Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Космических и Информационных Технологий
институт
Информационные системы
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой ИС
_____ П.П. Дьячук
подпись инициалы, фамилия
« ____» _____ 2020 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 – «Информационные системы и технологии»

Разработка Android приложения «Kras-carpooling»

Руководитель _____ доцент, к.т.н. И. А. Легалов
подпись, дата должность, учёная степень инициалы, фамилия

Выпускник _____ А.Э. Лутков
подпись, дата инициалы, фамилия

Красноярск 2020

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Космических и Информационных Технологий
институт
Информационные системы
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой ИС
_____ П.П. Дьячук
подпись инициалы, фамилия
«____» _____ 2020 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Луткову Артему Эдуардовичу

Группа: КИ16-14Б Направление: 09.03.02 «Информационные системы и технологии»

Тема выпускной квалификационной работы: «Разработка Android приложения «Kras-carpooling»»

Утверждена приказом по университету № 6499/с от 22.05.2020 г.

Руководитель ВКР: И.А. Легалов, к.т.н., доцент кафедры «Информационные системы» ИКИТ СФУ.

Исходные данные для ВКР: Требования к разрабатываемой системе, методические указания научного руководителя, учебные пособия.

Перечень разделов для ВКР: Введение, теоретическая часть, структура проекта, программная реализация, заключение, список использованных источников.

Перечень графического материала: Презентация, выполненная в Microsoft Office PowerPoint 2016.

Руководитель ВКР _____ И. А. Легалов
подпись, дата инициалы, фамилия

Задание принял к исполнению _____ А. Э. Лутков
подпись, дата инициалы и фамилия студента

«____» _____ 2020 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Разработка Android приложения «Kras-carpooling»» содержит 46 страниц текстового документа, 29 иллюстраций, 11 использованных источников.

Актуальность.

Карпулинг – это совместное использования автомобиля с целью снижения расходов на топливо, парковку, уменьшения выброса вредных веществ и что порой немаловажно – использование карпулинга позволяет избежать стресса от езды за рулём. А если же карпулинг получит широкое распространение, то еще одним его следствием может стать снижение транспортного потока. Создание приложения для услуг карпулинга в городе Красноярске позволит людям в любое время и в любом месте найти поездку или же попутчика.

В качестве объекта исследования выступает создание приложения по поиску водителей и попутчиков.

Предметом исследования является процесс создания приложения и исследования рынка развития карпулинга в Красноярске.

Целью выпускной квалификационной работы является изучение всех необходимых материалов для разработки приложения по поиску попутчиков, а также получение полного представления о рынке карпулинга в городе Красноярске.

Основные задачи:

- Изучить существующие решения и спрос на данные услуги
- Выбрать наполнение системы и определиться с логикой приложения
- Изучить необходимые технологии для разработки приложения
- Реализовать приложение

Практическая значимость проекта заключается в том, что с помощью этого приложения люди смогут найти попутчиков и водителей, а также всю необходимую информацию о них, в городе Красноярске.

СОДЕРЖАНИЕ

1 Теоретическая часть.....
1.1 Объект работы 5
1.2 Райдшеринг.....
1.2.1 Понятие каршеринга.....
1.2.2 Карпулинг 7
1.3 Анализ существующих решений..... 7
1.4 Анализ рынка..... 9
1.4.1 Гипотезы 9
1.4.2 Интервью..... 9
1.5 Технологии и средства для разработки приложений 11
1.5.1 ОС Android..... 11
1.5.2 Язык программирования Java 12
1.5.3 Среда разработки Android Studio..... 13
1.5.4 Сборка проектов на языке Java..... 15
2 Структура проекта.....1
2.1 Проектирование модулей и сервисов 17
2.2 База данных NoSQL 19
3 Программная реализация.....2
3.1 Множественные окна одного приложения..... 23
3.2 Модуль входа..... 24
3.3 Модуль навигации 27
3.4 Модуль пользователя..... 30
3.5 Файловый модуль..... 33
3.6 Модуль поездки..... 36
ЗАКЛЮЧЕНИЕ 4
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 4

ВВЕДЕНИЕ

Проблема заполненных улиц, магистралей и парковок известна уже очень давно и продолжает распространяться по всему миру. А проблема загрязнения атмосферы выхлопными газами не теряет, а только наращивает свой потенциал. Так вот аналитиками ряда стран в качестве выхода видится такое явление как карпулинг или райдшеринг, ведь очень много людей, которые передвигаются по одному и тому же маршруту каждый день и для них, найти такого же попутчика будет поводом отказаться от автомобиля.

Карпулинг или райдшеринг – это совместное использование частного автомобиля с помощью онлайн-сервисов поиска попутчиков. Есть разные сайты, группы, приложения по поиску водителей и пассажиров, которым «по пути», но если задаться этим вопросом и попытаться найти в городе Красноярске удобный сервис, где можно будет предлагать или искать поездку, то найти ничего подходящего не удастся.

Целью выпускной квалификационной работы является изучение всех необходимых материалов для разработки приложения по поиску попутчиков, а также получение полного представления о рынке карпулинга в городе Красноярске.

Для достижения поставленной цели, были выделены следующие задачи:

- Изучить существующие решения и спрос на услуги карпулинга
- Выбрать наполнение системы и определиться с логикой приложения
- Изучить необходимые технологии для разработки приложения
- Реализовать приложение

А для вовлечения наибольшего количества человек, в приложении будут использоваться передовые решения, которые позволят максимально удобно пользоваться сервисом.

1 Теоретическая часть

1.1 Объект работы

Объектом работы является создание приложения на платформе Android, в котором будут связываться водители и попутчики в современной информационной среде.

Будет разработано программное приложение поиска водителя с личным транспортом для перевозки пассажиров при том, что водителю, так же как и пассажиру, нужно добраться до указанной точки назначения. Организовано это будет в городе Красноярске. Это решение поможет не только избежать лишних финансовых вложений для сторон, так как водитель не оплачивает расходы на поездку, а попутчик платит только лишь минимальные расходы за передвижение, но и повлияет на обстановку в городе Красноярска в целом (проблема заполненных улиц - заторов, проблема парковок, проблема выбросов вредных веществ).

Приложение будет разработано на языке Java, при использовании среды разработки Android Studio. В качестве базы данных была выбрана облачная нереляционная база данных (NoSQL) от Firebase.

1.2 Райдшеринг

Если с английского переводить слово Райдшеринг дословно, то оно обозначает «совместные автомобильные поездки». Идея состоит в том, чтобы одни люди – экономили на расходах на автомобиль, другие - с большим комфортом, чем на поезде или автобусе, добирались до нужного места. Так же, помимо этих достоинств данного направления, стоит отметить уменьшения транспортного потока, то есть снизиться заторы на дорогах, на парковках, а также уменьшится выброс вредных веществ от автомобилей.

Райдшеринг относительно старое слово, которое зародилось еще во время второй мировой войны и приравнивалось оно к словам карпулинг и каршеринг. Все они означали одно - водитель берет попутчика который оплачивает часть стоимости бензина. Спустя некоторое время начали разделять различные виды и модели.

1.2.1 Понятие каршеринга

Каршеринг – это один из видов краткосрочной аренды авто. Срок аренды в каршеринге может составлять любое количество времен, чего не встретишь в обычном прокате автотранспорта.

Данный вид аренды развился относительно недавно с повсеместным применением Интернета и смартфонов, ведь чтобы арендовать автомобиль по каршерингу не обязательно приезжать в офис и заключать договор, все это делается заранее через Интернет, а машина берётся из припаркованных авто, указанных в приложении.

Каршеринг можно разделить на четыре модели:

- Roundtrip.

Модель, в которой человек берет автомобиль на специализированной стоянке и обязан вернуть в исходное место.

- One-way.

Модель, в которой человек может оставить автомобиль в том месте, где ему это понадобилось, не нарушая правила ПДД.

- Peer-to-peer.

Это модель, которая позволяет арендовать автомобиль с помощью программы-агрегатора, собирающей предложения владельцев в одном сервисе.

- Fractional.

Модель, в которой транспортное средство находится в собственности какой-либо организации, которая занимается обслуживанием его и предоставляет в пользование людям, входящим в эту организацию.

В России самая популярная модель является One-way.

1.2.2 Карпулинг

Карпулинг – это совместное использование автомобиля, который принадлежит частному лицу (чаще всего водителю). Идея состоит в том, что выбирается путь, который совпадает с маршрутом следования водителя и участники поездки следуют до места на одном автомобиле.

В Карпулинге можно выделить три модели:

- Классическая.

Модель, которая подразумевая собой поездки на длительные расстояния.

- Динамическая.

Модель, которая конкурирует с такси (внутригородские поездки.)

- Регулярная.

Модель, в которой люди изо дня в день повторяют один и тот же путь.

Приложение направленно на классическую модель карпулинга.

1.3 Анализ существующих решений

BlaBlaCar

Один из крупнейших в мире международных сервисов карпулинга, основаный в 2006 году французом Фредериком Маззеллой. Данный сервис действует в 22 странах и насчитывает около 50 миллионов пользователей. В основном BlaBlaCar работает для междугородних поездок, а расходы на дорогу, по правилам карпулинга, делятся пропорционально среди всех участников поездки.

На территории Российской Федерации данный сервис распространился, выкупив сервис карпулинга BeepCar от Mail.Ru Group и на данный момент является одним из самых популярных сервисов карпулинга в стране.

Основными преимуществами данного решения являются:

- Интеграция с мобильным устройство
- Низкая стоимость поездки за счет разделения стоимости между участниками поездки
- Узнаваемость на рынке

К недостаткам можно отнести:

- Трата времени на ожидание сбора всех участников поездки
- В основном для междугородних поездок (в связи с настройкой приложения BlaBlaCar занял свою нишу на арене междугородних поездок)

dovezu.ru

Сайт, для попутчиков по все Российской федерации.

Из функционала, который представлен: можно авторизоваться и опубликовать объявления, в котором указывается маршрут следования. Так же можно произвести поиск по сайту и найти нужный маршрут из представленных объявлений.

Достоинства :

- Работает по всей России
- Указываются промежуточные станции

Недостатки:

- Не оптимизировано под мобильную версию
- Не удобный поиск попутчика/водителя (Объявление состоит из набора информации, не указывается актуальность объявления, нельзя подать заявку на поездку, можно только связаться с помощью сообщений)

Яндекс.Попутка (Заморожен с 2018 года)

Яндекс.Попутка — сервис, который был запущен компанией Яндекс в 2018 году. Приложение работает с маршрутами от 10 километров, поэтому с помощью Яндекс.Попутки удобно добираться из аэропорта до дома или из пригорода до работы.

Приложение само показывает примерную сумму за поездку. Например, за маршрут длиной от 10 до 20 километров это 50 рублей, от 20 до 30 — 100 рублей.

