст соответствующего вопроса. Чтобы пользователь ввел полезные для дальнейшего использования данные, он должен понимать, что он делает. В случае с выбором объема отправления, подсказка служит для однозначного понимания всеми пользователями мер объема. Такая реализация интерфейса позволяет избавить

пользователя от ввода числовых данных и быстро заполнить форму без потери корректности и полезности вводимой информации.

Внешний вид открытых подсказок активности Addition открытой в режиме добавления запроса изображен на рисунке 4.8.

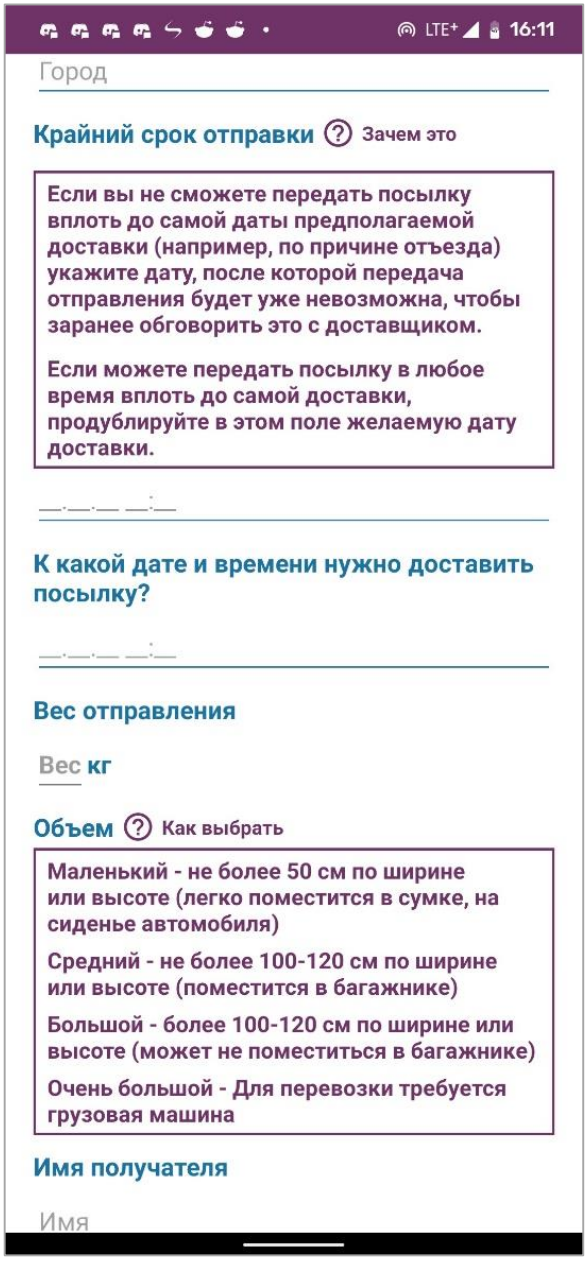
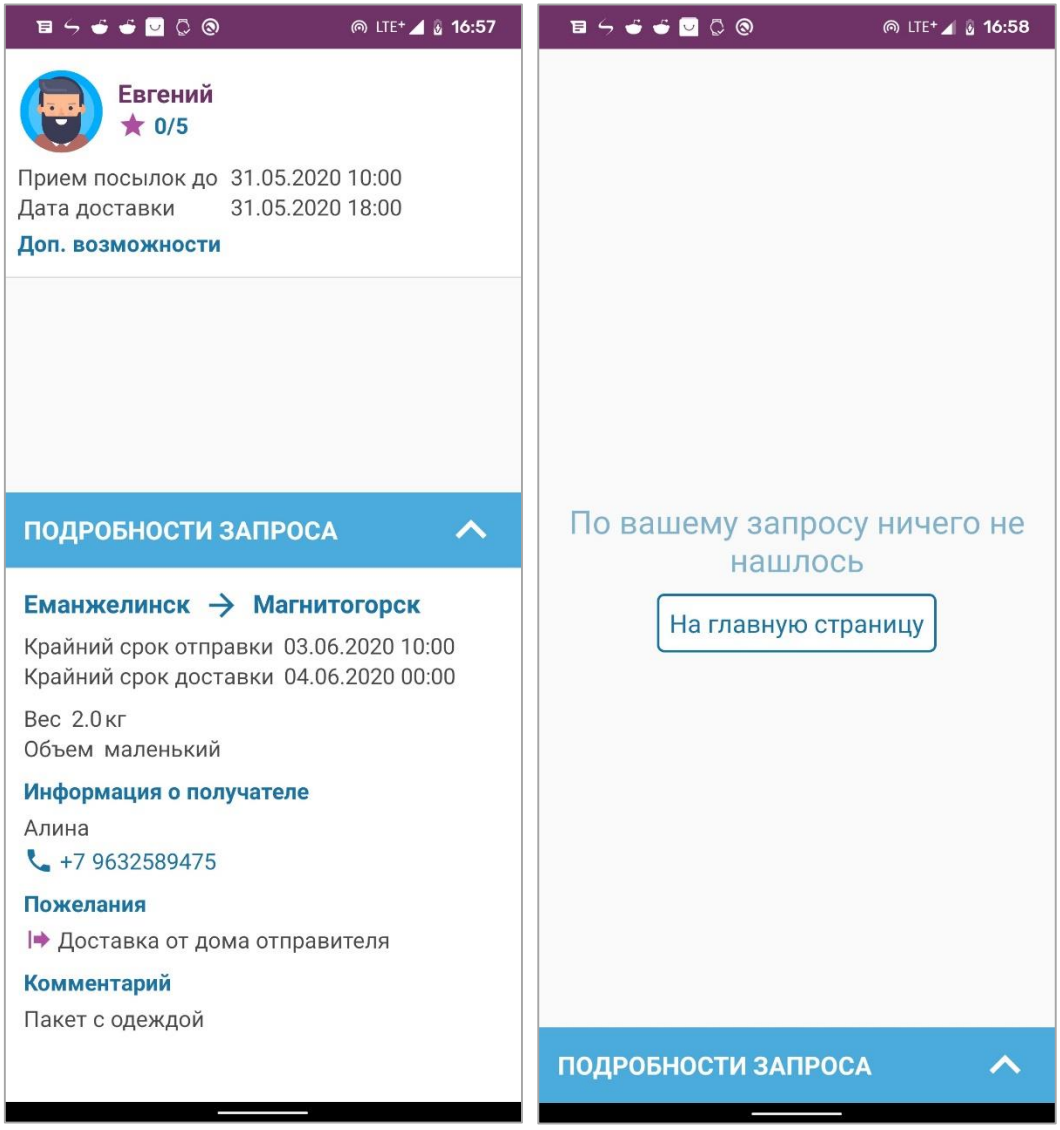


Рисунок 4.8 – Фрагмент экрана активности Addition при получении из Intent указания типа формы «запрос» (подсказки демонстрируются)

После добавления запроса пользователь попадает в активность OffersList, которая содержит список предложений, если они были найдены (рисунок 4.9, а) и содержит сообщение об отсутствии результатов поиска в противном случае (рисунок 4.9, б).

