МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра IT-инжиниринг

«ДОПУЩЕН К ЗАЩИТЕ»
Заведующий кафедрой
PhD, доцент
_____ Т.С. Картбаев
«____» _____ 2019 г.

ДИПЛОМНАЯ РАБОТА

На тему: Разработка мобильного приложения для логистической комании

Специальность 5В060200 – «Информатика»

Выполнил Заиров А. Я. Группа ИНФ 15-2

Научный руководитель доктор PhD, доцент _____ Ж. Бидахмет

Консультанты:
по экономической части: к.э.н., доцент _____ А.И. Бекишева
«30» 04 2019 г.

по безопасности жизнедеятельности: д.т.н., ст. преп. _____ Ш.Ш. Бекбасаров
«2» 04 2019 г.

по применению
программного обеспечения: ст. преп. _____ М.Н.Майкотов
«6» 05 2019 г.

Нормоконтролер: ст. преп. _____ Ж.К.Алимсейтова
«8» 05 2019 г.

Рецензент: PhD, доцент, _____ Ж.С. Есенгалиева
«____» _____ 2019 г.

Алматы, 2019

Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Институт Институт систем управления и информационных технологий

Кафедра IT-инжиниринг

Специальность 5В060200 – «Информатика»

ЗАДАНИЕ
на выполнение дипломной работы (проекта)

Студенту Заирову Азизу Ярмухаметовичу

Тема работы (проекта) Разработка мобильного приложения для логистической компании

Утверждена приказом по университету № 124 от «26» 10 2019 г.

Срок сдачи законченной работы (проекта) « 04 » 05 2019 г.

Исходные данные к работе (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): Необходимо разработать мобильное приложение, обеспечивающее эффективный поиск грузов и грузоперевозчиков.

Перечень вопросов, подлежащих разработке в дипломной работе, или краткое содержание дипломного проекта:

а) Предпроектный анализ предметной области;
б) Разработка информационной структуры приложения;
в) Разработка мобильного приложения;
г) Тестирование мобильного приложения;
д) Вопросы безопасности жизнедеятельности и охраны труда;
е) экономическая эффективность работ по стандартизации.

Перечень графического материала (с точным указанием обязательных чертежей) 58 иллюстраций и 13 таблиц.

Основная рекомендуемая литература:

1. Абдимуратов Ж. С., Мананбаева С. Е. Безопасность жизнедеятельности. Методические указания к выполнению раздела «Расчет производственного освещения» в выпускных работах для всех специальностей. – Алматы: АУЭС, 2013. – 20 с.

2. Бекишева А. И. Методические указания к выполнению экономической части дипломной работы для бакалавров специальности 5В0703 – Информационные системы – Алматы: АУЭС; 2013. – 24 с.

3. Java. Библиотека профессионала, том 1.10-е издание. : Пер. с англ. – М. : ООО «И. Д. Вильямс», 2016. – 864с.

Консультации по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант	Сроки	Подпись
Экономическая часть	Бекишева А.И.	26.02-30.04.19	
Безопасности жизнедеятельности	Бекбасаров Ш.Ш	26.02–02.04.19	
Программная часть	Майкотов М. Н.	1.04–6.05.19	
Нормоконтролер	Алимсеитова Ж. К.	1.04–8.05.19	

График
подготовки дипломной работы (проекта)

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
1. Теоретические основы разработки ПП	01.01. – 01.02.2019	Ж. Бидахмет
2. Разработка приложение	01.02 –01.03.2019	Ж. Бидахмет
3. Вопросы БЖД	01.03 – 14.03.2019	Ж. Бидахмет
4. Экономическая эффективность ПП	14.03 – 30.04.2019	Ж. Бидахмет

Дата выдачи задания «14» января 2019 г.

Заведующий кафедрой _____ Т.С.Картбаев

Научный руководитель работы Ж. Бидахмет Ж.Бидахмет

Задание принял к исполнению студент _____ А.Я. Заиров

Аңдатпа

Бұл тезис жобасы логистикалық компания үшін тауарлар мен жүк жөнелтушілерді тиімді іздестіруді қамтамасыз ететін мобильдік қосымшаны әзірлеуге арналған.

Бұл қосымша барлық қажетті ақпаратты заманауи гаджеттердің ыңғайлы құралымен алуға мүмкіндік береді. Онда пайдаланушы қол жетімді жүк немесе көліктік қызмет көрсететін көлік хабарландыруларды ғана көре алмайды, сонымен қатар оларды жасай алады.

Бағдарламалық жасақтама Android Studio бағдарламасымен, нысанға бағытталған Java бағдарламалау тілін қолдану арқылы жүзеге асырылады. Үшінші бөлімінде мобильді қосымшасы толығымен сипатталды. Ал деректер базасы Firebase Database арқылы іске асады.

Аннотация

Данный дипломный проект посвящен разработке мобильного приложения для логистической компании, обеспечивающий эффективный поиск грузов и грузоотправителей. Это приложение позволит получать всю необходимую информацию в удобной форме для пользователя с помощью современных устройств. В нем пользователь может не только просматривать доступные объявления о грузах или транспортах, поставляющие услуги по транспортировке, но и создавать их.

Программная часть реализована с помощью Android Studio, объектно-ориентированный язык программирования Java. В третьем разделе было полностью описано разработанное мобильное приложение. В качестве базы данных использовался Firebase Database.

Annotation

This thesis project is devoted to the development of a mobile application for a logistics company, providing an effective search for goods and shippers. This application will allow you to get all the necessary information in a convenient form for the user using modern devices. In it, the user can not only view the available cargo or transport announcements that provide transportation services, but also create them.

The software is implemented using Android Studio, an object-oriented Java programming language. In the third section, the developed mobile application was fully described. Firebase Database was used as a database.

Содержание

Введение

В реалиях нынешнего мира без автомобильного транспорта невозможна деятельность ни одной отрасли хозяйства: обеспечение работоспособности промышленных и торговых предприятий, сельского хозяйства, жизнедеятельности городов и районов страны, социальной сферы, деловых и культурных потребностей населения. Почти все трудящийся страны работает в настоящее время на автомобильном транспорте: диспетчеры и экономисты, водители и ремонтники, руководители и менеджеры.

Главным типом услуг предприятия, предоставляющие транспортные услуги – это транспортировка грузов на ближние и дальние (свыше 50 км) расстояния. Перевозка особых видов грузов требует обеспечения особых условий их сохранности, например, химически активные вещества, сыпучие грузы, жидкости, необходимость поддержания определенного температурного режима; и доставки, например, крупногабаритные и тяжеловесные грузы.

Автомобильные грузоперевозки – одна из самых важнейших отраслей народного хозяйства, развивается как неделимая часть единой транспортной системы. В нынешних условиях дальнейшее развитие экономики невозможно без хорошо налаженного автотранспортного обеспечения. От его чёткости и надёжности во многом зависят трудовой ритм предприятий промышленности, строительства и сельского хозяйства. Он гарантирует наряду с другими видами транспорта рациональное производство и обращение продукции промышленности и сельского хозяйства, удовлетворяет потребности населения в перевозках.

Производственное значение транспорта определяется объективной потребностью перемещения груза от места производства к месту потребления.

Одним из главных задач транспорта является своевременное, качественное и полное удовлетворение потребностей народного хозяйства и населения в автоперевозках. Эффективный поиск грузов для перевозки позволяет увеличить объем перевозок, тем самым увеличив производительность автомобильного транспорта.

Исходя из выше сказанного, разработка мобильного приложения для эффективного поиска грузов для перевозок с помощью автомобильного транспорта является весьма актуальным, тем самым подтверждая необходимость темы моей дипломной работы.

Тема моей дипломной работы: «Разработка мобильного приложения для логистической компании».

На сегодняшний день в Казахстане существует несколько аналогов предлагаемого в данной дипломной работе приложения, однако многие из них обладают необщительным интерфейсом, а также многие аналогичные мобильные приложения обладаю несколькими посредниками, то есть, нет гарантий, что заявки, не прошли через несколько рук автодиспетчеров.

1 Анализ современного состояния вопроса

Автомобильные перевозки широко используется во всех областях экономики, – в торговле, производстве, сельском хозяйстве. Процесс транспортной логистики тесно взаимосвязан со многими отраслями и своевременная доставка грузов очень важна для обеспечения их функционирования. Эффективный поиск грузов для перевозки позволяет увеличить объем перевозок, осуществляемых транспортными предприятиями, снизить их издержки, повысить производительность водителей. Такой эффективный поиск грузов для перевозок транспортными средствами может быть реализован с помощью мобильного приложения, позволяющей сохранять:

– информацию о водителях, их стаже работы;
– информацию о грузах, их стоимость;
– информацию о транспортном средстве.

На сегодняшний день все больше возрастает популярность мобильных приложений, они широко используются в любой сфере современной жизни, начиная от развлечений и заканчивая образованием. Каждый день миллионы людей устанавливают и используют мобильные приложения. Согласно исследовательским данным, количество приложений которые люди используют на смартфонах и планшетах в течение месяца, в среднем составляет 33-39. Более подробные данные отражены на рисунке 1.