Достоинтсва:

- Сервис от компании-гиганта, которую знает каждый пользователь интернета

Недостатки:

- Сервис был рассчитан на маршрут от 10 км (Город Красноярск не настолько большой город и поездки на меньшее расстояние сразу отсекаются)
- Минимальная цена (50 рублей - 10 км), которая отпугивала большинство пользователей

1.4 Анализ рынка

1.4.1 Гипотезы

Для того, чтобы проанализировать спрос на услуги карпулинга в городе Красноярске, мною были составлены 5 гипотез, которые легли в основу опроса. Гипотезы:

- Большинство людей имеют систематический маршрут (маршрут, который повторяется изо дня в день)
- Людям не комфортно ездить в общественном транспорте
- Практически все люди достаточно разбираются в телефонах
- Люди могут отказаться от своего личного транспорта и передвигаться на автомобиле в качестве пассажира
- Люди на личном автомобиле не прочь брать с собой попутчиков

Опрос был проведен мной с использованием Google forms в социальных сетях (Вконтакте, Instagram, Facebook, Одноклассники) с людьми разного возраста и пола.

1.4.2 Интервью

Результаты интервью:

Рисунок 1 – Опрос

Рисунок 2 – Опрос

Рисунок 3 – Опрос

Рисунок 4 – Опрос

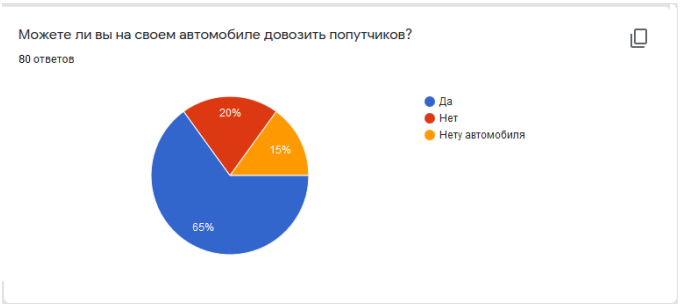

Рисунок 5 – Опрос

Вывод: анализируя полученные результаты опроса (рис. 1-5), у меня подтвердились все гипотезы. Конечно, далеко не все люди готовы садить к себе в автомобиль незнакомых людей, не все люди готовы отказаться от личного транспорта, но главное, что есть довольно большая часть людей, которые готовы пойти на такие шаги.

Исходя из всего вышесказанного, приложение по поиску водителей и попутчиков имеет место быть в городе Красноярске.

1.5 Технологии и средства для разработки приложений

1.5.1 ОС Android

Android – операционная система, основанная на ядре Linux и собственной реализации виртуальной машины Java (JVM от Google). Данная операционная система поддерживает более 100 языков, в том числе и русский. [1]

Операционная система Андроид — это программное ядро более 60% смартфонов и других устройств, выпущенных по всему миру. Благодаря ей стало возможным производство «портативного компьютера», дающего своему владельцу доступ к любым медиафайлам и сети интернет. [2]

Первое устройство с операционной системой Android было выпущено в далеком 2008 году. За эти годы операционная система изменилась до неузнаваемости и успешно развилась до того, что данную систему выбирают практически все компании, выпускающие современные мобильные гаджеты начиная

от устройств, устанавливающихся на автомобили и заканчивая смарт- телевизорами для дома. Доля использования данной операционной системы среди мобильных устройств составляет более чем 80% по всему миру, а со времени выпуска было активировано более 2 миллиардов устройств на данной операционной системе. И как следствие, за этот период появилось множество производительных и эффективных инструментов разработки, которые позволяют создавать качественный продукт, при минимальных трудозатратах на разработку. Стоит отметить и то, что все стандартные инструменты разработки доступны бесплатно и все что вам нужно это рабочий персональный компьютер. [3]

Учитывая вышеизложенное, можно считать, что операционная система Android является одним из лучших выборов для разработки приложения.

1.5.2 Язык программирования Java

Язык программирования Java – это сильно типизированный объектно-ориентированный язык, разработанный американской компанией Sun Microsystems. На 2019 год является самым популярным из языков программирования.

История создания языка уходит в далекий 1995 год, когда он назывался еще ‘Oak’ что в переводе означает дуб. Разрабатывался данный язык Джеймсом Гослингом для простого программирования бытовых электронных устройств. Но язык программирования с таким названием уже существовал и вскоре данный язык переименовался в хорошо уже знакомую Java, в дань уважения марки кофе которое пили разработчики в то время. В связи с этим на официальном логотипе языка изображена чашка с горячим кофе. [4]

Работает данный язык следующим образом – программы, написанные на Java, транслируются в байт код Java выполняемый виртуальной машиной Java (Java Virtual Machine – JVM).

JVM – программа, которая обрабатывает байт код и передает инструкции оборудованию, обычно процессору, как интерпретатор.

Основным достоинством подобного способа выполнения является независимость байт кода от операционной системы и оборудования, что позволяет выполнять программы, написанные на языке Java на любом устройстве, где установлена соответствующая виртуальная машина. Другой не менее важной особенностью является и то, что выполнение программы контролируется виртуальной машиной, что позволяет иметь гибкую систему безопасности, например любые операции, которые превышают полномочия программы, вызывают немедленное прерывание данной программы. [5]

1.5.3 Среда разработки Android Studio

Перед тем как начинать разработку проекта, необходимо установить соответствующую интегрированную среду для разработки.

Интегрированная среда разработки (IDE) – комплекс программных средств, используемый для разработки программного обеспечения. Как правило, среда включает в себя:

- Текстовый редактор
- Компилятор и/или интерпретатор
- Средства автоматической сборки
- Отладчик

Иногда содержит в себе средства для интеграции с системами управления версиями и разнообразные инструменты для упрощения конструирования графического интерфейса пользователя.

Одной из таких IDE является Android Studio разработанная компанией JetBrains, которая включает в себя все вышеупомянутые пункты и является официальным средством разработки Android приложений. (рис. 6) [6]

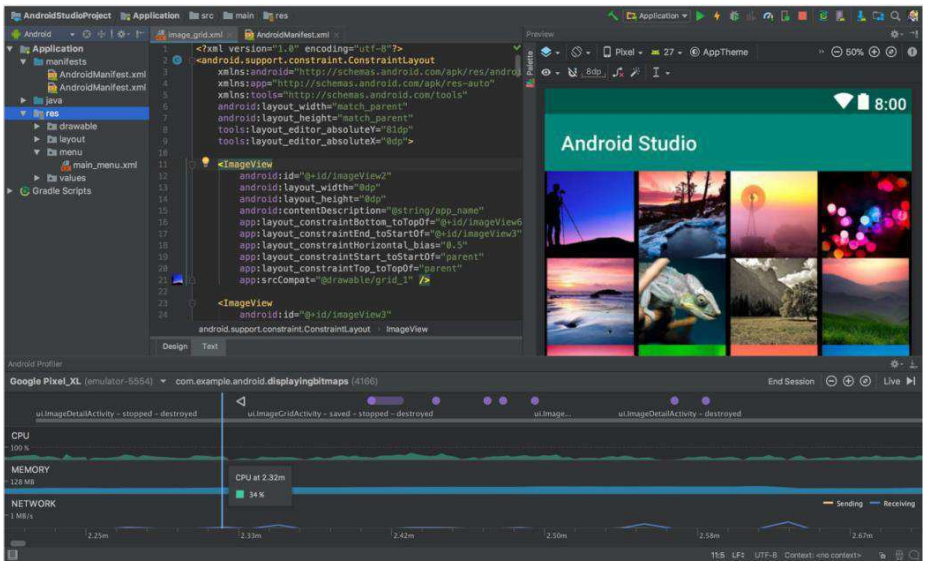

Рисунок 6 – Средство разработки Android Studio

Средство разработки Android Studio содержит в себе мощный редактор кода, удобные шаблоны и мастера создание проектов, возможность в стадии разработки увидеть, как приложение будет выглядеть на разных экранах, а также большой выбор виртуальных устройств – эмуляторов, для тестирования приложений

Поддерживаемые языки программирования: Java, Kotlin, C++. При создании проекта среда разработки Android Studio генерирует необходимую первоначальную структуру, которая включает в себя создание папок и файлов и размещение их в соответствующей структуре иерархии (рис. 7).

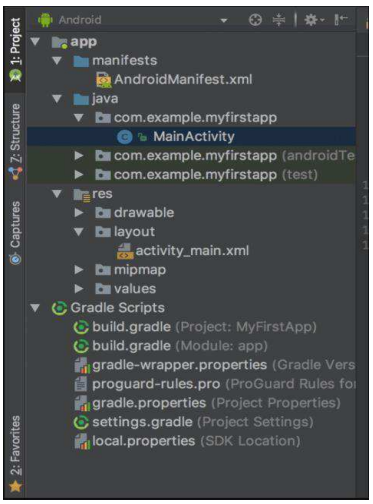

Рисунок 7 – Структура иерархия в Android Studio

Стоит уделить внимание файлу AndroidManifest.xml, который содержит важную информацию о приложении, которая требуется системе Android. Только получив данную информацию, система сможет выполнить какой либо код приложения. Например, информация о пакетах, компонентах, теме, названии и логотипе приложения. Главное окно, оно же точка входа, задается именно в AndroidManifest и по умолчанию им служит MainActivity, но его можно поменять на любое другое окно приложения в зависимости от требований. [7]

Также, немаловажным компонентом среды разработки Android Studio является система автоматической сборки Gradle. Всякий раз, когда создается проект в Android Studio, система сборки автоматически генерирует все необходимые файлы сборки Gradle, такие как:

- Файл сборки верхнего уровня верхнего уровня

Каждый проект Android Studio содержит один файл сборки Gradle верхнего уровня. Этот файл build.gradle является первым элементом, который появляется в папке «Сценарии Gradle» и имеет четко обозначенный Project.

- Файлы сборки уровня на уровне модуля

В дополнение к файлу сборки Gradle на уровне проекта каждый модуль имеет собственный файл сборки Gradle.

- Другие файлы Gradle

В дополнение к файлам build.gradle папка Gradle Scripts содержит некоторые другие файлы Gradle. В большинстве случаев они будут автоматически обновляться при внесении любых изменений в проект.