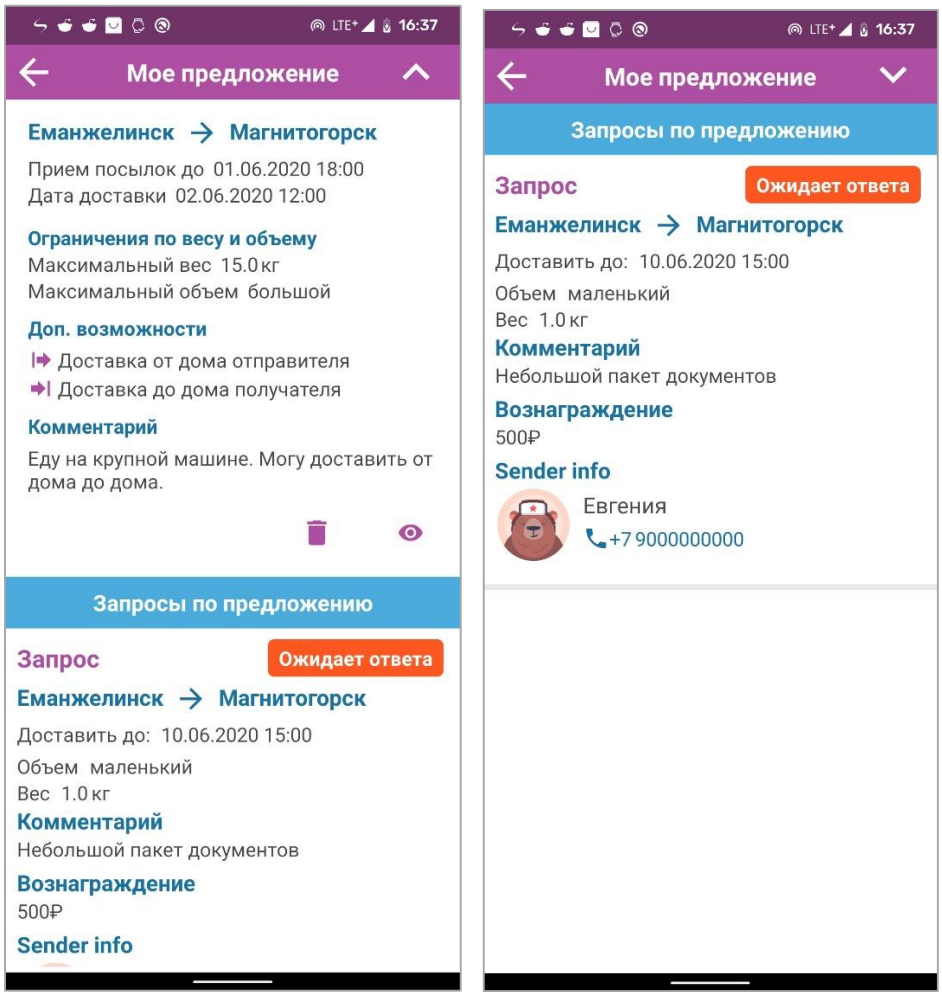


а) б)

Рисунок 4.9 – Внешний вид активности OffersList при наличии (а) и отсутствии (б) результатов поиска

Для удобства пользователя реализованы и другие возможности организации элементов. Например, при просмотре своего предложения, есть возможность свернуть подробности предложения, чтобы удобнее просматривать список поступивших по нему запросов, как показано на рисунке 4.10.

Режим видимости запроса и возможность редактирования также визуально отображаются в активности MyOffer. На рисунке 4.10, а видно, что статус видимости предложения «видимое», а редактирование недоступно по причине присутствия запросов по данному предложению.



а) б)

Рисунок 4.10 – Результат нажатия на кнопку, скрывающую подробности предложения в активности MyOffer

На рисунке 4.11 наоборот, изображено предложение со статусом видимости «невидимое» и доступное к редактированию по причине отсутствия входящих запросов.

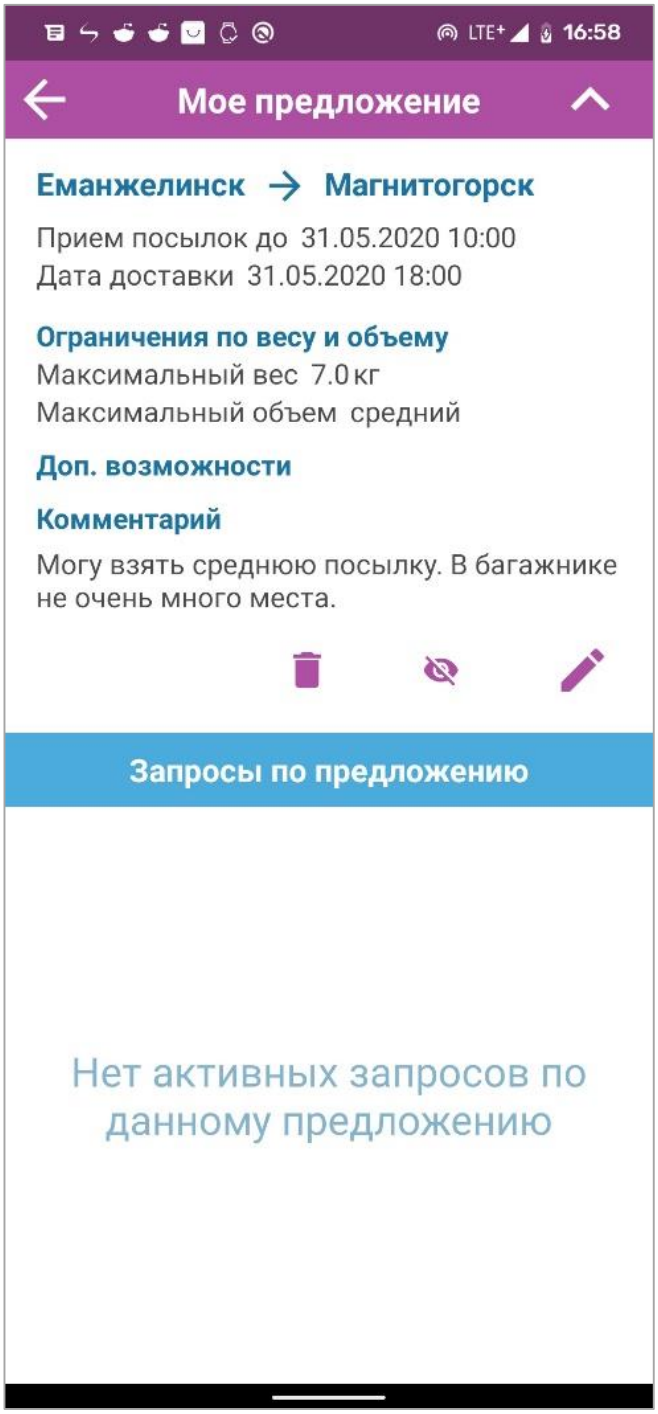
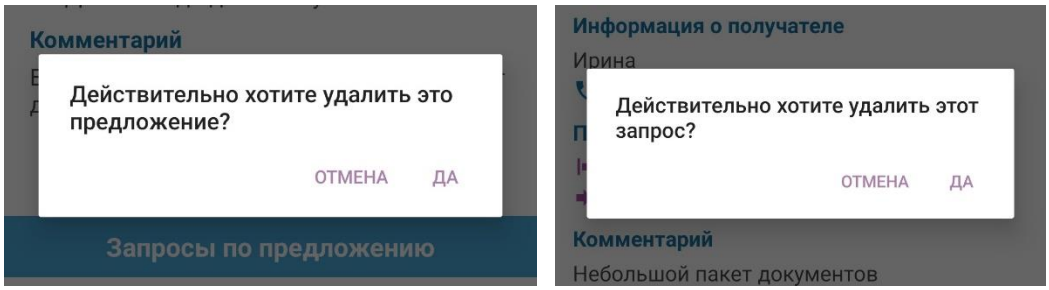


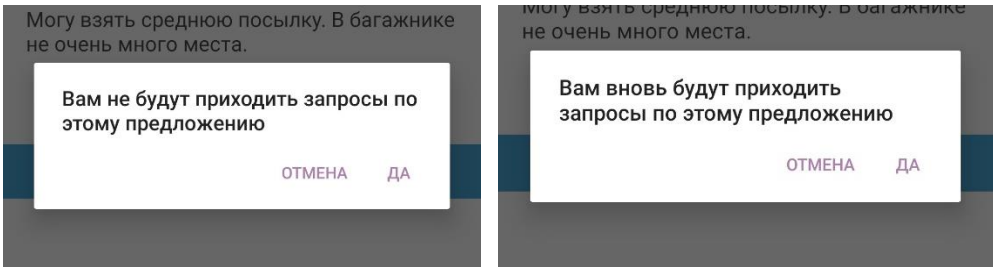
Рисунок 4.11 - Активность MyOffer, отображающая внешний вид невидимого для поиска запроса

Для подтверждения действий удаления и смены видимости предусмотрены специальные диалоговые окна. Непреднамеренное нажатие на кнопки смены видимости и удаления могут привести к неприятным для пользователя последствиям, поэтому важно запрашивать подтверждение для таких действий. Примеры уведомлений в различных случаях появления диалоговых окон представлены на рисунке 4.12, рисунке 4.13.



а) б)

Рисунок 4.12 – Диалоговые окна подтверждения удаления запроса (а) и предложения (б)



а) б)

Рисунок 4.13 – Диалоговые окна подтверждения смены видимости предложения на «невидимое» (а) и видимое (б)

Для редактирования запросов, предложений и информации профиля предусмотрены отдельные активности. Они выглядят аналогично активностям для первичного ввода данных о запросе и предложении (Addition) и информации профиля (RegInfo).

Инструменты ввода в приложении организованы таким образом, чтобы пользователю не приходилось думать, какой формат ввода является верным.