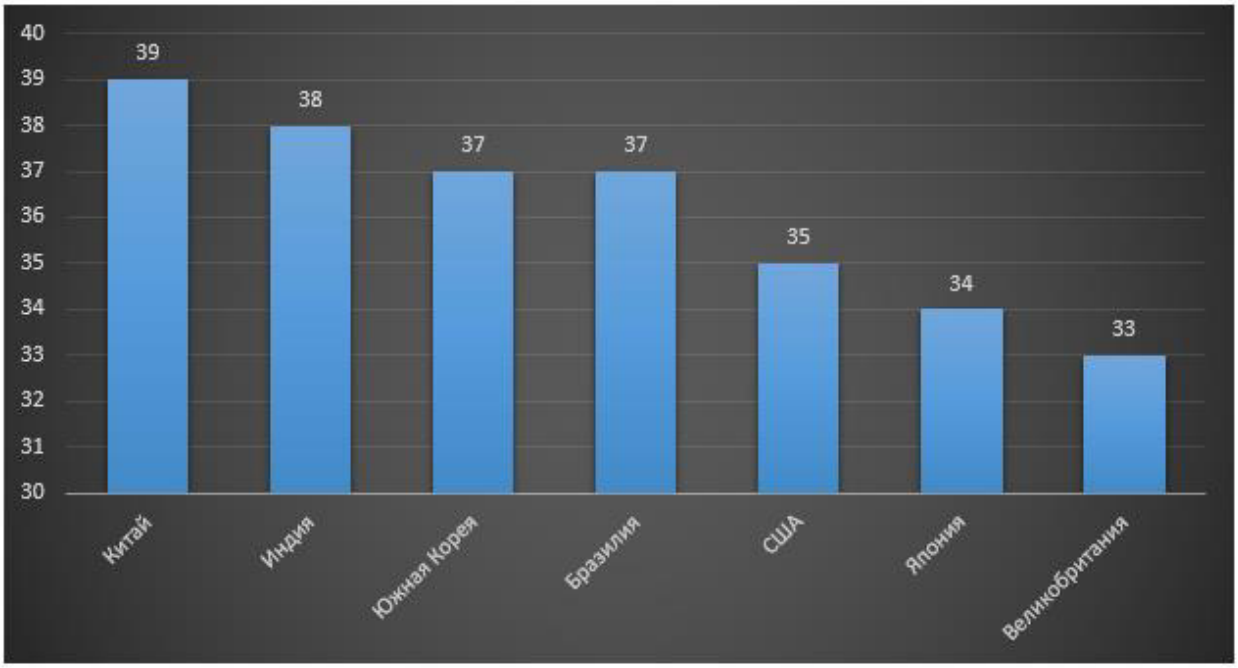

Рисунок 1 – Среднее количество ежемесячно используемых приложений

На протяжении многих лет популярными мобильными операционными системами являются Android компании Google, iOS – Apple и Windows Phone компании Microsoft. Лидирующую позицию среди данных систем занимает ОС Android. По данным на декабрь 2015 года она стала лидирующей не только среди мобильных ОС, но обогнала по суммарному количеству интернет-пользователей, использующих ОС семейства Windows (включая компьютерные операционные системы). На рисунке 2 видно, что по результатам трех месяцев лидирует Android (41.3% среднесуточных посетителей). Строка «сумма выбранных» показывает сумму долей всех пользователей платформами от Microsoft и составляет 41% в среднем за 3 месяца.

отчет: количество посетителей с разными ОС — по дням | по неделям | по месяцам

значения: среднесуточные	декабрь 2015 г.		ноябрь 2015 г.		в среднем за 3 месяца	
Android	70,629,069	42.0%	68,212,908	41.4%	67,708,902	41.3%
Windows 7	40,780,603	24.2%	40,715,944	24.7%	40,562,596	24.7%
iOS iPhone	18,422,030	11.0%	18,194,590	11.0%	18,160,447	11.1%
Windows 8	10,031,547	6.0%	10,162,611	6.2%	10,029,082	6.1%
Windows XP	9,426,365	5.6%	9,437,389	5.7%	9,481,850	5.8%
Windows 10	5,121,773	3.0%	4,246,066	2.6%	4,225,589	2.6%
iOS iPad	4,663,544	2.8%	4,734,672	2.9%	4,702,975	2.9%
Windows Phone	2,225,858	1.3%	2,167,833	1.3%	2,162,788	1.3%
SymbianOS	2,167,195	1.3%	2,239,217	1.4%	2,239,536	1.4%
MacOS	1,444,612	0.9%	1,428,942	0.9%	1,429,924	0.9%
Unix	1,324,892	0.8%	1,307,054	0.8%	1,303,343	0.8%
Другие	1,222,486	0.7%	1,244,661	0.8%	1,247,516	0.8%
Windows Vista	611,150	0.4%	628,826	0.4%	626,309	0.4%
Windows 2003	86,012	0.1%	87,377	0.1%	87,302	0.1%
Windows	52,430	0.0%	55,267	0.0%	53,225	0.0%
Не определена	5,783	0.0%	30,096	0.0%	14,913	0.0%
сумма выбранных	68,335,741	40.6%	67,501,317	40.9%	67,228,745	41.0%
всего	168,215,354		164,893,461		164,036,306	

Рисунок 2 – Статистика пользователей, выходящих в сеть с различных ОС

Исходя из приведенных данных и статистики, можно сделать вывод, что мобильные приложения, написанные для работы на устройствах с операционной системой Android, на сегодняшний день необычайно популярны, а, следовательно, и разработка таких приложений является достаточно актуальной.

Ниже будут рассмотрены некоторые продукты-аналоги с магазина приложений «Google Play».

1.1 «АвтоТрансИнфо»

Система “АвтоТрансИнфо” (АТИ сайт грузоперевозок) – самая популярная база машин и грузов. Есть расчет расстояний и рейтинг участников. Работает минимальный необходимый для работы с грузами функционал: поиск и добавление грузов, отправка предложений, просмотр информации по фирме. На рисунке 3 представлен интерфейс приложения.

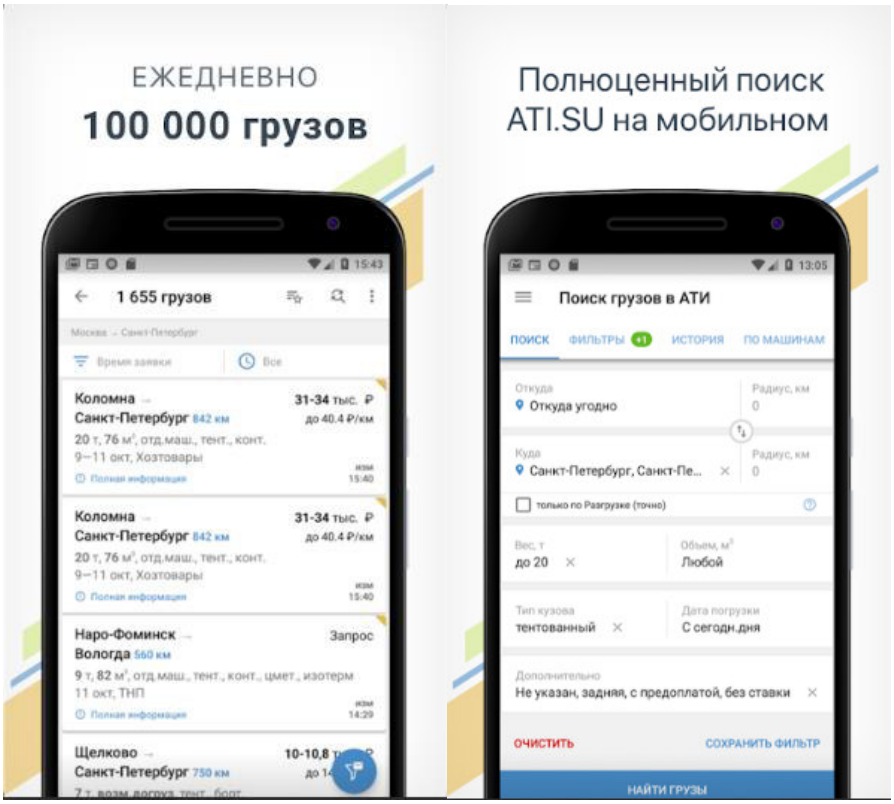

Рисунок 3 – АвтоТрансИнфо

Хочу выделить некоторые плюсы:

– страхование груза;
– поиск машин;
– рейтинги перевозчиков и владельцев груза.

Несмотря, на достаточно большие плюсы, существуют и недостатки. Такие как:

– очень много различных посредников. Т.е. нет гарантий, что заявки, не прошли через несколько рук автодиспетчеров. Это грозит тем, что перевозчику остаются копейки, остальную большую часть забирают посредники;

– достаточно открытая система. Эта система стала разгулом криминала. Т.е. грузы уводятся перевозчиками, а заказчики, наоборот, не платят перевозчикам за выполненную работу.

1.2 «Della автоперевозки»

Этот транспортный портал открылся в 1995 году. В начале своей истории он выглядел как обычная веб-форма, которую заказчики и перевозчики использовали для введения сведений о наличии транспорта и грузов. После определенной обработки информация о свободной продукции и транспорте рассылалась на электронную почту пользователей. Прошедшее время внесло множество немаловажных и значительных изменений во внешнее оформление и работу портала DELLA грузоперевозки.

Транспортный сайт может похвастаться отсутствием перезагруженности страниц, свободным дизайном в минималистическом стиле, приятным оформлением, выполненным в «пастельных» тонах, и наличием транспортных таблиц упрощающих поиск необходимой информации. Главная страница сайта указана на рисунке 4.

Рисунок 4 – Главная страница «Della автоперевозки»

Хочу выделить некоторые плюсы:

– информационный сайт ДЕЛЛА грузоперевозки является одним из лидеров среди отечественных порталов, предоставляющих подобного рода услуги. По своим качествам он практически не уступает известным международным транспортным порталам;

– работа портала практически не дает сбоев, для нее не важен вид интернета, программного обеспечения или браузера компьютера;

– поиск машин;

– в случае изменения контактной информации, она проходит определенную проверку, причем, иногда, представители компании запрашивают у клиента подтверждения новых данных.