1.5.4 Сборка проектов на языке Java

Gradle – система автоматической сборки, построенная на широко используемых принципах Apache Maven и Apache Ant, но использующая предметно ориентированный язык Groovy. [8] Процесс развертывания Gradle (рис. 8)

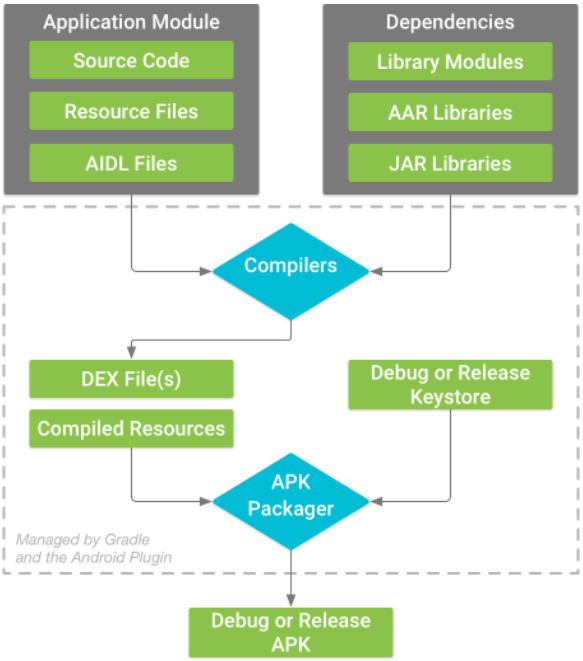

Рисунок 8 – Процесс развертывания

Зачем это вообще нужно? Ручная сборка проектов на Java довольно трудоёмкий процесс. Нужно правильно указать нужные проекту библиотеки и фрэймворки, от которых проект зависит. Тогда-то и приходят на помощь системы сборки проектов, которые автоматизируют этот процесс.

2 Структура проекта

2.1 Проектирование модулей и сервисов

Первоначально для создания приложения, требуется разработать архитектуру приложения, которая будет отвечать всем стандартам разработки программного обеспечения. Правильно спроектированная архитектура приложения позволяет избежать серьезных уязвимостей в коде, оптимизировать время выполнения работ, улучшить качество проекта и позволит создать проект, который можно будет легко поддерживать и расширять в будущем.

Требуется спроектировать модули и сервисы, визуально показать их зависимости между собой и проанализировать результат.

Функциональность проекта должна включать в себя следующее:

- Регистрация пользователей
- Аутентификация и авторизация пользователей
- Выход пользователя из информационной системы
- Работа с персональными данными пользователя (просмотр и редактирование).
- Работа с фотоизображением пользователя, включающая выбор фотоизображения из внутреннего хранилища устройства или использования камеры для получения фотоснимка
- Поиск, удаление и создание поездки с информацией об автомобиле, пути следования, дате выезда и описании поездки
- Выбор точки отбытия и прибытия через систему геолокации и графических карт
- Обработка поступающих заявок от попутчиков на участие в поездке
- Просмотр начавшейся поездки с выводом информации о текущих данных поездки, в которое входит – текущее местоположение транспортного

средства, пути следования, информация об оставшемся времени и расстоянии следования

Проанализировав требуемую функциональность для приложения, была составлена его схема (рис. 9). Практически все модули приложения взаимосвязаны друг с другом. Причем связь их взаимообратная это означает, что из любого модуля можно перейти в связанный модуль и обратно.

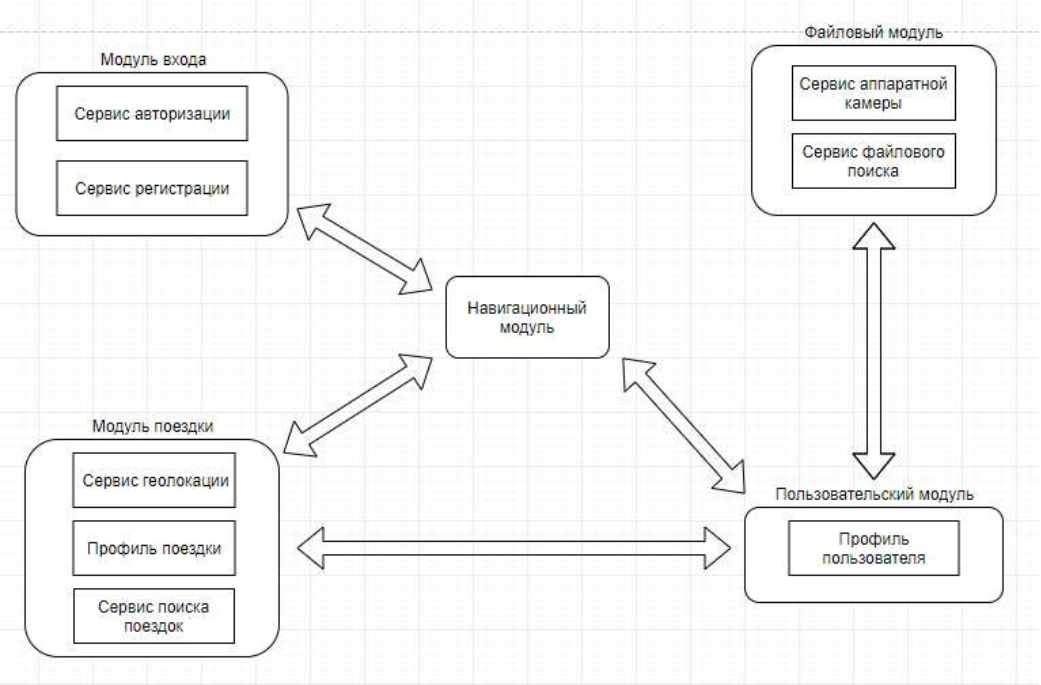

Рисунок 9 – Архитектура приложения

Каждый модуль отвечает за определенный функционал приложения:

− Модуль входа – данный модуль предназначен в первую очередь для работы с аутентификационными данными пользователя. В него входит сервис авторизации, в котором включена форма для ввода аутентификационных данных уже созданого пользователя и сервис регистрации, для регистрации новых пользователей.

− Навигационный модуль – служит для навигации в системе приложения и перехода между основными модулями системы

− Пользовательский модуль – модуль для просмотра и редактирования пользовательской информации.

− Файловый модуль – предназначен для работы с файловой системой ОС Android, а также работы с камерой устройства для получения фотоснимков профиля.

− Модуль поездки – самый большой модуль, который обрабатывает всю логическую модель, связанную с геолокацией, графическими картами и поездками пользователей.

В приложении пользователи не будут разделяться на роли и могут, как создавать поездки, так и подавать заявку на участие в поездки в качестве перевозчика. Данное решение позволяет избежать дополнительной нагрузки на пользователя в плане регистрации в качестве одной или другой роли.

2.2 База данных NoSQL

В проекте будет использоваться облачная нереляционная база данных (NoSQL) от Firebase. Данное решение набирает популярность за счет развития облачных технологий, их повсеместному использованию и легкости встраивания. База данных от Firebase выделяет относительная простота технологий построенных на модели данных ‘ключ’ – ‘значение’ позволяет сделать масштабируемое и производительное мобильное приложение. [9]

Сама структура базы данных в интерпретации в реляционную версию выглядит следующим образом (рис. 10)

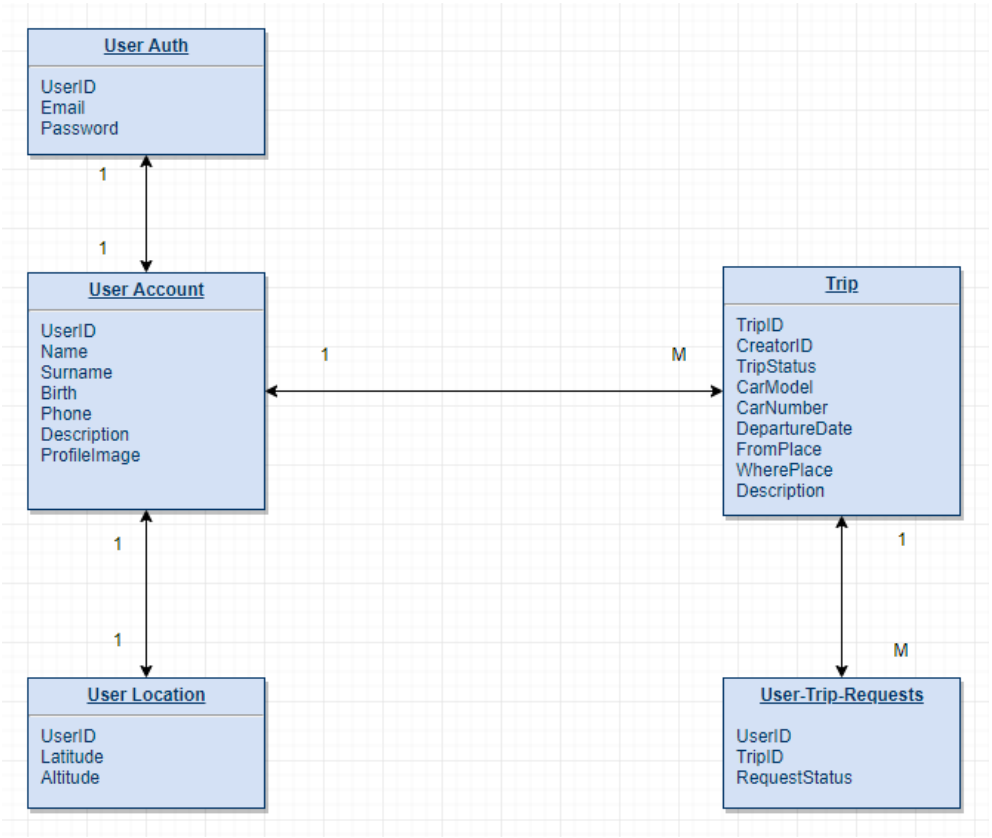

Рисунок 10 – Структура базы данных

База данных в реляционной интерпретации будет содержать следующие сущности:

UserAuth. Сущность, отвечающая за хранение аутентификационных данных пользователя, в которые входит следующие поля:

- UserID – идентификационный номер пользователя
- Email – электронная почта пользователя
- Password – захэшированный пароль пользователя

UserLocation. Сущность, отвечающая за хранение информации о текущем местоположении пользователя. Содержит следующие поля:

- UserID – идентификационный номер пользователя
- Latitude – данные о географической широте
- Altitude – данные о географической долготе

UserAccount. Сущность, содержащая основную информацию о пользователе системы:

- UserID – идентификационный номер пользователя

- Name – имя пользователя
- Surname – фамилия пользователя
- Birth – дата рождения
- Phone – контакный номер
- Description – дополнительное описание о пользователе
- ProfileImage –фотоизображение пользователя

Trip. Сущность поездки, содержащая в себе всю информацию о поездке:

- TripID – идентификационный номер поездки
- CreatorID – идентификационный номер создателя поездки
- TripStatus – статус поездки
- CarModel – модель транспортного средства
- CarNumber – государственный регистрационный номер транспортного средства
- DepartureDate – дата начала поездки
- FromPlace – место отбытия
- WherePlace – место прибытия
- Description – дополнительная информация о поездке

User-Trip-Requests. Данная сущность содержит информацию о заявках попутчиках на участие в поездке:

- UserID – идентификационный номер перевозчика
- TripID – идентификационный номер поездки
- RequestStatus – статус заявки

Отсюда следует, что в структуре базы данных будет присутствовать две основные сущности – это пользователь и поездки. Между ними связь один ко многим, то есть один пользователь может создать множество поездок, и в тоже время, у одной поездки может быть только один пользователь, он же создатель.