Для полей, требующих ввода названия населенного пункта, используется элемент AutoCompleteTextView, на который прикреплен адаптер, предлагающий варианты ввода, в соответствии с введенной последовательностью символов. Адаптер работает независимо от регистра и ищет любые вхождения введенной подстроки в строки с имеющимися вариантами. Названия загружаются из базы данных. В базе хранится информация о 2504 населенных пунктах. Элементы AutoCompleteTextView используются на трех активностях: Addition, UpdateOffer и UpdateRequest. Пример работы автозаполнения представлен на рисунке 4.14.

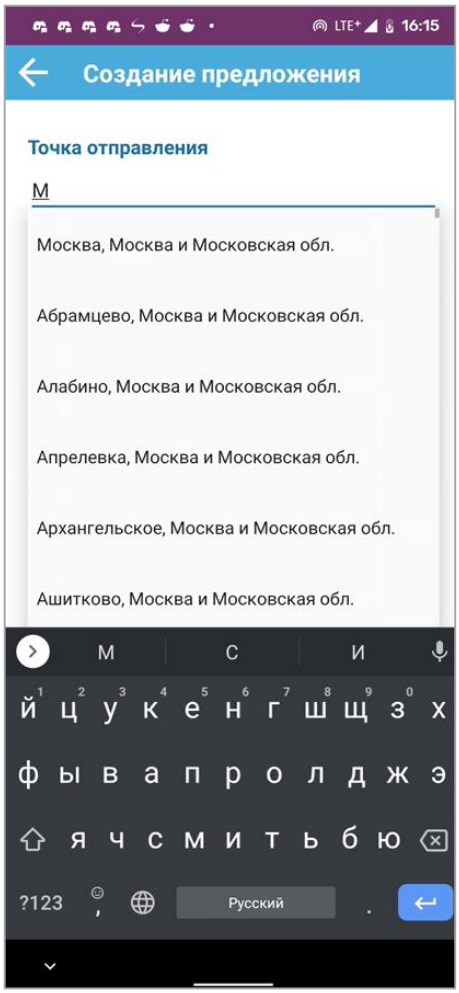


Рисунок 4.14 – Результат ввода первой буквы названия населенного пункта

Для заполнения полей, требующих ввода даты и времени, используются шаблонные Android-элементы DatePicker и TimePicker. Делающие ввод удобным и корректным. Такие элементы используются в трех активностях: Addition, UpdateOffer, UpdateRequest и RegInfo. Пример отображения DatePicker представлен на рисунке 4.15, пример отображения TimePicker - на рисунке 4.16.

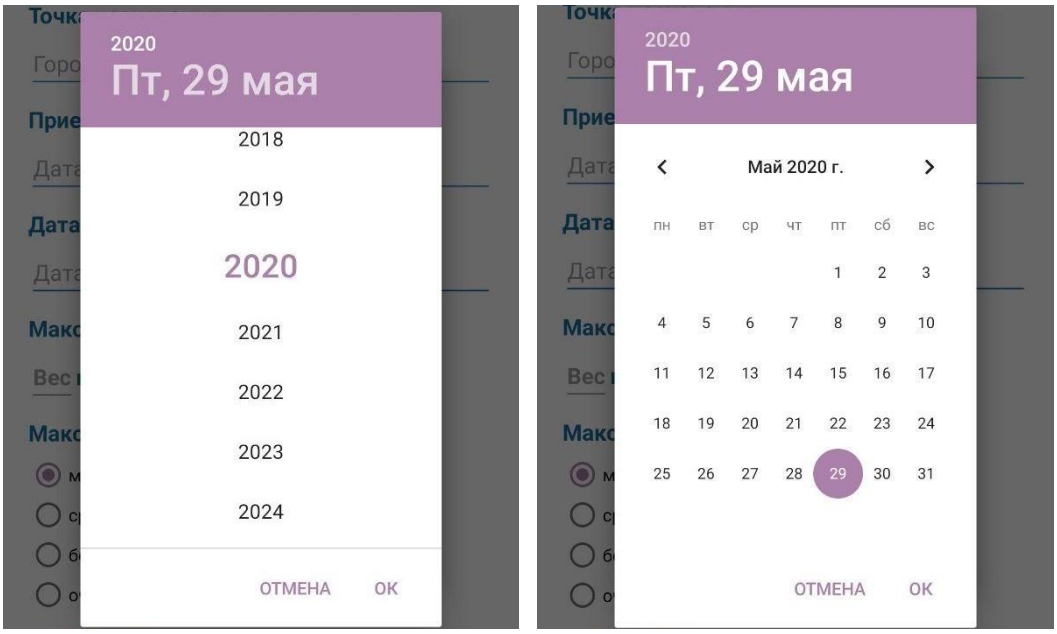


Рисунок 4.15 – Ввод даты при помощи элемента DatePicker

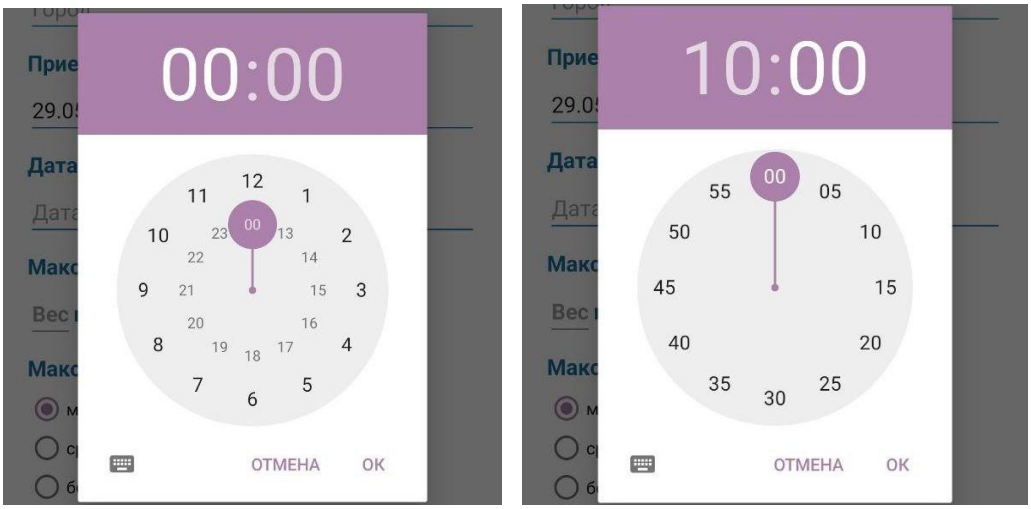


Рисунок 4.16 - Ввод времени при помощи элемента TimePicker

Одним из самых важных визуальных компонентов являются статусы запросов. Их отображение организовано таким образом, что пользователю не требуется тратить время на чтение текста статуса. Быстрое восприятие возможно благодаря цветовой ассоциации. Статус «отклонен» имеет красный цвет (рисунок 4.17), «создан» - голубой (рисунок 4.18), «ожидает ответа» - оранжевый (рисунок 4.19), «подтвержден» - зеленый (рисунок 4.20).

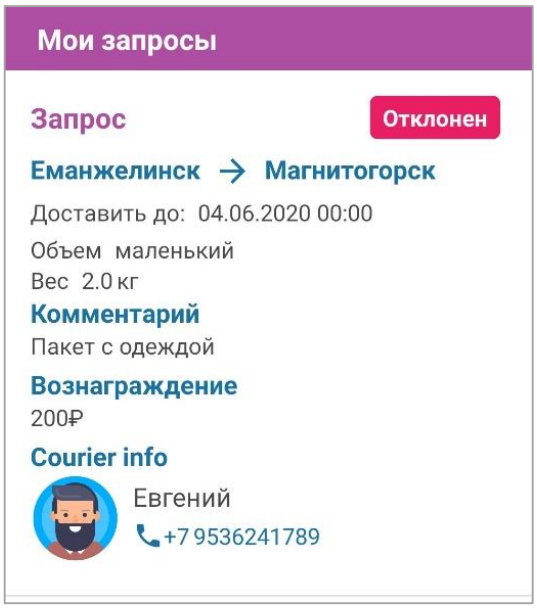


Рисунок 4.17 – Запрос со статусом «отклонен»