Как бы организаторы портала не стремились создать безупречный коммерческий продукт, кое-какие недочеты есть и у него. Ниже будут рассмотрены некоторые из них:

– некоторые ставят под сомнение полезность определенных дополнительных функций. К примеру, раздел «Цена на перевозки» предлагает подсчет стоимости транспортировки 1 тонны продукции на автотранспорте, имеющем различную грузоподъемность, а также стоимость доставки груза за 1 км. по различным направлениям. Результатом подобных расчетов можно воспользоваться лишь для сравнения размещенных на сайте предложений. Это связано с тем, что существует отдельная малотоннажная продукция (массой до 2 т), а также грузы, чья масса не соответствует их объему (например, металл или пенопласт);

– после обновления страницы возврат на предыдущую становится невозможным: пользователя будет «выкидывать» на ту страницу, где он менял условия поиска. В связи с этим, возникает вероятность потери данных «на заметку». Даже, если они не являются слишком важными для пользователей, их исчезновение вызывает определенный дискомфорт;

– ограниченный алгоритм формы для размещения информации не позволяет свободно излагать дополнительные условия, требуя выбора из прилагаемого списка.

1.3 «Fa-Fa.kz» – Грузоперевозки в Казахстане и СНГ

Сайт-биржа грузов. Здесь вы можете найти груз или перевозчика для доставки вашего груза. Владельцам автотранспорта, желающим постоянно иметь достаточное количество заказов на грузоперевозки по Казахстану, и хозяевам грузов, стремящимся снизить логистические затраты, будет одинаково полезен сайт fa-fa.kz. Здесь регулярно размещаются самые разные предложения по перевозкам. С помощью сайта грузоперевозчики могут заработать, а грузоотправители – сэкономить. Поэтому регистрация на сайте одинаково выгодна и тем, и другим.

Преимущества интернет-выбора транспорта и грузов для грузоперевозок по Казахстану:

– на интернет-ресурсе легко подобрать не только грузовик нужной грузоподъемности, но и рефрижератор, плитовоз, длинномерный транспорт, автовоз и другую машину для транспортировки конкретных грузов;

– имеется обширная база заказов и предложений от перевозчиков с номерами контактных телефонов;

– ресурс позволяет компаниям, отправляющим грузы, экономить на грузоперевозках по Казахстану до 50-70% средств;

– на сайте находится разнообразная информация – допустимые размеры транспортных средств, сведения о закрытых дорогах, типовые договора на предоставление услуг. Клиенты могут воспользоваться помощью диспетчеров, которые подберут подходящий груз, рассчитать нагрузки на ось, стоимость перевозки, расстояние между населенными пунктами и даже пообщаться на форуме;

– регистрация на ресурсе проста и не вызывает сложностей. Достаточно указать электронную почту, номер телефона и свой город.

Несмотря, на достаточно большие плюсы, существуют и недостатки. Такие как:

– дизайн немного староват;

– нету автоматического обновления для страницы со списком грузов, необходимо нажать самому «Обновить»;

– недостаточно информации по машине: нет ADR, не описаны подробно характеристики машины.

Главная страница сайта указана на рисунке 5.

Рисунок 5 – Главная страница «fa-fa.kz»

1.4 «20тонн для грузоперевозок»

20tonn.kz – это система, где вам достаточно зарегистрироваться, поместить заявку и получать предложения от проверенных перевозчиков, выбирая лучшего по цене и рейтингу. В процессе перевозки вы сможете получать слежение за маршрутом, а после – полный пакет документов и удобные способы оплаты.

Хочу отметить некоторые плюсы:

– поиска груза по направлениям;
– поиск груза по областям;
– размещение грузов в приложении или с сайта 20tonn.kz;
– поиск груза по радиусу от указанного населенного пункта;
– настройка уведомлений по новым заявкам на грузоперевозку.

На рисунке 6-7 приведены скриншоты приложения и веб-сайта.

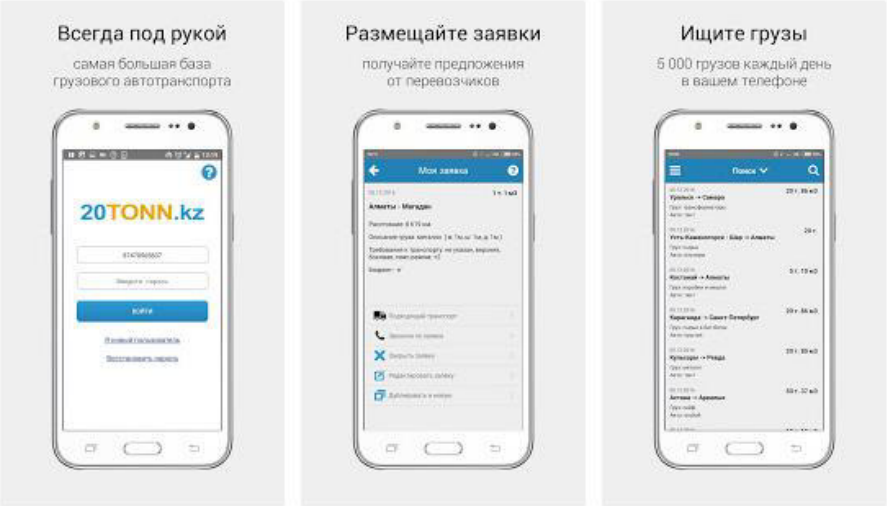

Рисунок 6 – «20тонн для грузоперевозок»

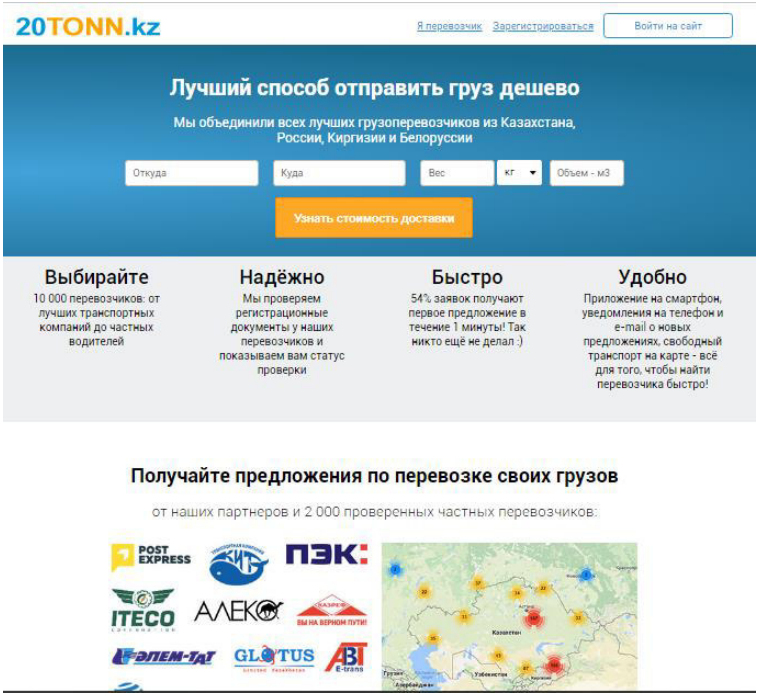

Рисунок 7 – Сайт «20тонн»

1.5 Постановка задачи дипломной работы

Необходимо создать мобильное приложение с помощью среды разработки «Android Studio», где будет использоваться операционная система Android. Приложение должно будет иметь возможность вести поиск по ключевым словам и подача объявления путем его добавления в базу данных, которая находится в облаке.

Прежде, чем приступить к разработке мобильного приложения, необходимо найти уже существующие аналоги, провести их анализ и выявить их плюсы и минусы, чтобы учесть их при разработке собственного приложения.

База данных, которая будет привязана к приложению, должна иметь определенные требования для его полноценной работы:

- ввод данных
- вывод данных
- стабильность
- гибкость
- умение обрабатывать большой объем данных

Блок о грузах будет содержать следующую информацию:

- вид груза;
- тип погрузки;
- вес;
- цена;
- пункт назначения;
- пункт отправки;
- дата погрузки;
- тип кузова;
- оплата;
- и т. д.

Требования к программному обеспечению

Приложение должно будет реализовано под операционную систему Android.

2 Разработка информационного обеспечения системы

2.1 Информационная структура приложения

Проектируемое приложение задумывается как Android - приложение. В ходе изучения предметной области были выявлены достоинства и актуальность использования операционной системы Android.

Поскольку основная часть Android - приложений построена с использованием Java-машины, то в качестве языка программирования выбран язык Java.

При реализации данного мобильного приложения можно выделить несколько основных задач:

– разработка информационных блоков;
– разработка графического интерфейса пользователя;
– разработка блока, содержащего информацию об имеющихся грузах.

При проектировании приложения данной тематики, была разработана функциональная структура систем, которая показывает список задач, которые будут выполнять следующие подсистемы:

– подсистема «Грузоперевозчик»;
– подсистема «Грузоотправитель»;
– подсистема «Гость».

Сама функциональная структура показана на рисунке 8.