У каждого пользователя имеются аутентификационные данные со связью один к одному, то есть у одного пользователя могут быть только одни аутентификационные данные, и в тоже время у одних данных аутентификации может быть только один пользователь.

У каждого пользователя есть его текущие координаты – географическая ширина и долгота. Связь между географическими данными и пользователем один к одному.

У каждой поездки может быть множество заявок от попутчиков желающих добраться до выставленного пункта назначения, в тоже время у одной заявки может быть только одна поездка.

3 Программная реализация

3.1 Множественные окна одного приложения

Реализовать связь между модулями возможно с помощью одной из функции в ОС Android - множественных окон одного приложения. То есть окна в Android не закрываются после перехода в другое окно, оно может быть оставлено открытым, по усмотрению разработчика. Данный функционал реализуется при помощи жизненного цикла окна, и у каждого такого окна есть свой собственный жизненный цикл (рис. 11).

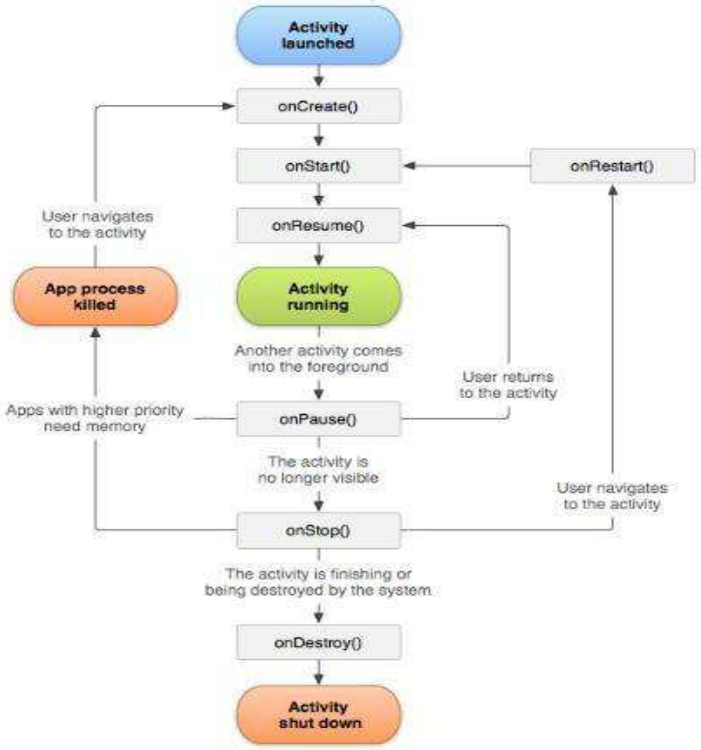

Рисунок 11 – Основные жизненные циклы окна

На данном графике показана упрощенная версия основных жизненных циклов окна. Как видно у окна есть 7 основных состояний:

− onCreate() – выполняется сразу же при программном создании окна

- onStart() – выполняется при запуске окна, когда окно становится видимым пользователю
- onResume() – выполняется после старта единожды а также каждый раз после того как пользователь обращается к данному окну
- onPause() – выполняется как индикация того, что пользователь переключился на другое окно, но это еще не означает что окно исчезло из поля зрения пользователя.
- onStop() – выполняется после того как окно окончательно исчезло из поля зрения пользователя
- onDestroy() – при полном закрытии окна или его программном завершении
- onRestart() – при переходе пользователя обратно к окну, когда окно появляется в поле зрения пользователя.

3.2 Модуль входа

Точкой входа в приложение является модуль входа. Для разработки модуля требовалось подключить внешнюю зависимость Firebase компонента – Firebase Authentication через систему автоматической сборки Gradle. Данный компонент отвечает за аутентификацию, регистрацию и хранение аутентификационных данных пользователя в облачном хранилище Firebase.

После этого требовалось разработать пользовательский интерфейс входа в приложение и реализовать сервисы аутентификации и авторизации пользователя.

Стоит упомянуть, что при разработке визуальной составляющей пользовательского интерфейса использовались специальный конструктор (рис. 12), встроенный в Android Studio, а также программное описание в XML формате.

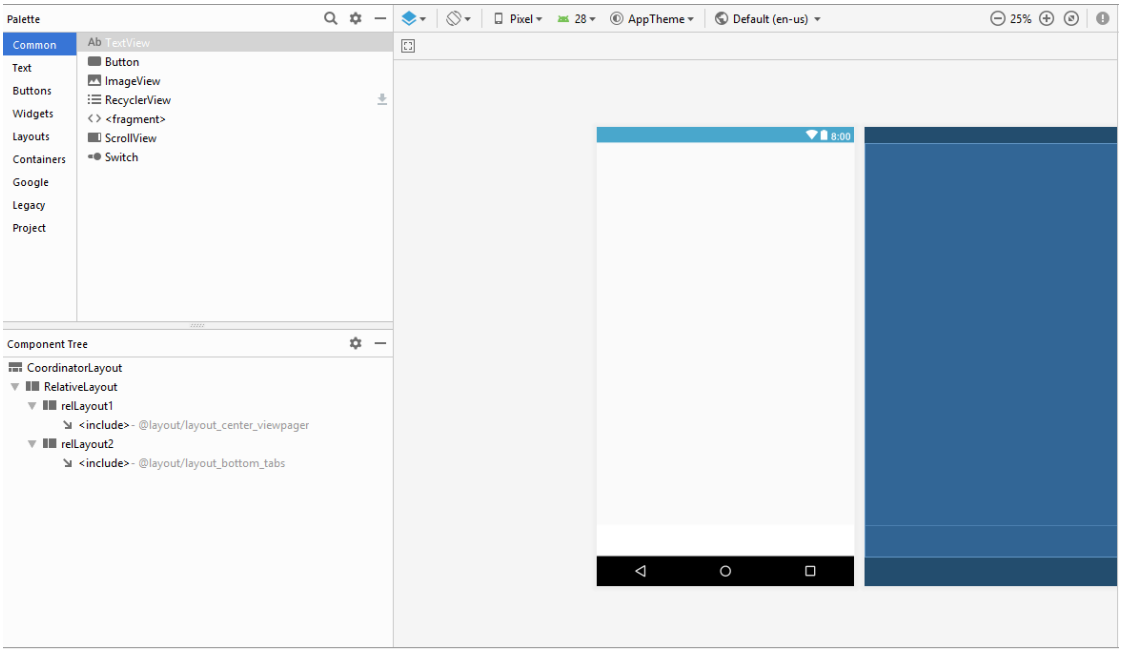

Рисунок 12 – Конструктор встроенный в Android Studio

Форма входа (рис. 13) состоит из двух полей для ввода данных электронной почты и пароля, а также одной кнопки и одной гиперссылки, которая перенаправляет к форме регистрации в приложении

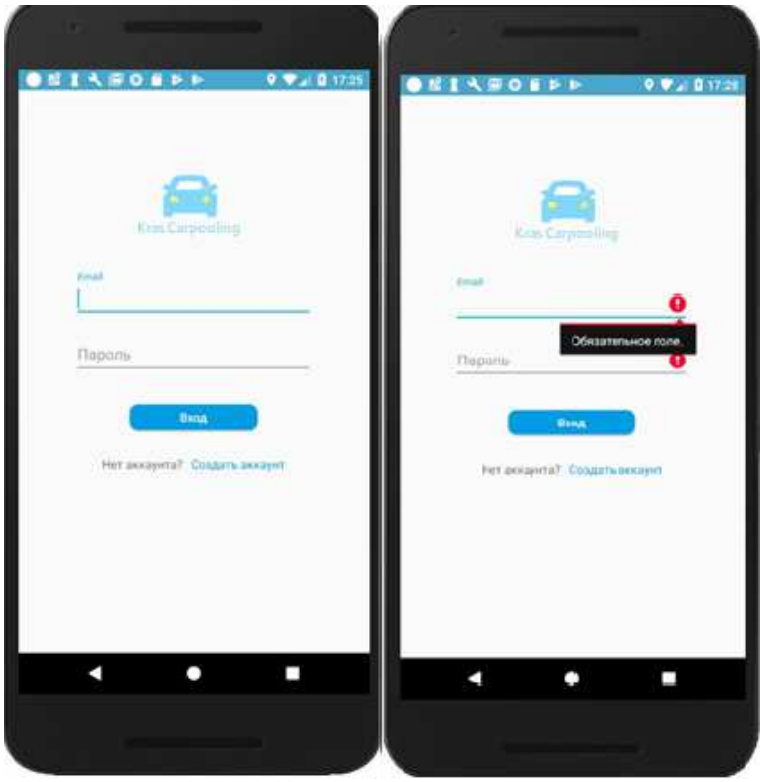

Рисунок 13 – Форма входа и демонстрация ошибки

При нажатии кнопки ‘Вход’ происходит валидация данных и, если данные заполнены неверно или незаполненные вовсе, пользователю выводиться сообщение об ошибке над полями которые заполнены некорректно.

Далее, если все данные заполнены корректно, они отправляются на аутентификацию в Firebase и происходит подписка на событие завершения проверки системой Firebase Auth – addOnCompleteListener.

В ответном сообщении приходит результат (AuthResult) проверки, и, если результат проверки успешен, происходит авторизация пользователя в систему и переход в модуль навигации.

Сервис регистрации также содержит в себе форму для заполнения и внутренние программные средства и методы для регистрации нового пользователя. Перейти к регистрации можно используя кнопку ‘Создать аккаунт’ на форме входа (рис. 14). Данное действие перенаправит пользователя к форме регистрации где потребуется ввести пользовательские регистрационные данные.

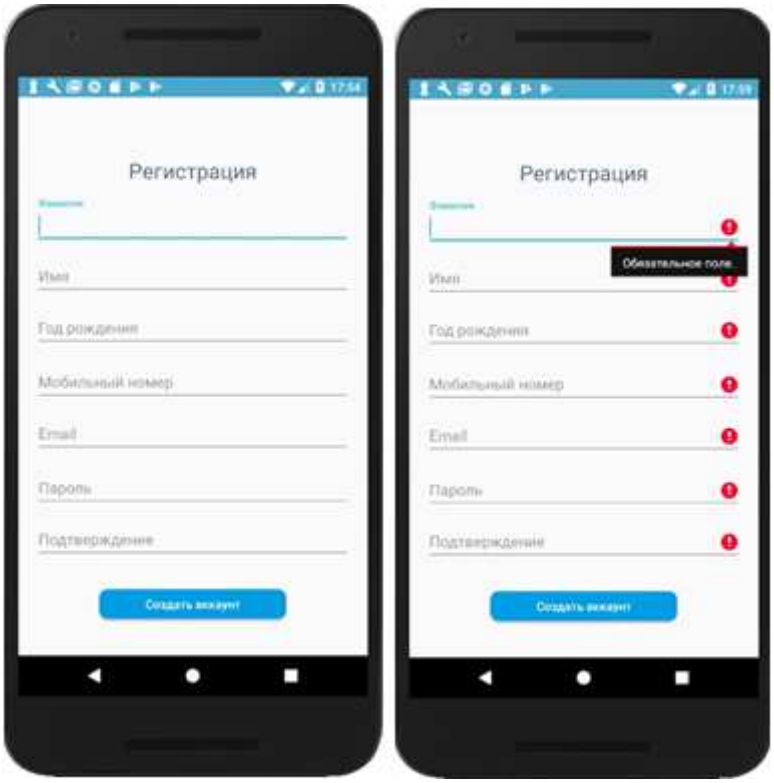

Рисунок 14 – Форма регистрации и демонстрация ошибки

Форма содержит в себе следующие поля для ввода:

- Фамилия
- Имя
- Год рождения
- Мобильный номер
- Email
- Пароль
- Подтверждение пароля

А также кнопку ‘Создать аккаунт’ для валидации и отправки вводимых данных.