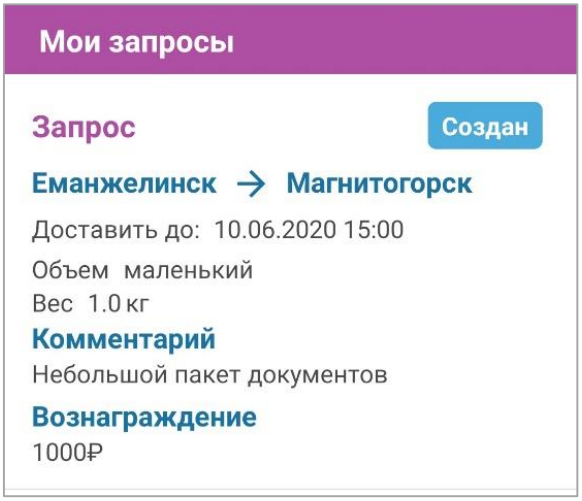


Рисунок 4.18 – Запрос со статусом «создан»

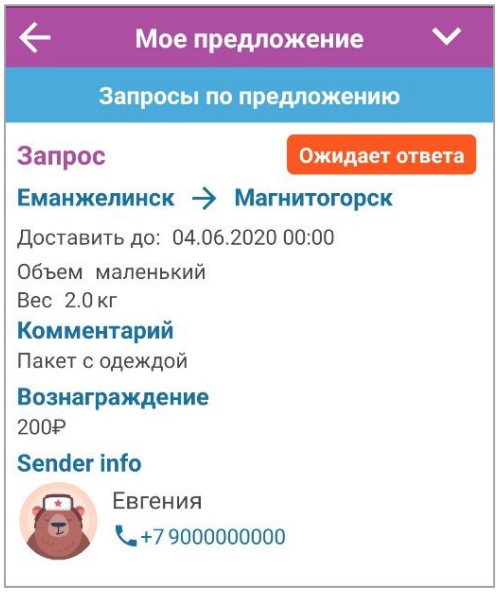


Рисунок 4.19 – Запрос со статусом «ожидает ответа»

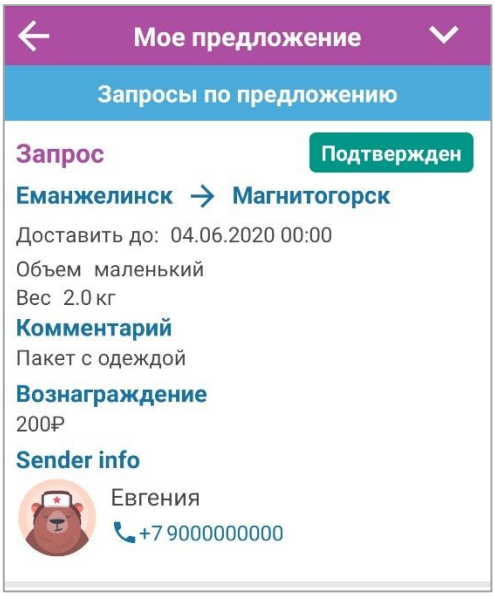
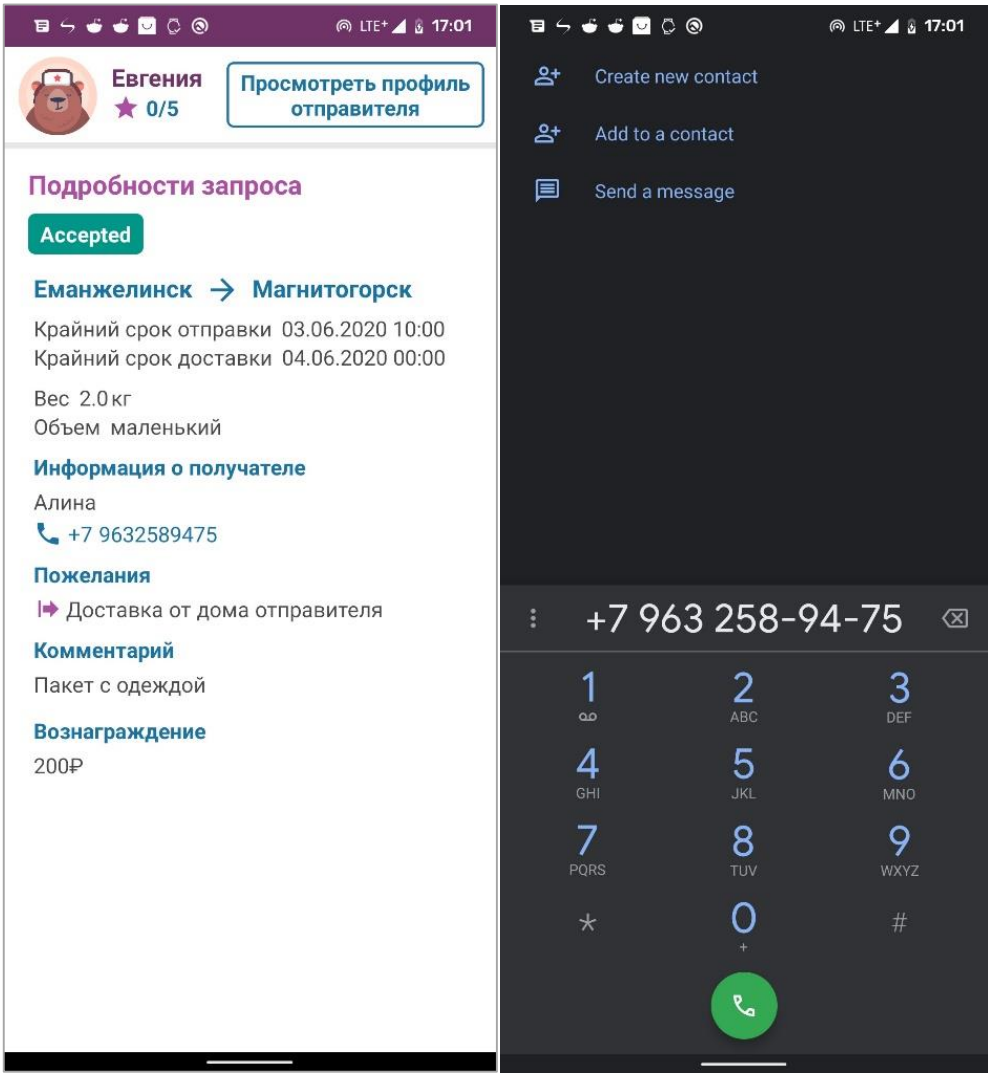


Рисунок 4.20 – Запрос со статусом «подтвержден»

Дополнительно предоставляется возможность автоматического набора номера для совершения звонка и написания сообщения. Такая возможность присутствует в двух активностях: ViewIncomingRequest и AnotherProfile. Кнопки вызова и написания сообщения, а также отзывчивость к нажатию строки с номером телефона получателя посылки позволяют быстро совершить звонок или написать сообщение, не прибегая к ручному вводу и устраняют надобность в записи телефонов

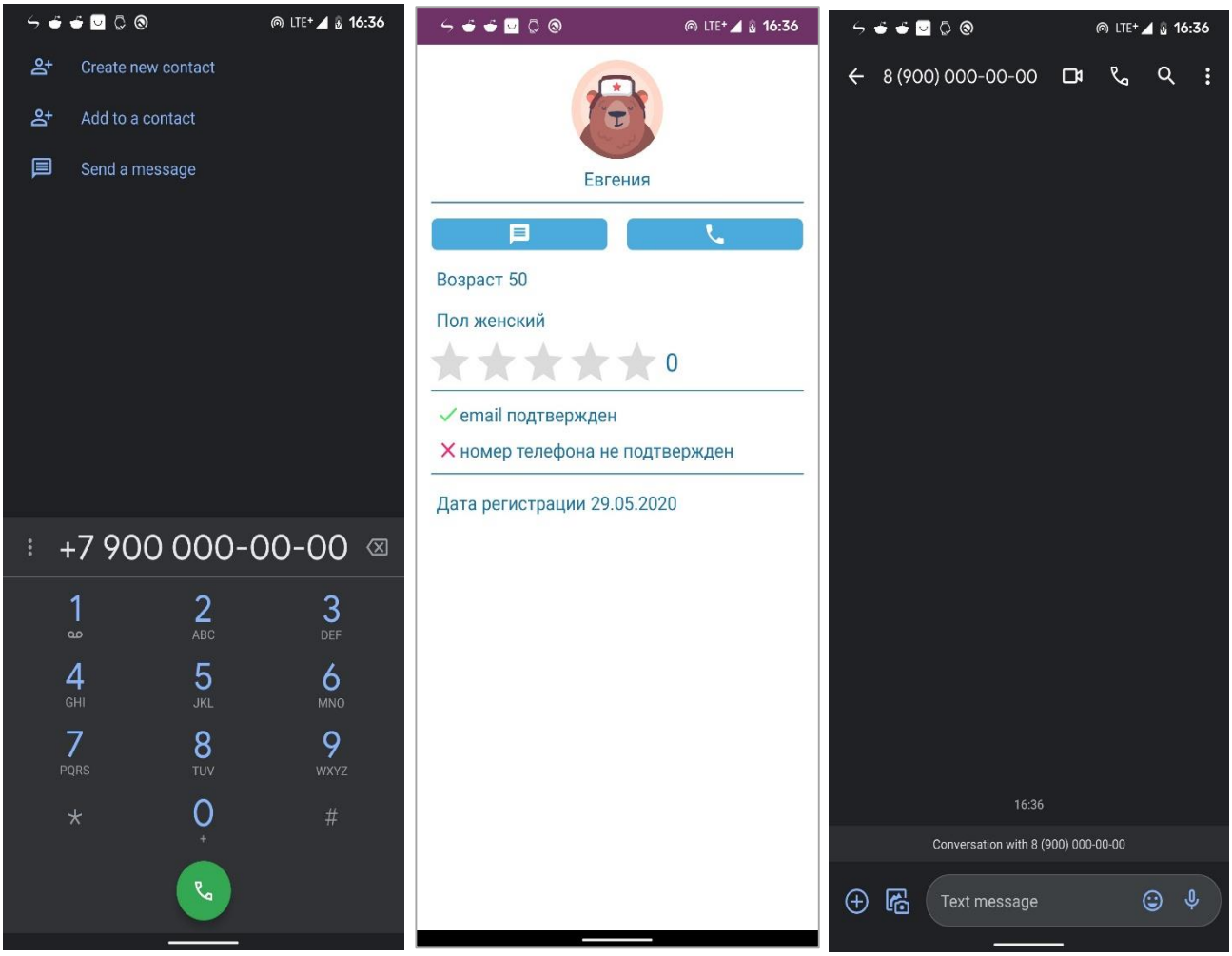
в контакты. Пример перехода от номера получателя к набору номера представлен на рисунке 4.21.