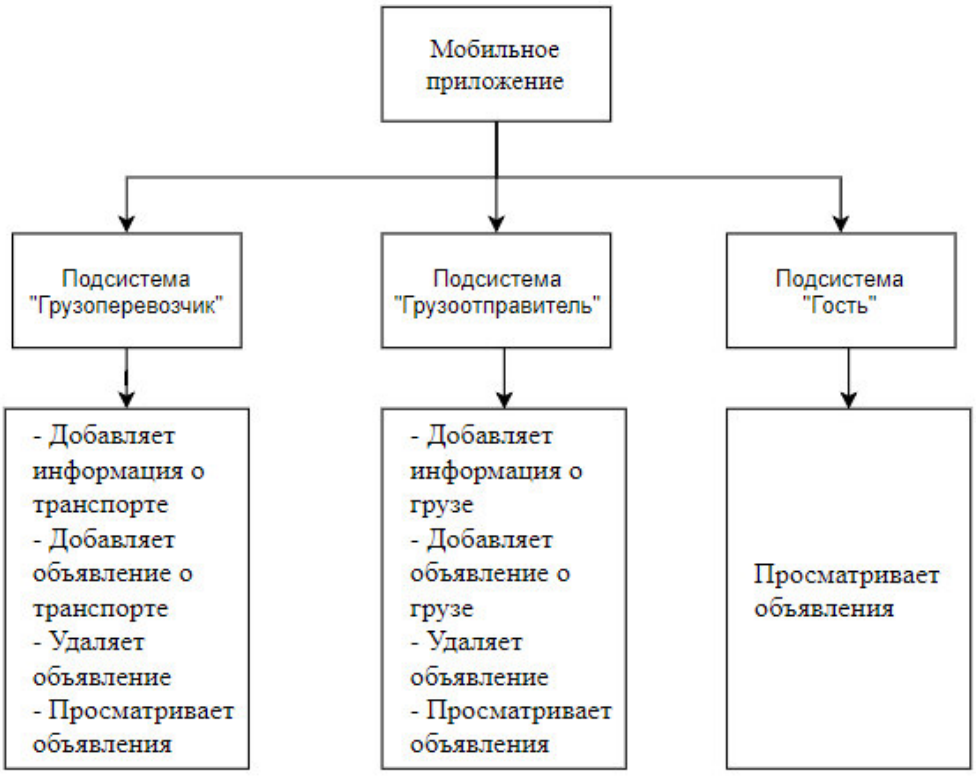

Рисунок 8 – Функциональная струтура приложения

2.2 Обоснование выбора СУБД

На сегодняшний день имеются очень много видов СУБД, но мало из них имеют такие же качества как у FireBase. Мой выбор СУБД пал именно на FireBase так у него очень много достоинств и гибкости в использовании. FireBase — американская компания, поставщик облачных услуг, основанная в 2011 году Эндрю Ли и Джеймсом Тэмплином, и поглощённая в 2014 году корпорацией Google.

Основной сервис — облачная СУБД класса NoSQL, позволяющая разработчикам приложений хранить и синхронизировать данные между несколькими клиентами. Поддержаны особенности интеграции с приложениями под операционные системы **Android** и **iOS**, реализовано API для приложений на JavaScript, Java, Objective-C и Node.js, также возможно работать напрямую с базой данных в стиле REST из ряда JavaScript-фреймворков, включая AngularJS, React, Ember.js и Backbone.js. Предусмотрено API для шифрования данных.

Среди других услуг, предоставлявшихся компанией — запущенный 13 мая 2014 года хостинг для хранения статических файлов (таких как CSS, HTML, JavaScript), обеспечивающий доставку через CDN и сервис аутентификации клиента с использованием кода только на стороне клиента с поддержкой входа через Facebook, GitHub, Twitter и Google (Firebase Simple Login).

Кроме этого, под лицензией MIT компанией выпущен веб-редактор кода Firepad, обеспечивающий одновременную совместную работу нескольким пользователям с одним документом, который стал основой редакторов Stash Realtime Editor фирмы Atlassian и Koding. Ещё один свободный проект компании — бесплатный мессенджер Firechat, также выпущенный под лицензией MIT.

Основные достоинства СУБД FireBase:

- синхронизация в реальном времени (рисунок 9);
- совместно работать с устройствами (рисунок 10);
- создание без-серверных приложений (рисунок 11);
- оптимизирован для автономного использования (рисунок 12);
- высокий уровень безопасности (рисунок 13).

Рисунок 9 – Синхронизация в реальном времени для данных JSON «Инфографика синхронизаций в реальном »

База данных Firebase Realtime представляет собой облачную базу данных NoSQL, которая позволяет хранить и синхронизировать данные между вашими пользователями в режиме реального времени.

К основным функциям Cloud Firestore относятся:

– документы и коллекции с мощным запросом;

– iOS, Android и Web SDK с автономным доступом к данным;

– синхронизация данных в режиме реального времени;

– автоматическая многополосная репликация данных с сильной консистенцией;

– SDK Node, Python, Go и Java.

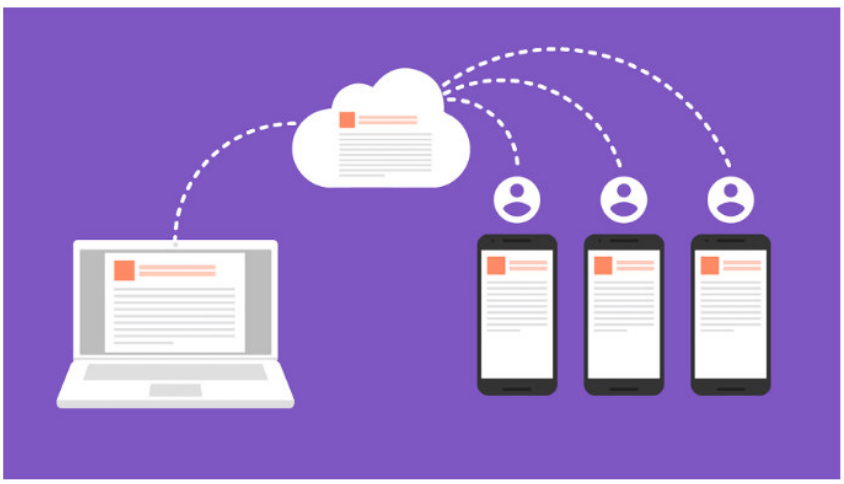

Рисунок 10 – Совместная работа с различными устройствами «Инфографика работы с различными устройствами »

Синхронизация в реальном времени позволяет пользователям получать доступ к своим данным с любого устройства: веб-или мобильным, и это помогает вашим пользователям сотрудничать друг с другом.

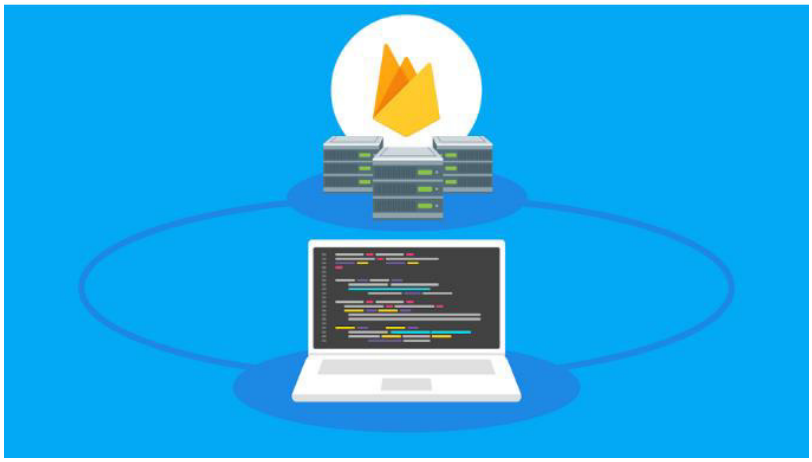

Рисунок 11 – Возможность создавать безсерверные приложении «Инфографика базы данных Realtime с мобильными и веб-SDK»

База данных Realtime поставляется с мобильными и веб-SDK, поэтому вы можете создавать приложения без необходимости серверов. Вы также можете выполнить код бэкэнд, который реагирует на события, инициируемые вашей базой данных, используя Cloud Function for Firebase.

Рисунок 12 – Автономное использование сервиса « Инфографика работы приложения без сети»

Когда пользователи переходят в автономный режим, SDK базы данных Realtime использует локальный кэш на устройстве для обслуживания и

сохранения изменений. Когда устройство подключается к сети, локальные данные автоматически синхронизируются.

Рисунок 13 – Высокий уровень безопасности
« Инфографика аутентификации для пользователей»

База данных Realtime интегрируется с Firebase Authentication для обеспечения простой и интуитивной аутентификации для разработчиков. Мы можем использовать эту декларативную модель безопасности, чтобы разрешить доступ на основе идентификатора пользователя или соответствия шаблонов вашим данным.

Так же имеются дополнения такие как:

– хранение и синхронизированные данные между пользователями и устройствами в реальном времени, используя базу данных NoSQL, размещенную в облаке. Обновленные данные синхронизируются через подключенные устройства в миллисекундах, и данные остаются доступными, если ваше приложение отключается, обеспечивая отличный пользовательский интерфейс, независимо от сетевого подключения.

– экономия времени на устранения неполадок, которая превратит лавину сбоев в управляемый список проблем. Имеется четкое и действенное представление о том, какие проблемы нужно решать в первую очередь, увидев влияние пользователя прямо на панели мониторинга Crashlytics. Предупреждения в реальном времени помогут нам оставаться на вершине стабильности даже в дороге.

– диагностика проблемы в своем мобильном приложении с подробными отчетами об ошибках и сбоях. Приоритет отчетов по частоте и важности воздействия на панели мониторинга Firebase Crash, где мы можем отслеживать общее состояние нашего приложения, а также отслеживать

пользовательские потоки.

– хранить и синхронизировать данные между пользователями и устройствами - в глобальном масштабе - с использованием базы данных noSQL, размещенной в облаке. Cloud Firestore дает нам живую синхронизацию и автономную поддержку наряду с эффективными запросами данных. Его интеграция с другими продуктами Firebase позволяет создавать действительно безсерверные приложения.

– Firebase Auth предлагает несколько методов аутентификации, включая электронную почту и пароль, сторонние поставщики, такие как Google или Facebook, и напрямую используя существующую систему учетных записей

– возможность создавать свое приложение с помощью пользовательского кода, не требуя управления и масштабирования собственных серверов. Функции могут быть вызваны событиями, которые испускаются продуктами Firebase, облачными службами Google или третьими лицами, используя веб-узлы.

– возможность хранить и распространять пользовательский контент, такие как изображения, аудио и видео, с мощным, простым и экономичным хранилищем данных, созданным для масштаба Google. Firebase SDK для Cloud Storage включают в себя безопасность Google для загрузки и загрузки файлов для наших приложений Firebase, независимо от качества сети.