После прохождения валидации данные считываются с полей ввода, и записываются в объект специально созданного для работы с пользовательской информацией класса UserAccount.

Эти данные отправляются в базу данных для создания аккаунта с подпиской на событие завершения регистрации addOnCompleteListener метода createUserWithEmailAndPassword. Если регистрация прошла успешно, то приложение перенаправляет пользователя на главное окно навигационного модуля, иначе приложение выведет информацию о провале регистрации пользователю. (рис. 17).

3.3 Модуль навигации

Для реализации навигационного модуля был разработан соответствующий пользовательский интерфейс и подключена необходимая для работы библиотека BottomNavigationViewEx. Ее можно найти на Open-Source портале GitHub [10]. Данная библиотека помогает в короткие сроки разработать анимированное навигационное меню с множеством вариаций анимации.

После подключения библиотеки, следующим шагом стало разработка пользовательского интерфейса навигационного модуля.

Навигационный модуль поделен на две части.

– Первая часть отвечает за нижнее навигационное меню переключения между окнами.

– Вторая же часть – основное навигационное меню, которое помогает ориентироваться в созданных поездках и поездках, на которые вы подали заявку в качестве водителя.

Для нижнего навигационного меню требовалось создать соответствующие иконки переходов. Это можно сделать при помощи встроенного генератора иконок Android Studio – Asset Studio, который генерирует векторные изображения формата SVG или изображения формата PNG (рис. 15).

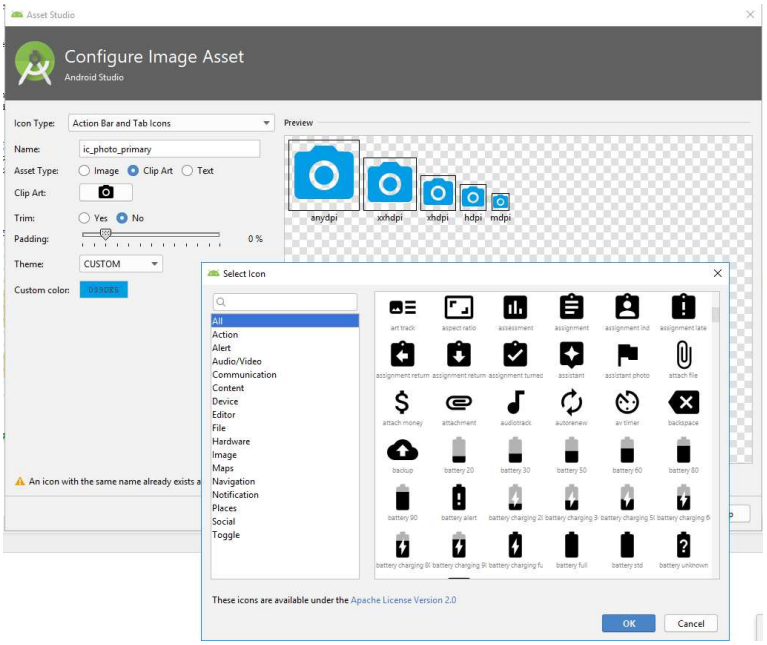

Рисунок 15 – Встроенный генератор иконок

Во время генерации изображений Android Studio открывает каталог drawable и формирует в нем новый каталог с названием сгенерированного изображение, после чего вносит изображение в данный каталог с разными вариантами разрешений изображения (рис. 16).

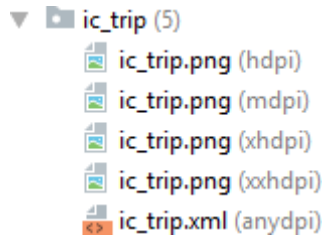

Рисунок 16 – Каталог drawable

Далее можно создать соответствующее меню, через описание XML, с иконками нужных изображений и подключить данное меню в навигационное меню библиотеки BottomNavigationViewEx.

Следующим этапом требовалось разработать основное навигационное меню. По условию, может быть несколько поездок на одного пользователя, отсюда следует, что требовалось разработать пользовательский интерфейс формы списка поездок и разработать сервис по отображению данных в созданный список.

Для этого был создан специальный класс KPTripAdapter и TripListInfo. В задачи класса TripListInfo входит хранение данных о поездке. В свое время как в KPTripAdapter должны поступать списки с экземплярами класса TripListInfo. После инициализации KPTripAdapter выполняется один из методов класса ListView - setAdapter с входящим параметром, который реализует базовый класс ArrayAdapter.

К слову ListView хранит в себе программную интерпретацию пользовательского интерфейса списка, который был разработан при помощи XML разметки. Метод setAdapter класса ListView позволяет установить класс описания (в данном случае KPTripAdapter) того, как должны быть расположены входящие данные.

Также после установки адаптера мы ‘вешаем’ обработчик события клика на элемент списка. Этот обработчик получает информацию о том, на какую именно поездку из списка нажал пользователь и переводит его к соответствующей поездке.

В итоге получаем следующий готовый навигационный модуль (рис. 17). Стоит упомянуть, что может быть ситуация, когда у пользователя нет планируемых поездок, и требуется также обработать и этот случай.

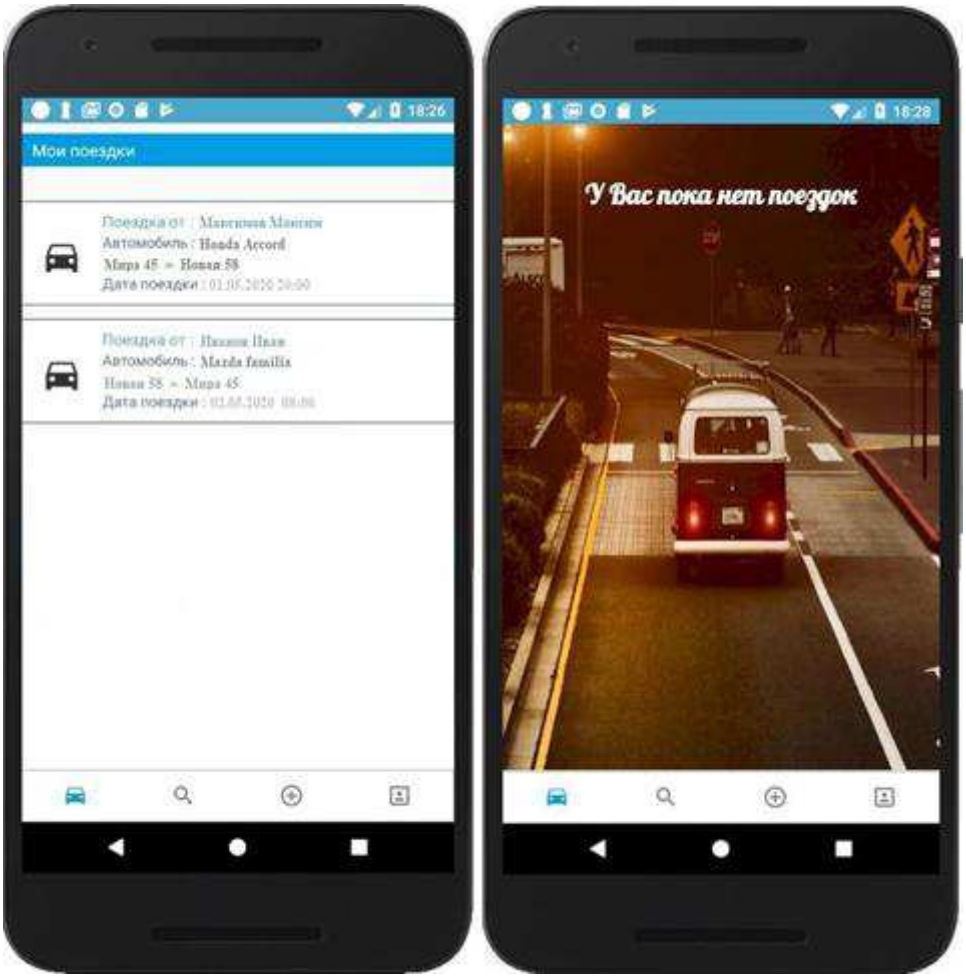

Рисунок 17 – Навигационный модуль

3.4 Модуль пользователя

Данный модуль состоит из двух основных разделов:

− Первый – профиль пользователя, где можно просматривать как свой профиль, так и профиль другого пользователя, с разной вариацией компоновки управляющих элементов

− Второй раздел - раздел редактирования пользовательской информации, где каждый пользователь может отредактировать личную информацию

Перейти к пользовательскому модулю можно нажав на соответствующий значок (крайний справа элемент) в нижнем навигационном меню. При переходе к разделу профиль пользователя - формируются данные о пользователе посредствам запроса к облачной базе данных Firebase программным методом getUserInfo класса ProfileActivity, и затем данные подставляются в соответствующие поля методом initProfileUI.

Для своего профиля (рис. 18) доступны управляющие элементы ‘редактировать’ и ‘выйти’. Для всех остальных профилей данные элементы скрыты.