а) б)

Рисунок 4.21 – Результат нажатия на номер получателя (б) в активности ViewIncomingRequest (а)

При просмотре профиля другого пользователя скрыт номер мобильного телефона. Звонок и написание сообщения осуществляются по нажатию кнопок. Результат нажатия на описанные кнопки представлен на рисунке 4.22.



а) б) в)

Рисунок 4.22 – Результаты нажатия на кнопки звонка (а) и написания сообщения (в) в активности AnotherProfile (б)

Во избежание ввода некорректной информации или пропуска ввода необходимых данных в приложении предусмотрена проверка данных при вводе во всех активностях, предусматривающих такие действия. Это активности: Auth, RegInfo, Addition, UpdateRequest, UpdateOffer, UpdateProfile.

В активности Auth предусмотрена проверка на совпадение пароля в полях для первичного и вторичного его ввода при регистрации (рисунок 4.23).

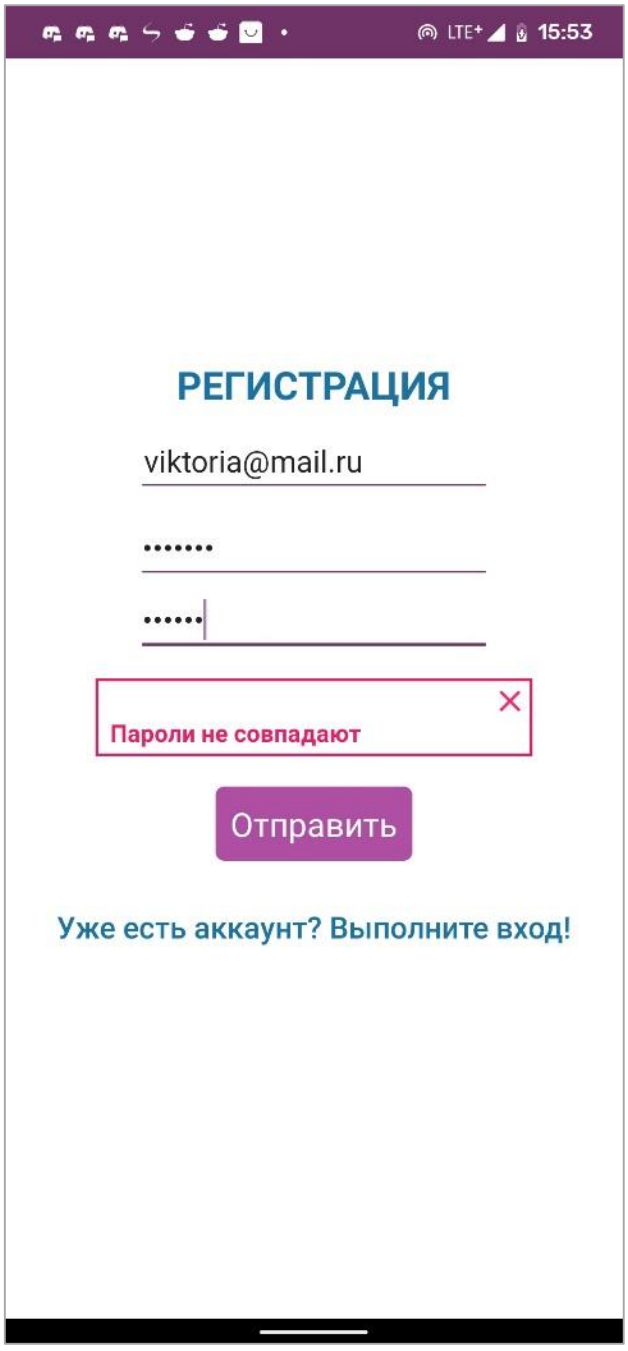


Рисунок 4.23 – Результат проверки ввода пароля в активности Auth

Для активности RegInfo предусмотрены следующие правила проверки: даты рождения (которая не должна быть в будущем времени или указывать на то, что потенциальному пользователю менее 12 лет); Имени, которое должно иметь длину не менее двух букв; Номера мобильного телефона длиной ровно 10 цифр и

начинающегося с цифры 9. Пример вывода сообщения об ошибках ввода в активности RegInfo представлен на рисунке 4.24.