– когда вы загружаете наши веб-ресурсы, мы автоматически выталкиваем их на наш глобальный CDN и предоставляем бесплатный сертификат SSL, чтобы наши будущие пользователи получали безопасную, надежную, скоростную работу независимо от того, где они находятся.

– выполнение автоматического и индивидуального теста для нашего приложения на виртуальных и физических устройствах, размещенных в Google. Имея Firebase Test Lab на протяжении всего жизненного цикла разработки, чтобы обнаружить ошибки и несоответствия.

Это только малая часть того что умеет делать данное СУБД. Его возможности, гибкости при разработки моего приложения будет в полном достатке. За последние 3 года Firebase стала платформой для разработки приложений Google; теперь у него 16 продуктов для создания и развития приложения.

База данных Firebase Realtime с ее клиентскими SDK и возможностями в реальном времени - все это делает процесс разработки приложений более быстрым и легким. С момента его запуска он был принят сотнями тысячами разработчиков, и по мере его принятия, так же как и шаблоны использования. На сегодняшний день разработчики со всего мира начали использовать базу данных Realtime для более сложных данных и для создания более крупных приложений, подталкивая за пределы возможности модели данных и производительность базы данных.

2.3 Обоснование выбора среды разработки

На сегодняшний день имеются очень много видов сред разработки, но мало из них имеют такие же качества как у Android Studio. Мой выбор IDE пал именно на Android Studio так у него очень много достоинств и гибкости в использовании.

Android Studio — интегрированная среда разработки производства Google, с помощью которой разработчикам становятся доступны инструменты для создания приложений на платформе Android OS. Android Studio можно установить на Windows, Mac и Linux. Учетная запись разработчика приложений в Google Play App Store стоит $25. На рисунке 14 – представлен интерфейс среды разработки.

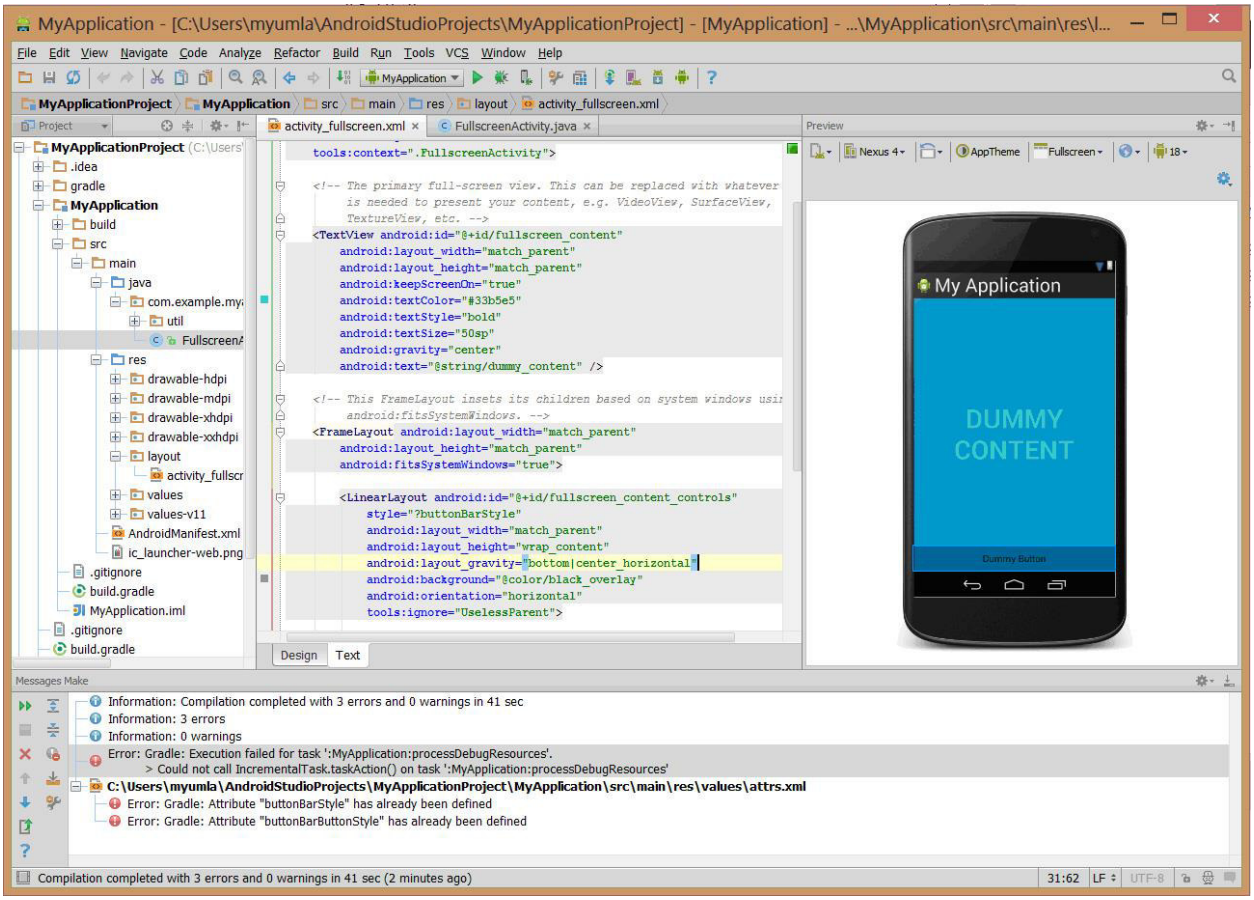

Рисунок 14 – Среда разработки Android Studio

IDE находилась в свободном доступе начиная с версии 0.1, опубликованной в мае 2013, а затем перешла в стадию бета-тестирования, начиная с версии 0.8, которая была выпущена в июне 2014 года. Первая стабильная версия 1.0 была выпущена в декабре 2014 года, тогда же прекратилась поддержка плагина Android Development Tools (ADT) для Eclipse.

Android Studio, основанная на программном обеспечении IntelliJ IDEA от компании JetBrains, – официальное средство разработки Android

приложений. Данная среда разработки доступна для Windows, OS X и Linux.

В основе рабочего процесса Android Studio заложен концепт непрерывной интеграции, позволяющий сразу же обнаруживать имеющиеся проблемы. Продолжительная проверка кода обеспечивает возможность эффективной обратной связи с разработчиками.

С помощью средств оценки производительности определяется состояние файла с пакетом прикладных программ. Визуализация графики дает возможность узнать, соответствует ли приложение ориентиру Google в 16 миллисекунд. С помощью инструмента для визуализации памяти разработчик узнает, когда его приложение будет использовать слишком много оперативной памяти и когда произойдет «сборка мусора». Инструменты для анализа батареи показывают, какая нагрузка приходится на устройство.

К обязательным инструментам относится Android SDK – набор средств программирования, который содержит инструменты, необходимые для создания, компиляции и сборки мобильного приложения.

Рассмотрим кратко наиболее важные инструменты, входящие в состав Android SDK:

– SDK Manager - инструмент, позволяющий загрузить компоненты Android SDK. Показывает пакеты Android SDK и их статус: установлен (Installed), не установлен (Not Installed), доступны обновления (Update available);

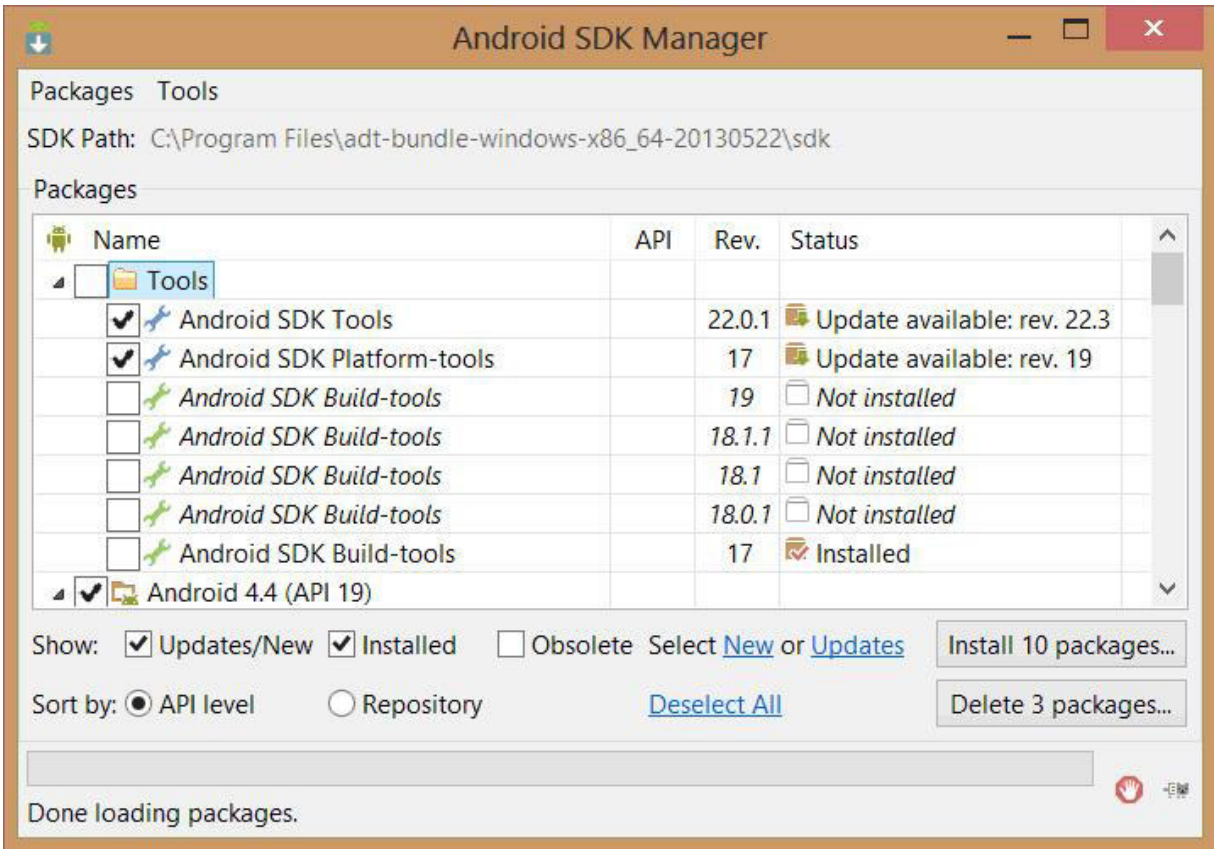

Рисунок 15 – Android SDK Manager

– Android Emulator (emulator) - виртуальное мобильное устройство, которое создается и работает на компьютере разработчика, используется для разработки и тестирования мобильных приложений без привлечения реальных устройств;

– AVD Manager - предоставляет графический интерфейс для создания виртуальных Android устройств (AVDs), предусмотренных Android Emulator, и управления ими.