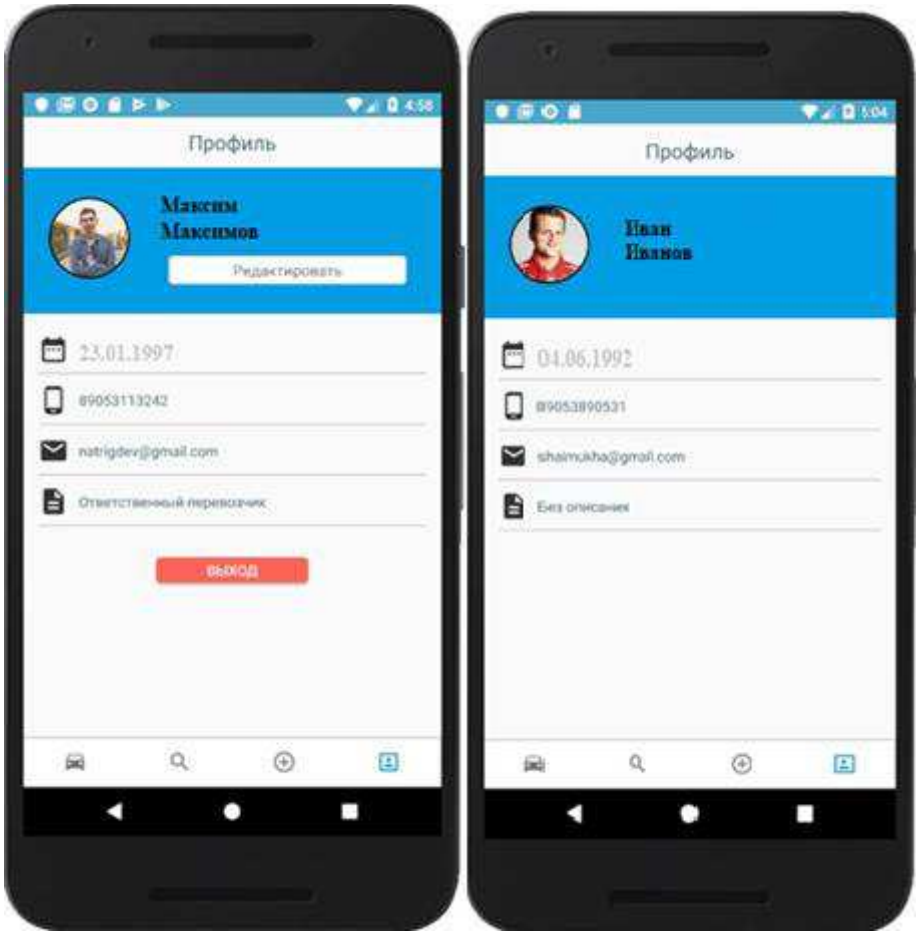

Рисунок 18 – Демонстрация своего профиля и для остальных

При нажатии кнопки ‘Выход’ происходит разлогирование с системы Firebase Auth и пользователя переводят в модуль входа.

При нажатии кнопки ‘Редактировать’ происходит переход в раздел редактирование (рис. 19), где пользователь может изменить свою персональную информацию, а также изменить свое профильное фотоизображение.

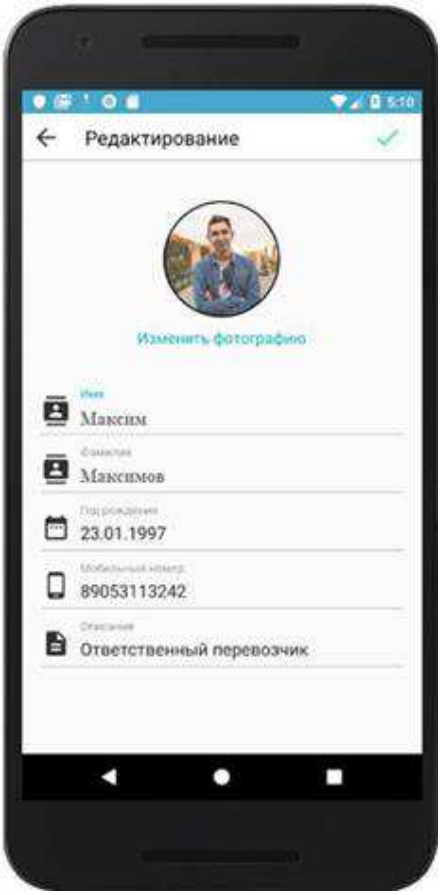

Рисунок 19 – Раздел редактирование

Следует отметить, что для облегчения ввода года своего рождения, был подключен специальный селектор даты (рис. 20)

Рисунок 20 – Селектор даты

Для изменения своего профильного фотоизображения требуется нажать на кнопку ‘Изменить фотографию’, которая произведет переход к файловому модулю

3.5 Файловый модуль

Файловый модуль, также как и пользовательский модуль состоит из двух разделов:

- Первый раздел – раздел фотогалереи, где пользователь может выбрать фотоизображение из уже существующих на данном устройстве изображений.
- Второй раздел – раздел фотокамеры, где пользователь может использовать фотокамеру устройства для снимка фотографии на профильное фотоизображение.

Для удобного перехода между разделами было использовано нижнее навигационное меню (рис. 21).

При переходе в модуль, приложение запрашивает доступ к камере устройства, а также файловому хранилищу.

Рисунок 21 - Файловое хранилище

Это было реализовано следующим образом: в первую очередь требовалось указать в AndroidManifest, какие разрешения нам потребуются, а это – разрешение на использование камеры, разрешение на чтение и запись внутреннего хранилища.

Далее идет проверка, выданы ли запрашиваемые разрешения, методом checkPermissionsArray (если разрешений несколько) или checkSinglePermission для одного конкретного разрешения. Если в ответе приходит false, это означает что какое то разрешение не выдано, для того чтобы выдать разрешение выполняется метод verifyPermissions.

После выдачи всех прав, можно приступать к работе с файловым хранилищем.

Раздел состоит из трех основных блока:

− Верхний навигационный блок, который позволяет закрыть модуль или выбрать изображение для установки фотоизображения, а также кнопки навигации по внутреннему хранилищу устройства.

Для навигации по внутреннему хранилищу, а также сбору информации об изображениях каждого раздела внутреннего хранилища был создан специальный класс FileSearch. Данный класс содержит в себе два метода, первый метод – getDirectoryPath который принимает путь к разделу, с которого собирает все пути к внутренним разделам текущего раздела, второй метод – getFilePath собирает пути к файлам текущего раздела. Для верхнего навигационного блока нам потребовалось использовать getDirectoryPath, чтобы составить все возможные пути к разделам внутреннего хранилища.

− Центральный блок – блок, содержащий в себе выбранное изображение и GridView (табличный список) всех изображений текущего раздела, где пользователь может выбрать нужное ему изображение.

− Нижний навигационный блок – блок перехода между разделами файлового модуля, который был описан выше.

При нажатии на элемент ‘Камера’, нижнего навигационного блока, система откроет камеру устройства для фотоснимка (рис. 22).

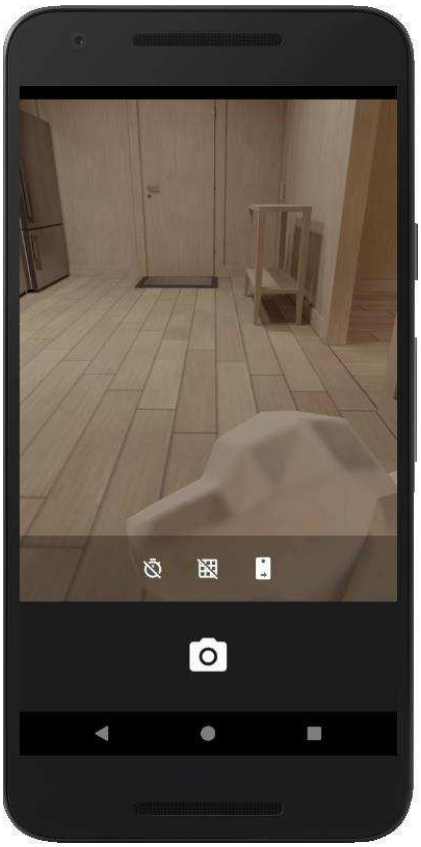

Рисунок 22 – Камера устройства

После фотоснимка или выбора фото и нажатия кнопки ‘Далее’, выбранное изображение переводится в формат bitmap, а после в байтовый массив, который после загружается в облачное файловое хранилище Firebase Storage.

Для работы с конвертацией файлов был создан специальный класс ImageManager. Данный класс содержит в себе методы getBitmap для конвертации выбранного файла в bitmap и метод getBytes, который переводит bitmap файл в байтовый массив с установленным значением качестве изображения. Методы данного класса вызываются в методе uploadProfilePhoto, который отвечает за загрузку изображения в Firebase Storage специальным методом класса UploadTask на который ‘вешаются’ три обработчика событий – addOnSuccesListener отрабатываемый при успешной загрузке,

addOnFailureListener отрабатываемый при неудачной загрузке и addOnProgressListener для отображения текущего прогресса загрузки.

3.6 Модуль поездки

Модуль поездки состоит из множества подразделов, описание которых приведено ниже. Данный модуль отвечает за основную логику приложения, связанную с карпулингом. Здесь обрабатываются геоданные, заполняются профиль поездки, обрабатываются входящие заявки на участие в поездке в качестве попутчика.

Первый раздел, который хотелось бы затронуть – раздел основной информации. Данный раздел отвечает за ввод основной информации о поездке. К нему относится выбор точки отправления и точки назначения, а также выбор даты начала поездки. Выглядит он следующим образом (рис 23):

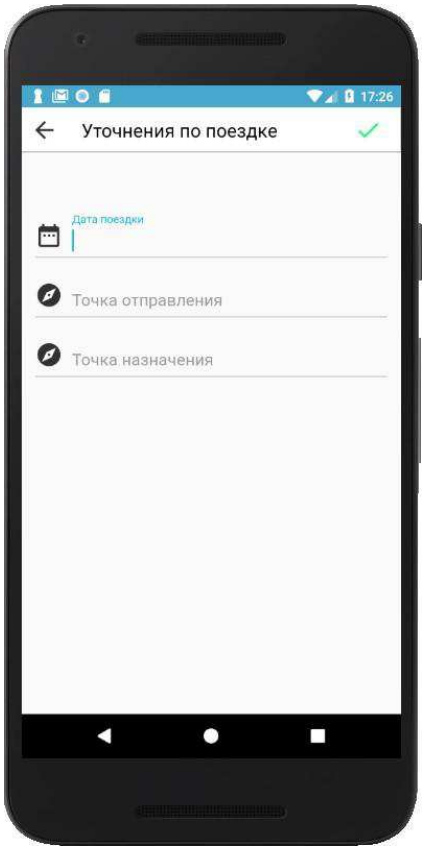

Рисунок 23 – Раздел основной информации о поездке

Как видно, для создания пользовательского интерфейса использовалось все вышеупомянутое, а именно поля ввода, иконки, а также селектор выбора даты (рис. 28). Но это не весь функционал данного раздела. Основным функционалом данного раздела является сервис карт, который открывается при двойном нажатии на ‘точку отправления’ или ‘точку назначения’.

Для использования карт, также нужны соответствующие разрешения. Для этого надо – указать требуемое разрешение в AndroidManifest и при переходе на карты проверять, выданы ли запрашиваемые разрешения. Если запрашиваемые разрешения не выданы, требуется запросить их, таким же методом как было упомянуто выше.