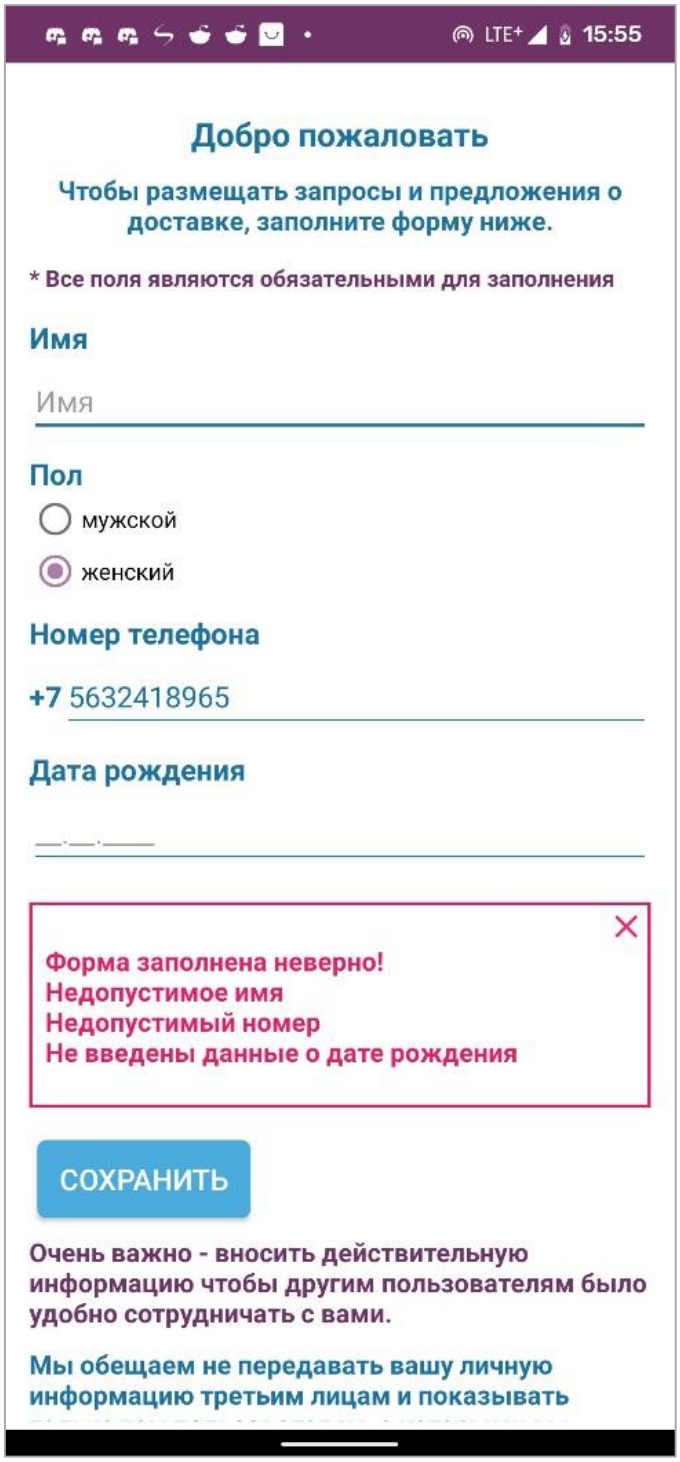
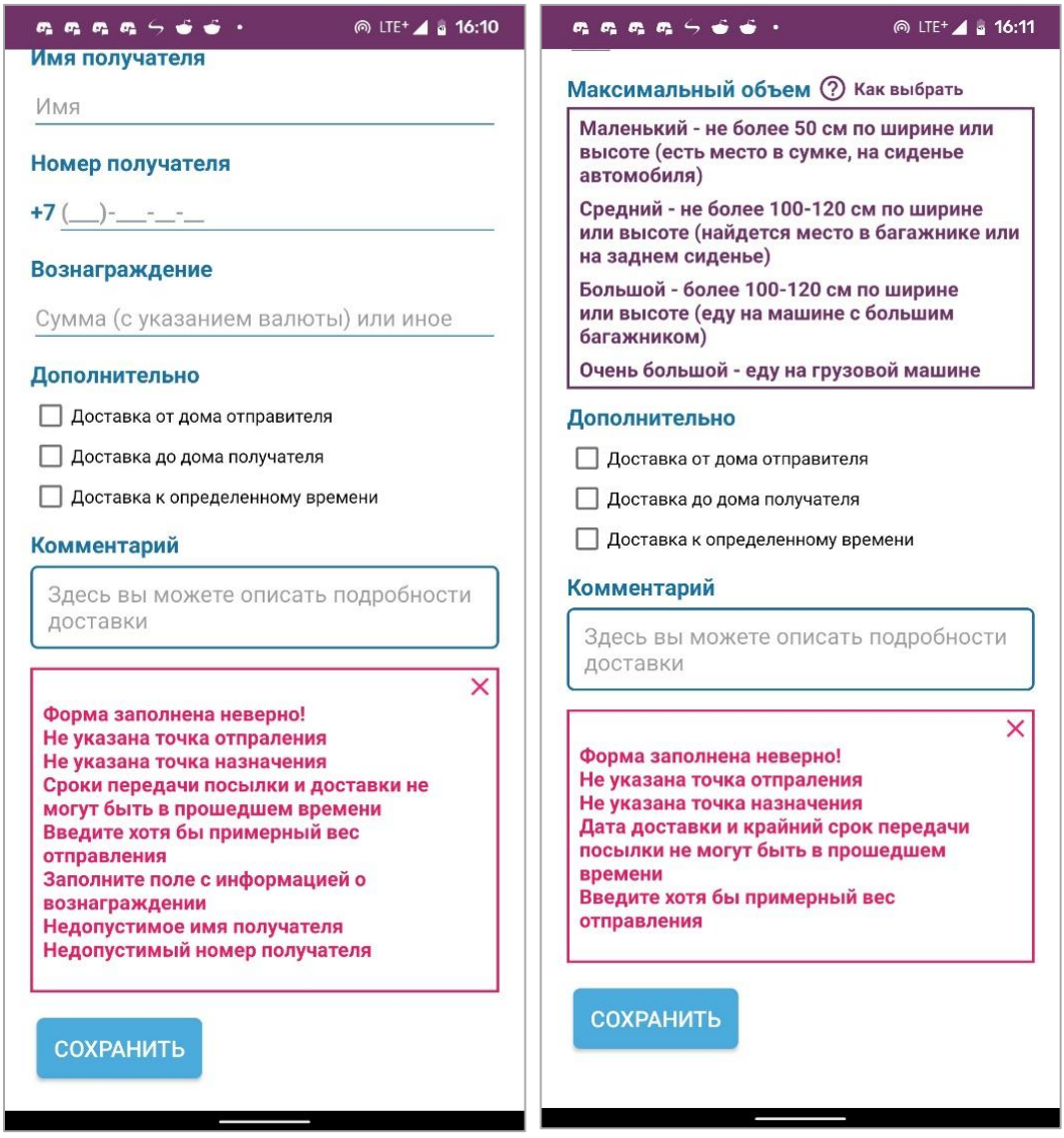


Рисунок 4.24 - Результат проверки ввода в активности RegInfo

На рисунке 4.25 Представлены результаты отправки пустой формы в активности Addition, открытой в режиме добавления запроса (рисунок 4.25, а) и предложения (рисунок 4.25, б).



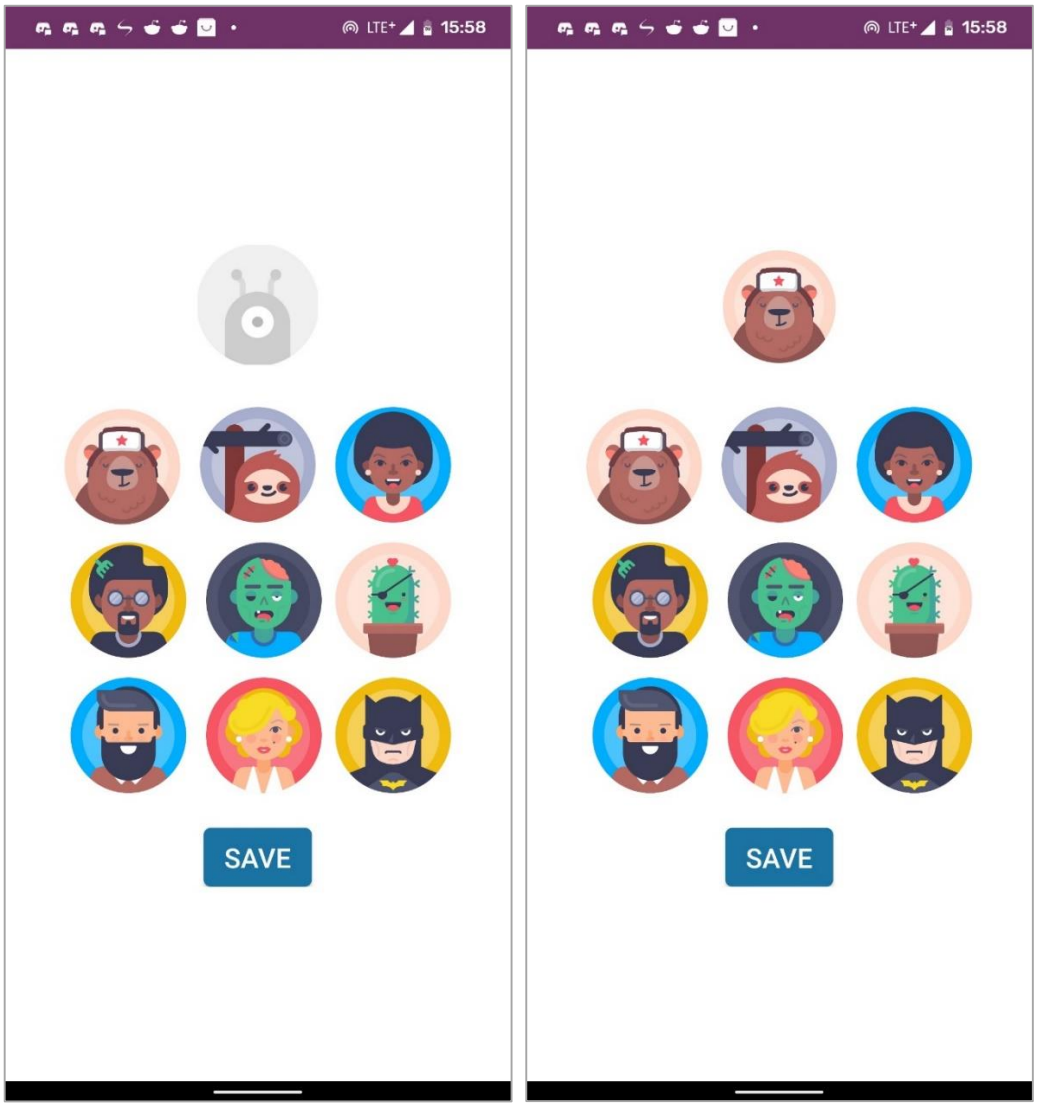
а) б)

Рисунок 4.25 – Результат попытки отправить пустую форму в активности Addition

Таким образом интерфейс и логика приложения исключают возможность ввода противоречивых и заведомо неверных данных.