3 Разработка программного продукта

3.1 Разработка прототипа интерфейса приложения

В ходе разработки интерфейса были учтены все требования предъявленные приложению: удобный интерфейс и наиболее дружелюбный интерфейс. Также были использованы все принципы Material Design, которая была добавлена, начиная с Android 5.0 (уровень API 21). Ниже описан интерфейс всего приложения.

Интерфейс страницы при создании груза, транспорта представлен на рисунке 16.

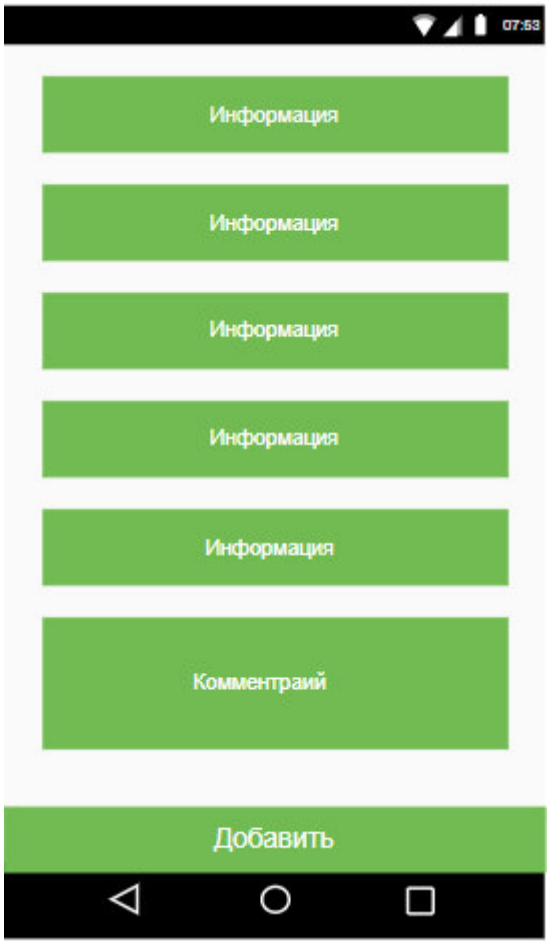

Рисунок 16 – Запись данных о грузе/транспорте в БД

Интерфейс страницы, которая содержит все записи о грузах/транспортах, представлен на рисунке 17.

Второстепенная кнопка в главном меню «Указывает авторство» (Рисунок 19).

Рисунок 17 – Все записи добавленные ранее в БД

Интерфейс страницы при запуске приложения представлен на рисунке 18.

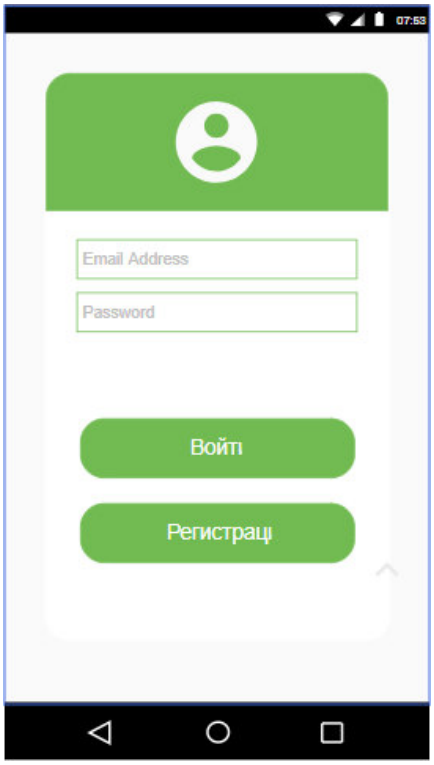

Рисунок 18 – Интерфейс страницы при запуске приложения

Второстепенная кнопка в главном меню «Указывает авторство» (Рисунок 19).

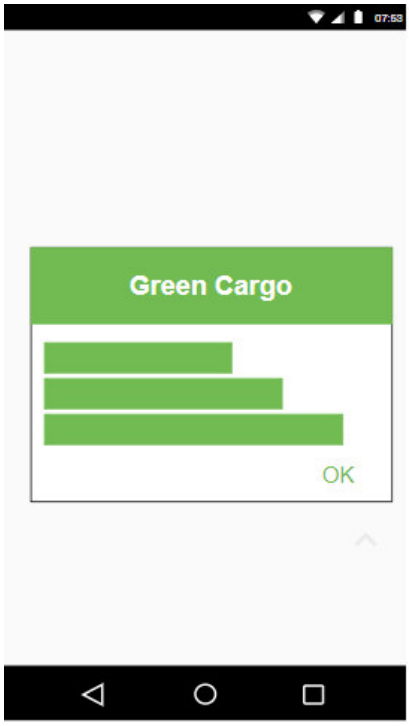

Рисунок 19 – Модальное окно, которое указывает на авторство

Интерфейс навигационной панели, выезжающая сбоку и использующаяся для навигации в приложении представлен на рисунке 20.

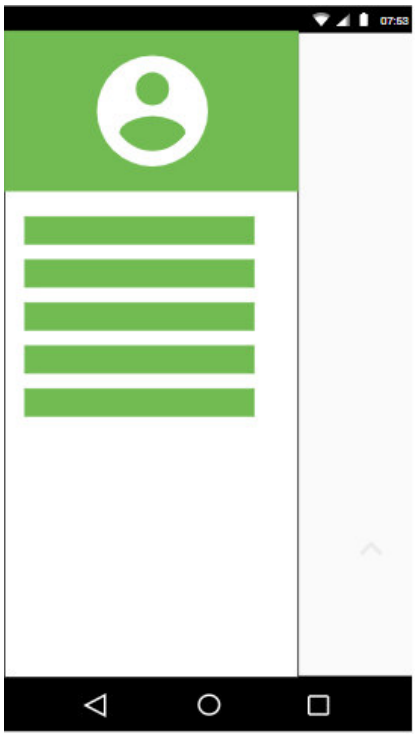

Рисунок 20 – Навигационная панель

Интерфейс страницы, с помощью которого осуществляется поиск грузов/машин, представлен на рисунке 21.

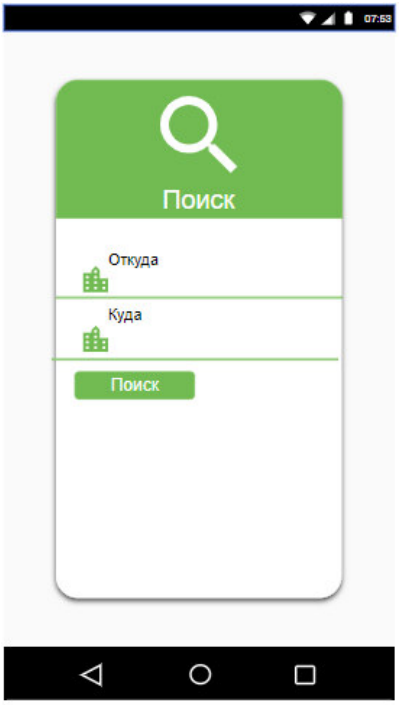

Рисунок 21 – Интерфейс страницы для поиска

Интерфейс экрана, отображающий более подробную информацию о грузе/транспорте, представлен на рисунке 22.

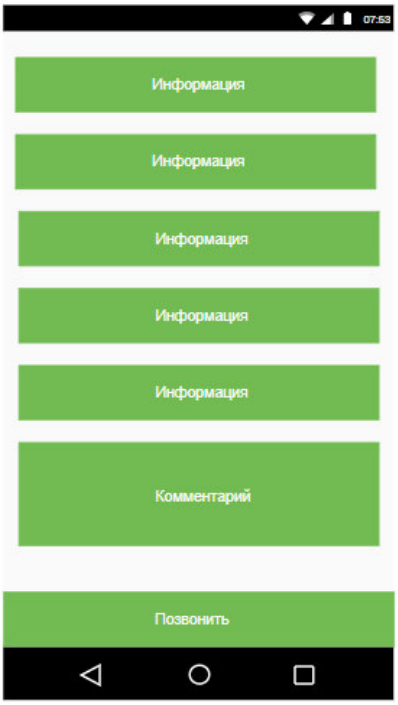

Рисунок 22 –Подробная информация о грузе/транспорте

3.2 Архитектура приложения

Программное приложение для ОС Android состоит из набора активностей, каждой из которых соответствует экран приложения. Каждая активность представлена в проекте классом, реализованном на языке Java, хранящемся в одноименном файле с расширением .java. Каждой активности соответствует xml файл-описание. В xml-файле описано в виде xml-кода расположение визуализируемых объектов. При запуске активности система Android автоматически распознает размер экрана мобильного устройства и приводит выводимый контент в соответствие с разметкой, описанной в xml-файле. Таким образом, одна и та же активность будет выглядеть одинаково независимо от диагонали используемого устройства. Также, для каждого приложения Android должен существовать xml-файл, в котором в виде xml-кода будут прописаны минимальные требования к системе, а также активность, вызываемая при запуске приложения [1].