Следующим шагом требуется подключить сервисы карт Google Maps. Это можно сделать следующим образом: для начала требуется сгенерировать приватный ключ и интегрировать его в проект для начала работы с сервисами Google Maps. Для этого в AndroidManifest в качестве мета данных нужно указать ссылку на ключ, в свою очередь который сгенерирован в директории string под наименованием google_maps_key.

После выдачи разрешения и интеграции ключа можно приступать к работе с Google Maps. Для этого приложение программно создает поле fragment в разметке окна Android для вывода карты.

Карта успешно интегрирована, но это лишь часть задачи. Теперь требуется настроить интерфейс и элементы управления на карте, реализовать возможность выставления маркеров, а также разработать сервис поиска нужной улицы в городе.

Начнем с разработки кнопки, по нажатию которой камера карты переведется к текущей геолокации устройства. К счастью в Google Maps данная кнопка встроена и ее всего лишь требуется программно установить на соответствующую дизайну позицию.

Дальше, требовалось реализовать сервис поиска улицы. Для этого нужно было создать соответствующую разметку и поместить поверх карты. После разработки разметки нужно было установить обработчик события на кнопку

поиска. Так, если кнопку нажали, то вызывается метод geoLocate() который переводит камеру к соответствующей введённой улице.

В первую очередь считывается введённый текст и затем собирается список возможных улиц методом getFromLocationName класса Geocoder. Если список не пуст, то есть, найдена соответствующая улица, выполняется метод moveCamera, который передвигает камеру на введённую улицу

Осталось реализовать выставление маркера на карту. При инициализации карты, мы ‘вешаем’ обработчик событий на клик по карте, при срабатывании которого вызывается метод setLocation(), который проверяет, выставлен ли уже маркер на карте, если маркер выставлен – он перезаписывает координаты маркера на новые, если маркера еще нет – то создается маркер с соответствующими координатами. После данной процедуры, из выставленных географических координат долготы и широты, используя метод getFromLocation класса GeoCoder, находим соответствующую улицу и записываем её в соответствующее поле (рис. 24)

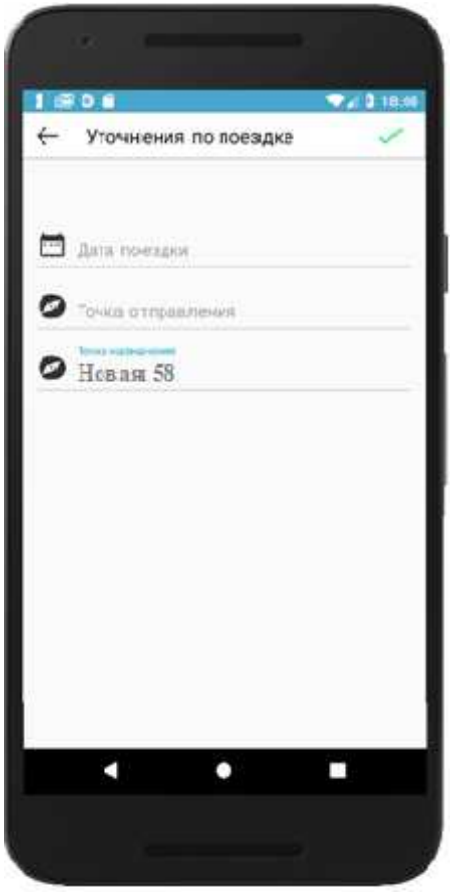

Рисунок 24 – Раздел основной информации о поездке

После того как все поля заполнены можно перейти к следующему этапу в зависимости от того какой пункт был выбран в нижнем меню навигации – поиск поездки или создание поездки. Если выбран пункт ‘поиск поездки’ то выстраивается список по заданным значениям всевозможных открытых поездок, которым нужны попутчики. Принцип построения списка такой же, как был указан выше в описании навигационного модуля. Если же был выбран пункт ‘создать поездку’, то приложение переходит в следующее окно создания заявки (рис. 25) где требуется ввести детали поездки.

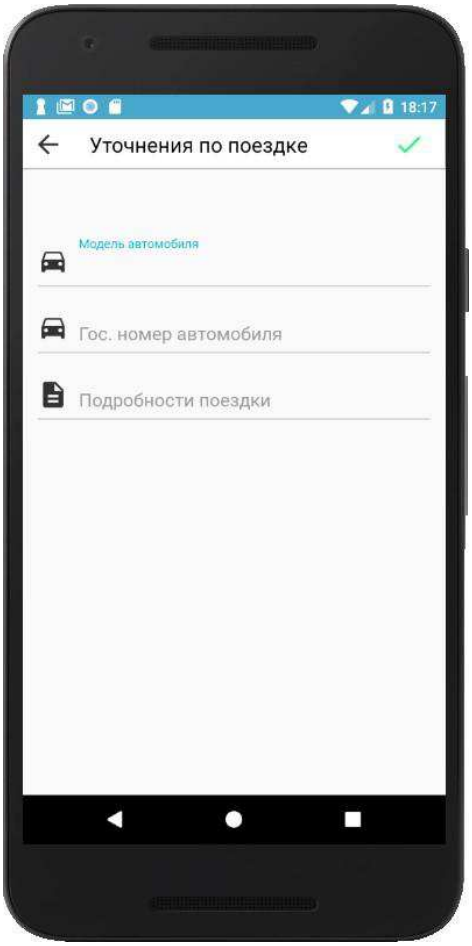

Рисунок 25 – Раздел уточнения по поездке

После ввода всех данных, создается поездка и производится переход во второй раздел модуля, а данные записываются в облачное хранилище Firebase с

присваиванием соответствующего статуса поездки. К слову статусов поездки может быть несколько:

- Поиск попутчика – первоначальный статус, когда поездка открыта для поиска попутчиков
- Ожидание выезда – статус при подтверждении водителя и ожидания начала поездки
- В дороге – статус присваивается, когда водитель начинает поездку
- Закончена – статус окончания поездки

В зависимости от того какой статус имеет поездка у создателя заявки, а также у попутчика отображаются соответствующие управляющие элементы.

Отображение при статусе ‘Поиск попутчика’ от лица создателя поездки и от лица попутчика (рис. 26).

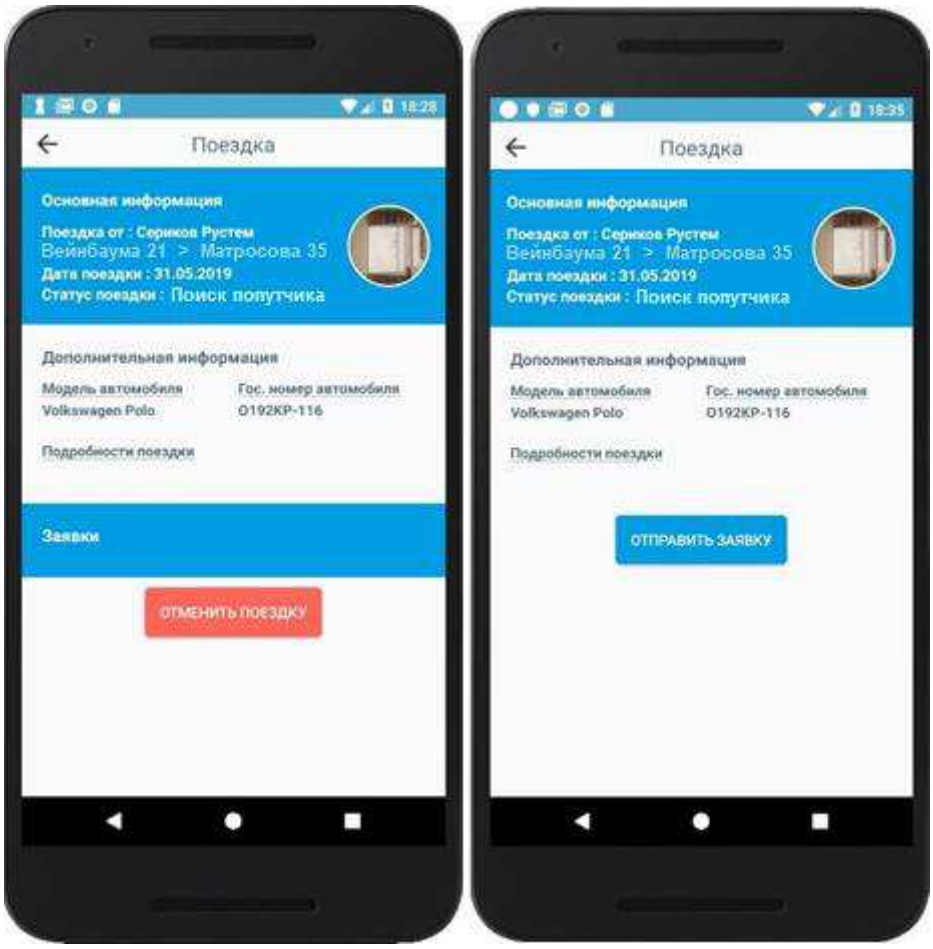

Рисунок 26 – Отображение при статусе «Поиск попутчика»

Также представлены отображения при отправке заявки на поездку при статусе ‘Поиск попутчика’ со стороны создателя поездки и попутчика (рис. 27).

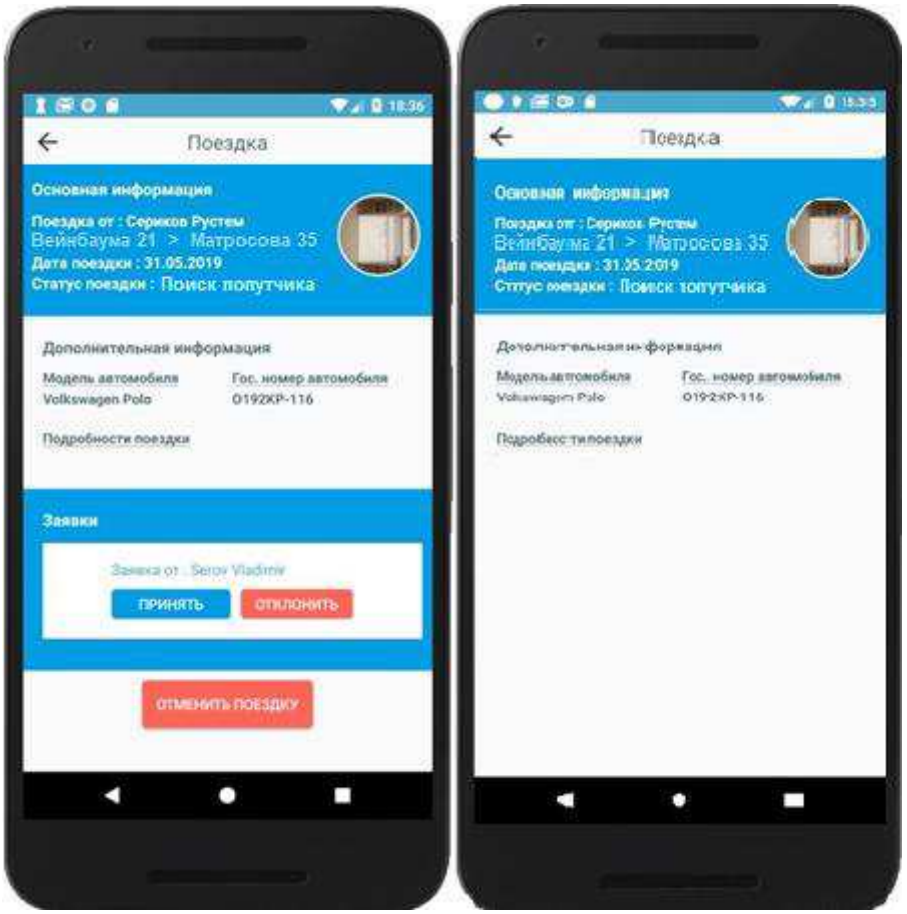

Рисунок 27 – Отображение при отправке заявки

Далее создатель по желанию может, как отклонить заявку, так и принять заявителя, предварительно просмотрев профиль попутчика, кликнув на его фотоизображение.