Поскольку приложение в своей последней версии еще не обладает функцией загрузки фотографий, в качестве аватара пользователю предлагается несколько изображений на выбор.

Эта функция носит как развлекательный характер, так и возможность ассоциировать детали запросов и предложений с изображениями и быстрее осуществлять поиск требуемой информации об участниках запланированной доставки. Активность для выбора аватара и результат его выбора представлены на рисунке 4.26.



а) б)

Рисунок 4.26 – Выбор аватара в активности AvatarPicker

5 ТЕСТИРОВАНИЕ

5.1 Методология тестирования

Smoke-тестирование

Когда программное обеспечение устанавливается после внесения изменений, в его работе возможны ошибки, которые могут критично влиять на функционал всей системы. Поэтому важно осуществлять проверку основных функций системы сразу после сборки. За это отвечает данный вид тестирования (smoke-testing).

Главной задачей данной процедуры является выявление дефектов сразу после сборки программного обеспечения. Тест при этом содержит малое количество сценариев, однако их должно хватать для того, чтобы выявить явные ошибки функциональности. Также smoke test проводится как приемочное испытание перед полноценным функциональным тестированием.

Функциональное тестирование

Благодаря проведению такого вида тестирования выполняется проверка способности системы решать пользовательские задачи в конкретных заданных условиях. Проводимые функциональные тесты полностью имитируют использование системы, помогая в своевременном выявлении системных ошибок в работе программ. Для функциональных тестов необходима разработка программы и методики испытаний. Функциональное тестирование производится с использованием заранее подготовленных сценариев. По окончании тестирования описывается список обнаруженных ошибок и рекомендаций по улучшению производительности системы.

Пользовательское приемочное тестирование

User Acceptance Testing (UAT) — это приемочное тестирование, которое проводится конечными пользователями. При любой разработке или доработке

программного обеспечения есть стадия такого тестирования. Так как обычный пользователь не может полноценно проверить совершенные доработки, важно проводить UAT-тестирование по заранее подготовленным сценариям. Они повышают качество проверки и существенно облегчают задачу конечных пользователей.

Альфа-тестирование и бета-тестирование

«Альфа-тестирование (alpha-testing) – это вид приемочного тестирования, которое обычно проводится на поздней стадии разработки продукта и включает имитацию реального использования продукта штатными разработчиками либо командой тестировщиков. Обычно альфа-тестирование заключается в систематической проверке всех функций программы.

Бета-тестирование (beta-testing) – интенсивное использование почти готовой версии продукта с целью выявления максимального числа ошибок в его работе для их последующего устранения перед окончательным выходом продукта на рынок, к массовому потребителю. Бета-тестирование представляет собой реально работающую версию программы с полным функционалом. И задача бета-тестов – оценить возможности и стабильность работы программы с точки зрения ее будущих пользователей» [13].

5.2 Проведение процедуры тестирования

На протяжении всей разработки применялось smoke-тестирование. Разработанное приложение тестировалось на двух виртуальных и одном физическом устройстве. В качестве виртуальных устройств были выбраны LG Nexus 5X и Google Pixel 3 с версиями ОС Android 9.0 и разрешениями экранов 1080x1920 и 1080x2160 соответственно. В качестве физического устройства для тестирования использовался смартфон Xiaomi Mi9T с версией ОС Android 10.0 и разрешением экрана 2340x1080. Приложение прошло этапы альфа-тестирования и UAT-тестирования.

На этапе UAT-тестирования были выявлены недостатки внешнего вида активностей. На основании заключений потенциальных пользователей были внесены изменения. Были добавлены заголовки и сообщения об отсутствии элементов списков и результатов поиска. Как показали результаты последующего тестирования, приложение корректно отображает все активности после внесения изменений.

В процессе функционального тестирования были выявлены и устранены некоторые ошибки логики. В частности, была устранена ошибка вывода некорректных результатов поиска после внесения изменений в предложение о доставке. Появилась возможность автозаполнения поля ввода названия населенного пункта. Сценарии тестирования и их результаты для последней версии приложения описаны в таблице 1.

Таблица 1 - Список сценариев тестирования приложения

Тест	Описание теста	Ожидаемый результат	Результат
Регистрация	Ввод адреса электронной почты и пароля, повтор пароля. Попытка отправить форму с несовпадающими паролями, с совпадающими паролями. Попытка еще раз зарегистрироваться с тем же адресом электронной почты.	При вводе несовпадающих первично и повторно введенного паролей должно появиться сообщение об ошибке. При корректном вводе регистрация должна пройти успешно и внешний вид активности должен смениться на ввод регистрационных данных. При попытке повторной регистрации должно появиться сообщение об ошибке.	Совпал с ожидаемым

Продолжение таблицы 1

Тест	Описание теста	Ожидаемый результат	Результат
Авторизация	Попытка авторизации с существующими и не существующими данными. Авторизация пользователя, вошедшего в систему впервые и пользователя, вошедшего повторно.	Вход в профиль в случае первичной авторизации с корректными данными авторизации и последующее отображение активности для ввода данных профиля. Вход в профиль при вторичной авторизации с корректными авторизационными данными и отображение главной активности. Появление сообщения об ошибке при вводе некорректных данных.	Совпал с ожидаемым
Ввод данных профиля	Попытка отправить пустую или некорректно заполненную форуму. Отправка формы, заполненной верно.	При вводе некорректных данных или их отсутствии и попытке отправить должно появиться сообщение об отсутствии возможности отправки и описания ошибок заполнения формы. При вводе корректных данных должна открыться главная страница приложения, где отображается имя пользователя, введенное при заполнении формы регистрации.	Совпал с ожидаемым

Продолжение таблицы 1

Тест	Описание теста	Ожидаемый результат	Результат
Изменение данных профиля	Попытка удалить данные и отправить пустую или изменить данные на некорректные и отправить форуму. Также тестируется отправка формы, заполненной верно.	При входе на страницу изменения данных профиля форма должна быть уже заполнена текущими данными. При попытке изменить их на некорректные или удалить с последующей отправкой формы должно появиться сообщение об ошибке. При изменении данных на корректные, но отличные от исходных и отправке формы пользователь должен быть перенаправлен в свой профиль с обновленной информацией.	Совпал с ожидаемым
Чтение сообщения	Первичный и повторный просмотр сообщения	Наличие специального значка непрочитанных сообщений на главной странице до прочтения. Пометка «новое» на непрочитанном сообщении при первичном входе в активность со списком входящих сообщений. Отсутствие специального значка непрочитанных сообщений на главной странице при возвращении на главную страницу. Отсутствие пометки «новое» на сообщении при повторном входе в активность со списком входящих сообщений.	Совпал с ожидаемым

Продолжение таблицы 1

Тест	Описание теста	Ожидаемый результат	Результат
Добавление предложения доставки	Попытка ввести корректные и некорректные данные в форму	Предупреждение о наличии ошибок заполнения формы при вводе некорректных данных. Уведомление об успешном добавлении при вводе корректных данных и переход на главную активность. Появление предложения в списке исходящих предложений пользователя и возможность перейти к просмотру его подробностей.	Совпал с ожидаемым
Добавление запроса на доставку	Попытка ввести корректные и некорректные данные в форму	Предупреждение о наличии ошибок заполнения формы при вводе некорректных данных. Уведомление об успешном добавлении при вводе корректных данных и переход на активность со списком предложений, соответствующих запросу (если таковые имеются, если нет, должна отобразиться страница списка со ссылкой для перехода на главную активность). Появление запроса в списке исходящих запросов пользователя и возможность перейти к просмотру его подробностей.	Совпал с ожидаемым
Поиск предложения в различных режимах видимости	Попытка найти по запросу подходящее предложение в режиме видимости и невидимости	Появление предложения в списке найденных в видимом режиме и отсутствие в невидимом.	Совпал с ожидаемым