Разрабатываемое мобильное приложение работает с базой данных Firebase Realtime Database. База данных Firebase Realtime Database представляет собой облачную базу данных NoSQL, которая позволяет хранить и синхронизировать данные между вашими пользователями в режиме реального времени. Firebase Realtime Database является базой данных типа «ключ-значение». Ключ может быть синтактическим или автосгенерированным, а значение может быть представлено строкой, JSON объектом, блобом (BLOB, Binary Large Object, большой двоичный объект) и т.д. Такие базы данных, как правило, используют хеш-таблицу, в которой находится уникальный ключ и указатель на конкретный объект данных. Существует понятие блока (bucket) — логической группы ключей, которые не группируют данные физически. В разных блоках могут быть идентичные ключи.

При таком подходе производительность сильно вырастает за счёт кеширующих механизмов, которые работают на основе маппингов. Чтобы прочитать значение, вам нужно знать как ключ, так и блок, поскольку на самом деле ключ является хешем (блок + ключ).

На рисунке 23 предоставлена пошаговая схема работы реализуемого мобильного приложения, демонстрирующая отображение данных о грузах из базы данных реального времени.

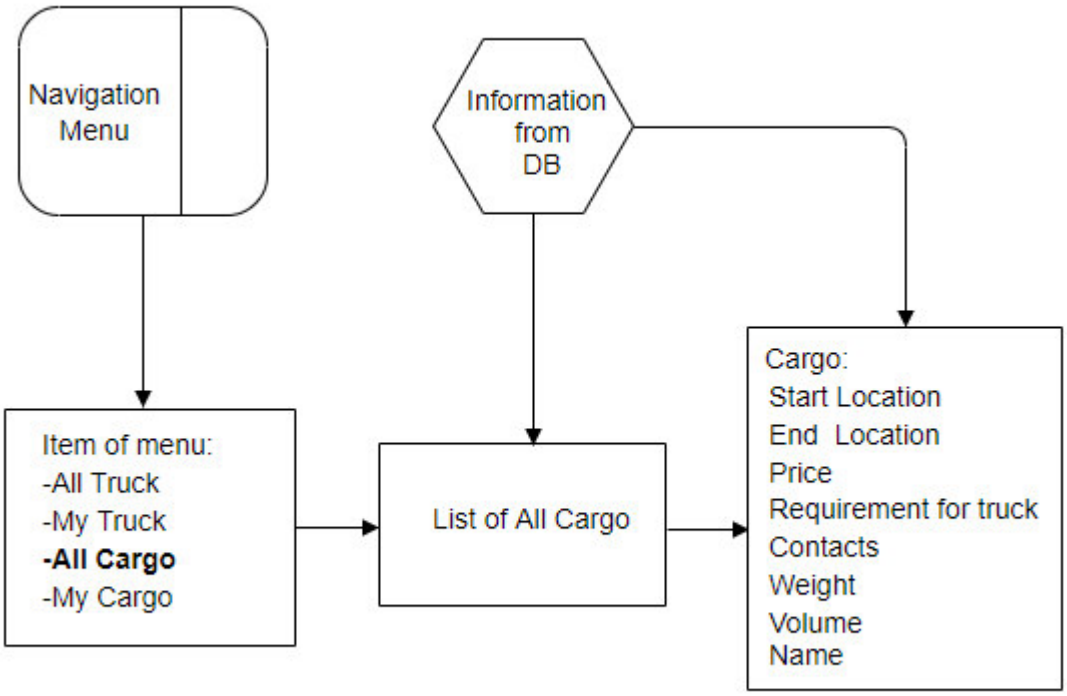

Рисунок 23 – Пошаговая схема работы фрагмента приложения

На рисунке 24 представлена диаграмма потоков данных. Потоки данных в рассматриваемой диаграмме распространяются от базы данных к остальным элементам посредством запросов характерных для Firebase Database.

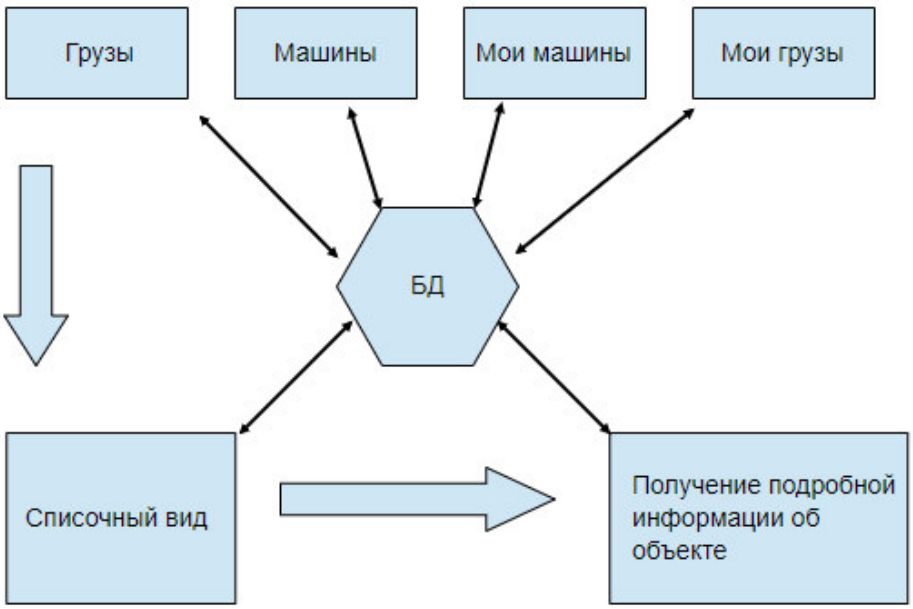

Рисунок 24 – Диаграмма потоков данных

В области разработки приложений используется специальный язык для графического описания объектного моделирования – язык UML. На стадии описания работы приложения для наглядного представления работы отдельных функций приложения приводится Диаграмма компонентов [2]. На рисунках 25 – 32 изображены UML диаграммы вариантов использования (use case diagrams) функционала приложения.