При отклонении данный пользователь заноситься в список отклоненных заявителей, после чего не сможет больше отправить заявку на данную поездку. Это сделано для избегания возможного ‘спама’ заявками.

Допустим, что создатель одобрил заявления попутчика, и принял его. Далее статус заявки обновляется на ‘Ожидание выезда, в ходе которой отображается информация о попутчике уже в самом профиле поездки. Отображение ин-

терфейса со стороны заявителя и отображение со стороны создателя поездки - водителя (рис 28).

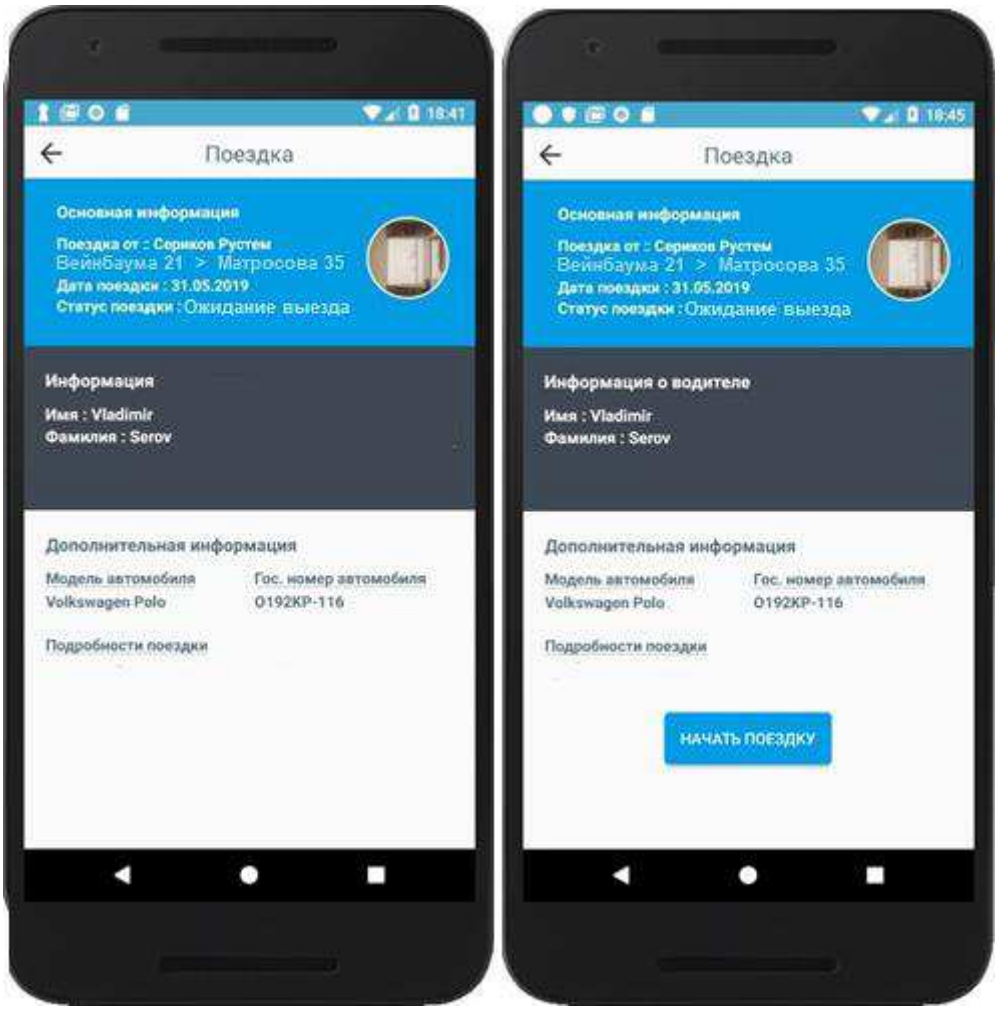

Рисунок 28 – Отображение интерфейса

Причем начать поездку водитель может не раньше даты начала поездки. После того как водитель нажал на кнопку ‘начать поездку’, статус поездки поменяется на статус ‘В дороге’.

Причем водитель не может завершить поездку до того момента пока не пребудет в точку назначения. А попутчик соответственно, может наблюдать за процессом поездки онлайн, нажав на кнопку ‘посмотреть местоположение машины’. После нажатия на данную кнопку мы переходим в третий раздел модуля поездки, а именно раздел отслеживания поездки, где нам открывается карта

поездки с маршрутом поездки, примерным расчётным временем длительности поездки и оставшегося расстояния поездки (рис. 29).

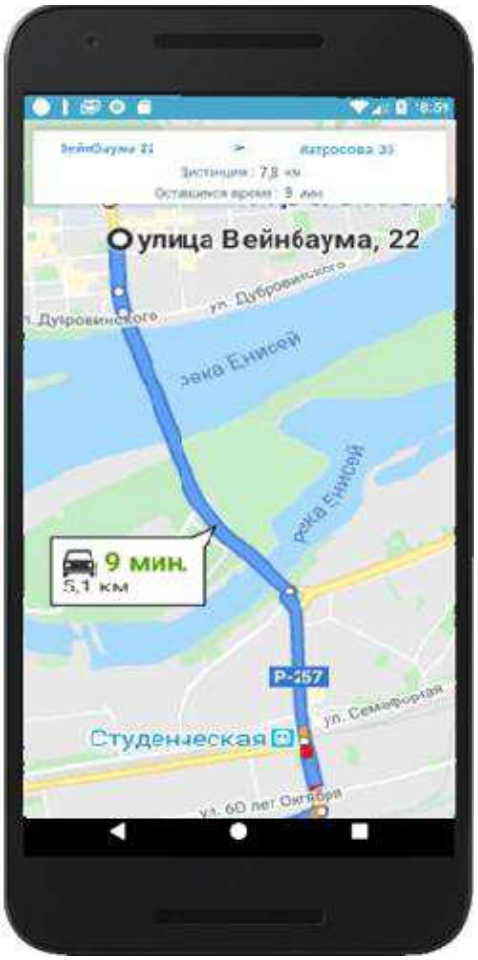

Рисунок 29 – Раздел отслеживания поездки

Работает это следующим образом – в первую очередь требовалось определять в режим реального времени текущее местоположение устройства пользователя. Для этого при запуске приложения запускается специальный код программы setGeolocation(), который через определенный промежуток времени, при изменении координат устройства – ширины и долготы, отправляет их в облачное хранилище Firebase по которым можно с легкостью определить местоположение на карте Google Maps.

И все обновления принимаются картой Google Maps в режиме реального времени, то есть можно буквально каждую секунду наблюдать за движением автомобиля. Далее при помощи сервисов Google Maps – DirectionsAPIRequest

строится соответствующии полилинии (Polylines), с учетом дорожной карты Google Maps и расчитывается приблизительное время поездки, а также расстояние до точки назначения. [11]

ЗАКЛЮЧЕНИЕ

В данной выпускной квалификационной работе был проведен анализ рынка карпулинга в Красноярске, были выделены несколько гипотез и проведено интервью, а так же проанализированы некоторые известные системы, выявлены их достоинства и недостатки.

Был изучен язык программирования и среда разработки приложений Android Studio, рассмотрены источники по теме работы, создано приложение по поиску водителей и попутчиков «Kras - carpooling».

Разработанное приложение позволяет человеку войти в систему, найти поездку с нужными параметрами или же предложить поездку со всей необходимой информацией. А также, для более комфортного пользования, было настроено отслеживание поездки в режиме реального времени.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Дейтел П. Android для разработчиков / Дейтел Х., Дейтел Э. – Санкт-Петербург: Питер: 2016. –384 с.
2. Майер, Р. Android 2. Программирование приложений для планшетных компьютеров и смартфонов / Р. Майер. – Москва: Эксмо, 2011. - 672 с.
3. Документация по разработке в ОС Android [Электронный ресурс] : переводы официальной документации Google по разработке приложения под Android. – Режим доступа: https://developer.android.com/docs, свободный.
4. Герберт Шилдт Java 8. Полное руководство / Герберт Шилдт - Перевод с английского – Москва: ООО "И.Д. Вильяме", 2015. - 1376 с.
5. Харди Б. Android. Программирование для профессионалов / Филлипс Б., Стюарт К., Марсикано К. – Санкт-Петербург: Питер, 2016. — 640 с.
6. Официальная страница Android Studio [Электронный ресурс] : полезные материалы по Android Studio – Режим доступа: http://developer.android.com/sdk/index.html, свободный.
7. Коматинени С. Android 4 для профессионалов. Создание приложений для планшетных компьютеров и смартфонов / Маклин Д. – Перевод с английского — Москва: ООО “И.Д. Вильямс”, 2012. — 881 с.
8. Gradle Build Tool. [Электронный ресурс] : сайт содержит информацию о истеме сборки Gradle. – Режим доступа: http://gradle.org, свободный.
9. Облачное хранилище Firebase. [Электронный ресурс] : вся необходимая информация о работать с Firebase Cloud на Android. – Режим доступа: https://firebase.google.com/, свободный.
10. Веб-сервис хостинга проектов [Электронный ресурс] : материалы для разработки приложения – Режим доступа: www.github.com., свободный.
11. Документация Google Maps API [Электронный ресурс] : переводы официальной документации Google по внедрению карт Google Maps API в проект – Режим доступа: https://developers.google.com/maps/documentation/, свободный.

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Космических и Информационных Технологий
институт
Информационные системы
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой ИС
_____ П.П. Дьячук
подпись инициалы, фамилия
«26» 06 2020 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 – «Информационные системы и технологии»

Разработка Android приложения «Kras-carpooling»

Руководитель _____ доцент, к.т.н. И. А. Легалов
подпись, дата должность, учёная степень инициалы, фамилия

Выпускник 26.06.2020 А.Э. Лутков
подпись, дата инициалы, фамилия

Красноярск 2020