Продолжение таблицы 1

Тест	Описание теста	Ожидаемый результат	Результат
Просмотр подробностей найденного предложения	Попытка перейти к деталям найденного предложения, к профилю его автора. Попытка перейти к набору номера и сообщения из профиля автора предложения.	Переход к деталям предложения по нажатию на выбранное предложение. Возможность просмотреть профиль автора предложения и прейти к набору номера или сообщения с автоматически введенным номером телефона.	Совпал с ожидаемым
Отправка запроса	Попытка отправить запрос по найденному предложению	В аккаунте автора запроса: отправка запроса, уведомление о добавлении запроса, изменение статуса запроса. В аккаунте автора предложения: появление информационных обозначений на главной странице и странице со списком предложений автора; появление входящего запроса на странице описания деталей предложения, по которому пришел запрос, возможность принять и отклонить запрос, возможность перейти к деталям входящего запроса, просмотреть профиль его автора и прейти к набору номера или сообщения с автоматически введенным номером телефона.	Совпал с ожидаемым

Продолжение таблицы 1

Тест	Описание теста	Ожидаемый результат	Результат
Ответ на запрос	Положительный и отрицательный ответ на запрос	В аккаунте автора предложения: изменение статуса входящего запроса (активность с деталями предложения) на «принят» в случае положительного ответа, исчезновение запроса – в случае отрицательного ответа. В обоих случаях должны исчезнуть специальные компоненты, отображающие наличие новых входящих запросов, если входящий запрос был единственным. В аккаунте автора предложения: Изменение статуса запроса в списке исходящих запросов и на странице просмотра деталей запроса. Появление сообщения об ответе на запрос.	Совпал с ожидаемым
Изменение запроса	Попытка удалить данные и отправить пустую или изменить данные на некорректные и отправить форуму. Также тестируется отправка формы, заполненной верно.	При входе на страницу изменения данных, форма должна быть уже заполнена текущими данными. При попытке изменить их на некорректные или удалить с последующей отправкой формы должно появиться сообщение об ошибке. При изменении данных на корректные, но отличные от исходных и отправке формы пользователь должен быть перенаправлен на страницу описания запроса (который был изменен) с обновленной информацией.	Совпал с ожидаемым

Продолжение таблицы 1

Тест	Описание теста	Ожидаемый результат	Результат
Изменение предложения	Попытка удалить данные и отправить пустую или изменить данные на некорректные и отправить форуму. Также тестируется отправка формы, заполненной верно.	Если для предложения есть входящие запросы, его изменение должно быть невозможно. При входе на страницу изменения данных, форма должна быть уже заполнена текущими данными. При попытке изменить их на некорректные или удалить с последующей отправкой формы должно появиться сообщение об ошибке. При изменении данных на корректные, но отличные от исходных и отправке формы пользователь должен быть перенаправлен на страницу с описанием предложения (которое было изменено) с обновленной информацией.	Совпал с ожидаемым
Удаление запроса	Попытка удалить запрос с различными статусами.	При удалении запроса, имеющего статус «создан» или «отклонен», запрос должен исчезнуть из списка исходящих запросов. При удалении запроса, имеющего статус «ожидает ответа» или «подтвержден», запрос должен исчезнуть из исходящих запросов у отправителя запроса и из входящих у автора предложения. Автору предложения должно прийти сообщение об отмене запроса отправителем.	Совпал с ожидаемым

Окончание таблицы 1

Тест	Описание теста	Ожидаемый результат	Результат
Удаление предложения	Удаление предложения с входящими запросами и без них.	При удалении предложения всем авторам запросов, закрепленных за данным предложением должно прийти уведомление об отмене доставки. Предложение должно исчезнуть из списка исходящих предложений.	Совпал с ожидаемым
Смена аватара	Попытка добавить аватар при его отсутствии и изменить.	При входе на страницу изменения аватара должны отображаться варианты изображений, доступные к установке в качества аватара и текущий аватар (над предложенными). По нажатию на предложенные изображения текущий аватар должен меняться. При выходе из активности смены аватара изменения должны сохраниться.	Совпал с ожидаемым

Согласно приведенным результатам тестирования, последняя версия разработанного приложения является стабильной и реализует основной функционал приложения.

ЗАКЛЮЧЕНИЕ

В результате работы было спроектировано и реализовано мобильное приложение «Pick And Go» на платформе Android.

К достоинствам реализованного приложения можно отнести удобный и интуитивно понятный интерфейс. Это значительно ускоряет работу с приложением, что очень важно, так как решение будет применяться в предметной области доставки посылок и использование приложения подразумевает пребывание в пути. Большое преимущество приложению также дает и выбранная БД, использующая синхронизацию данных вместо стандартных HTTP-запросов. В связи с этим, необходимая информация обновляется в режиме реального времени и пользователю не требуется обновлять активности для просмотра актуальных данных. Также следует отметить, что использование приложения может принести пользователю экономическую выгоду.

Приложение имеет перспективы для развития и расширения функционала. Планируется реализовать возможность общения пользователей в чате приложения, сделать возможным выбор точек отправления и назначения с карты, добавить возможность указывать адреса при доставке от дома к дому. Также планируется размещение приложения в Google Play Store.

На основе вышеизложенного можно заключить, что созданное приложение является удобным инструментом для организации частной доставки.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Travel Post – товары из Европы и США с попутчиками. https://play.google.com/store/apps/details?id=io.travelpost.mobile&hl=ru. Дата обращения: 05.05.2020

2 BlaBlaCar: совместные поездки на авто или автобусе. https://play.google.com/store/apps/details?id=com.comuto&hl=ru. Дата обращения: 05.05.2020

3 Google Play Store. https://play.google.com/store. Дата обращения: 05.05.2020

4 Getberry – «блаблакар» для посылок и отправлений. https://shag-vpered.org/getberry-blablakar-dlya-posylok-i-otpravlenij. Дата обращения: 05.05.2020

5 BlaBlaCar. Часто задаваемые вопросы. https://www.blablacar.ru/faq/question/mogu-li-otpravit-posilku-ili-jivotnoe-s-pomoshju-saita. Дата обращения: 05.05.2020

6 Android Studio. https://developer.android.com/studio. Дата обращения: 05.05.2020

7 Gradle. https://gradle.org. Дата обращения: 28.03.2020

8 GitHub. https://github.com. Дата обращения: 28.03.2020

9 Eclipse. https://www.eclipse.org/eclipseide. Дата обращения: 05.05.2020

10 Firebase. https://firebase.google.com/products. Дата обращения: 28.03.2020

11 MongoDB. https://www.mongodb.com. Дата обращения: 05.05.2020

12 Гриффитс, Д. Head First. Программирование для Android / Дон Гриффитс, Дэвид Гриффитс; пер. с англ. Е. Матвеев. — СПб.: Питер, 2016. – 704 с.: ил. – (Head first O’Reilly). – Библиография в тексте.

13 Филлипс, Б. Android. Программирование для профессионалов/Б. Филлипс, К. Стюарт, К. Марсикано, 3-е изд.— СПб.: Питер, 2017. — 688 с.: ил. — (Серия «Для профессионалов»).

14 Альфа- и Бета-тестирование. - https://training.qatestlab.com/blog/technical-articles/alpha-beta-testing. Дата обращения: 05.05.2020

15 Тестирование программного обеспечения. - http://testimsoft.ru/uat-testirovanie. Дата обращения: 05.05.2020

16 МакГрат, М. Программирование на Java для начинающих /М. Мак-Грат; [пер. с англ. М.А. Райтмана]. – Москва: «Э», 2016. – 192 с. – (Программирование для начинающих).

17 Блох, Д. Java: эффективное программирование/Д. Блох, 3-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 464 с.: ил. — Парал. тит. англ.

18 Нимейер, П. Программирование на Java /П. Нимейер, Д. Леук; [пер. с англ. М.А. Райтмана]. – Москва: Эксмо, 2014. – 1216 с. – (Мировой компьютерный бестселлер).

19 Meet Android Studio. https://developer.android.com/studio/intro. Дата обращения: 27.03.2020

20 Учебник по Android. Уроки для начинающих. https://startandroid.ru/ru/uroki/vse-uroki-spiskom.html. Дата обращения: 28.03.2020

21 Firebase guides. https://firebase.google.com/docs/android/setup. Дата обращения: 28.03.2020

22 Firebase Realtime Database. Easily add sign-in to your Android app with FirebaseUI. https://firebase.google.com/docs/auth/android/firebaseui. Дата обращения: 28.03.2020

23 Firebase Realtime Database. Authenticate with Firebase using Password-Based Accounts on Android. https://firebase.google.com/docs/auth/android/password-auth. Дата обращения: 28.03.2020

24 Firebase Realtime Database. Read and Write Data on Android. https://firebase.google.com/docs/database/android/read-and-write. Дата обращения: 30.03.2020

25 Firebase Realtime Database. Work with Lists of Data on Android. https://firebase.google.com/docs/database/android/lists-of-data. Дата обращения: 05.04.2020