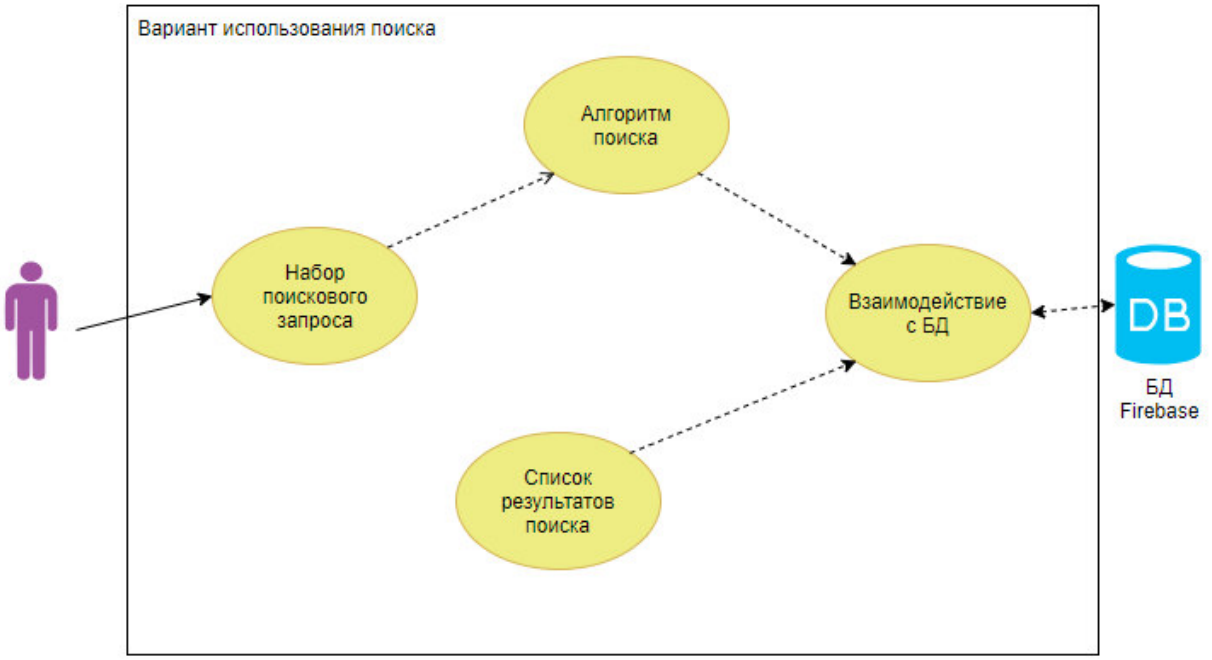

Рисунок 25 – UML-диаграмма экрана поиска

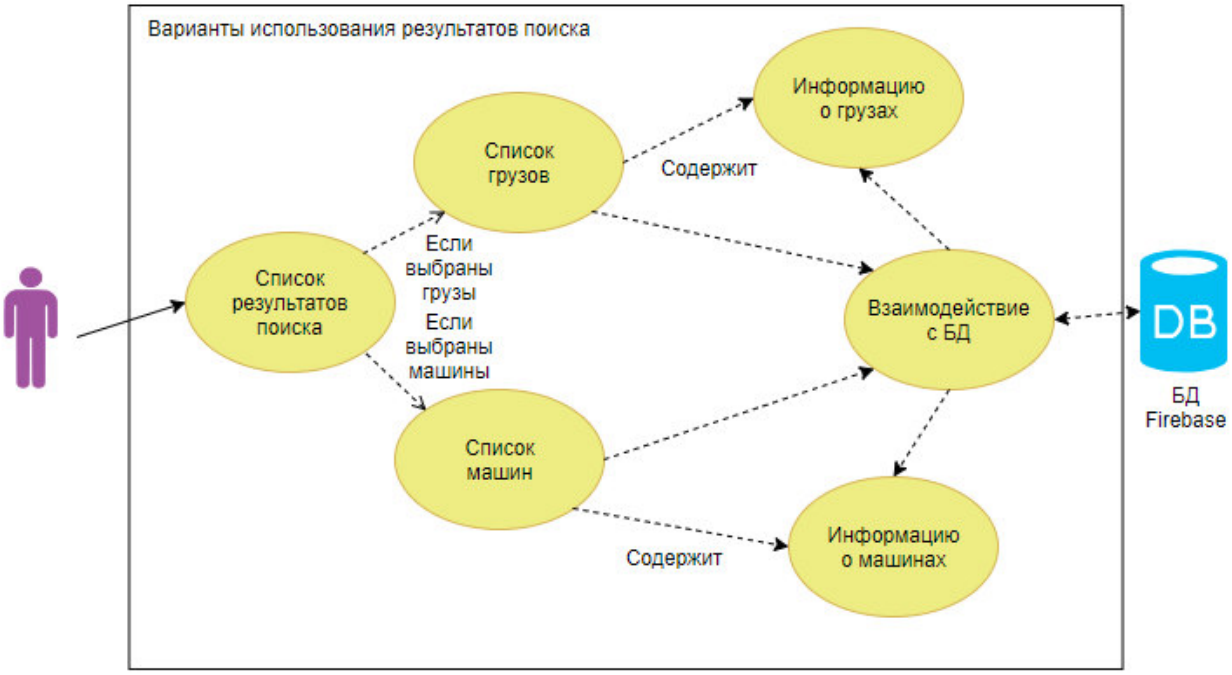

Рисунок 26 – UML-диаграмма вариантов использования результатов поиска

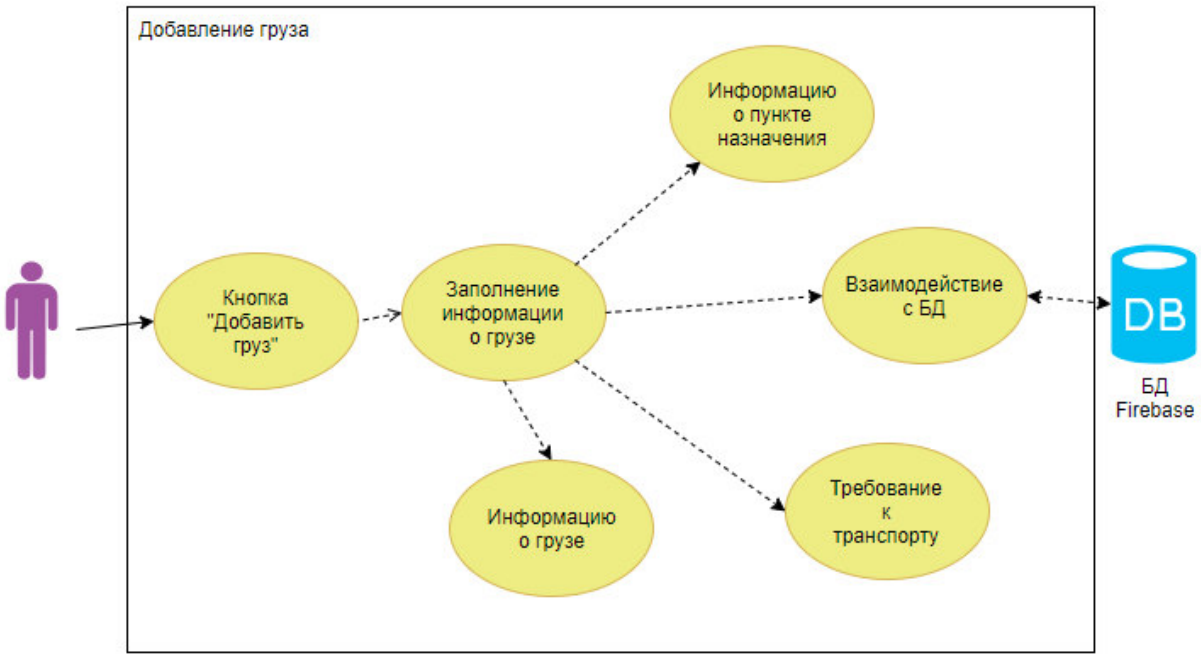

Рисунок 27 – Добавление груза

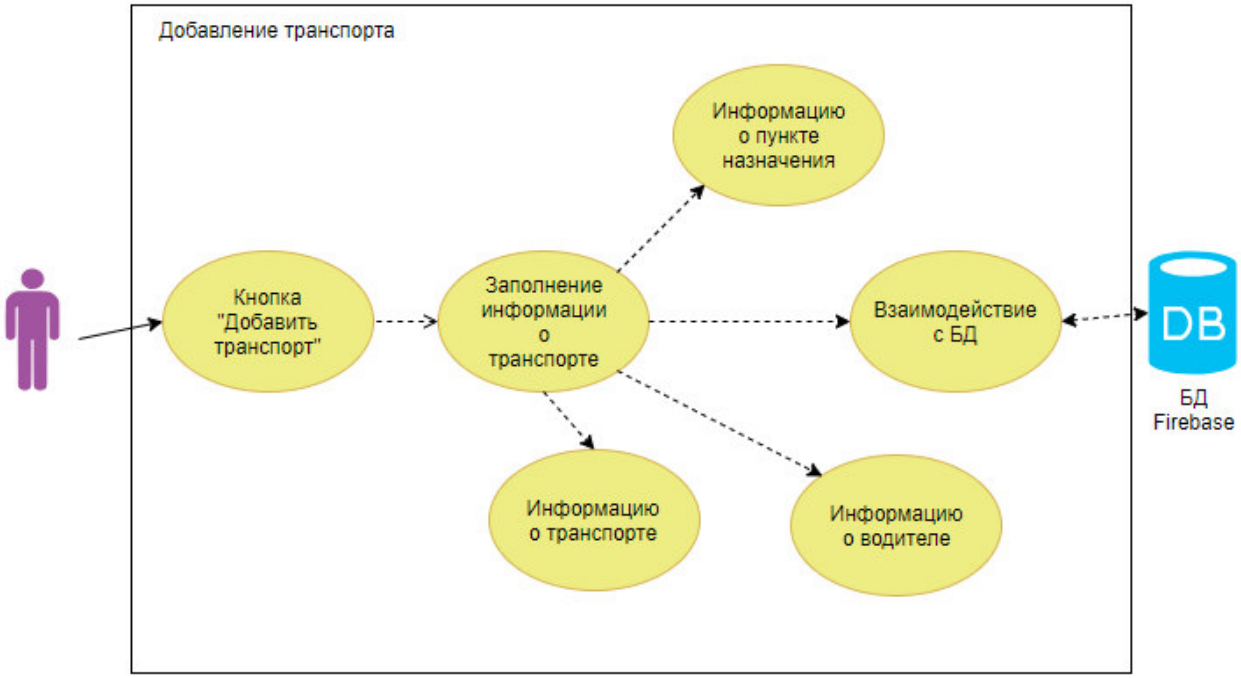

Рисунок 28 – Добавление транспорта

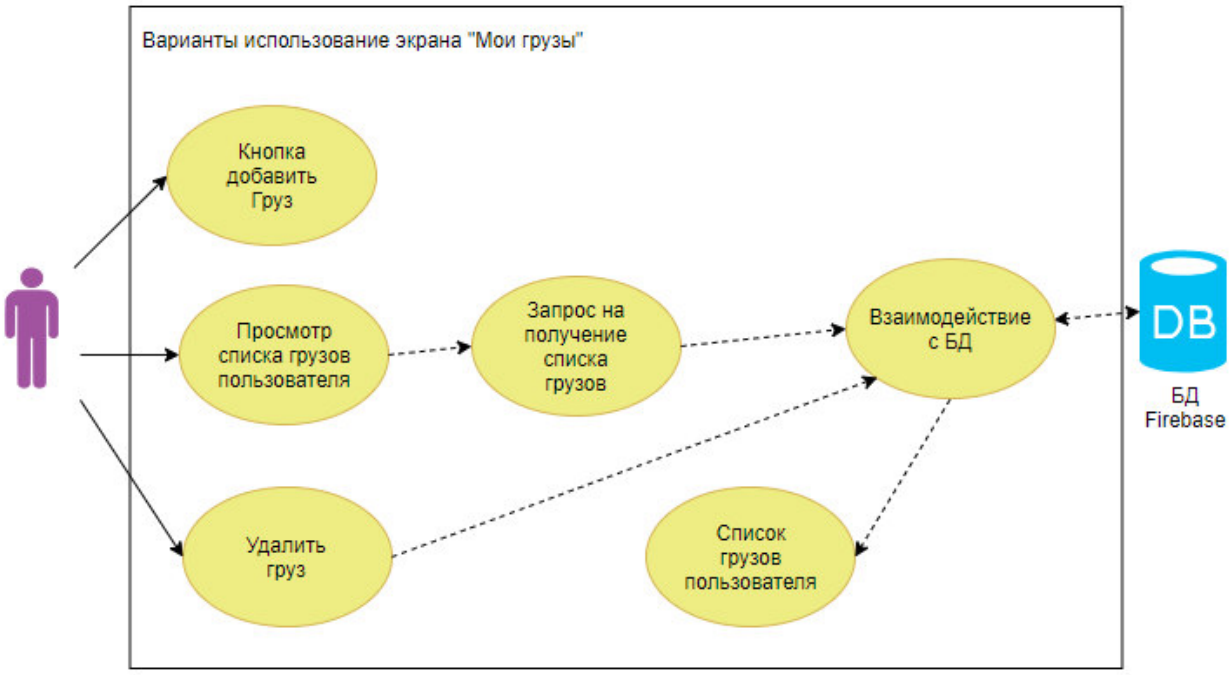

Рисунок 29 – Варианты использования экрана «Мои грузы»

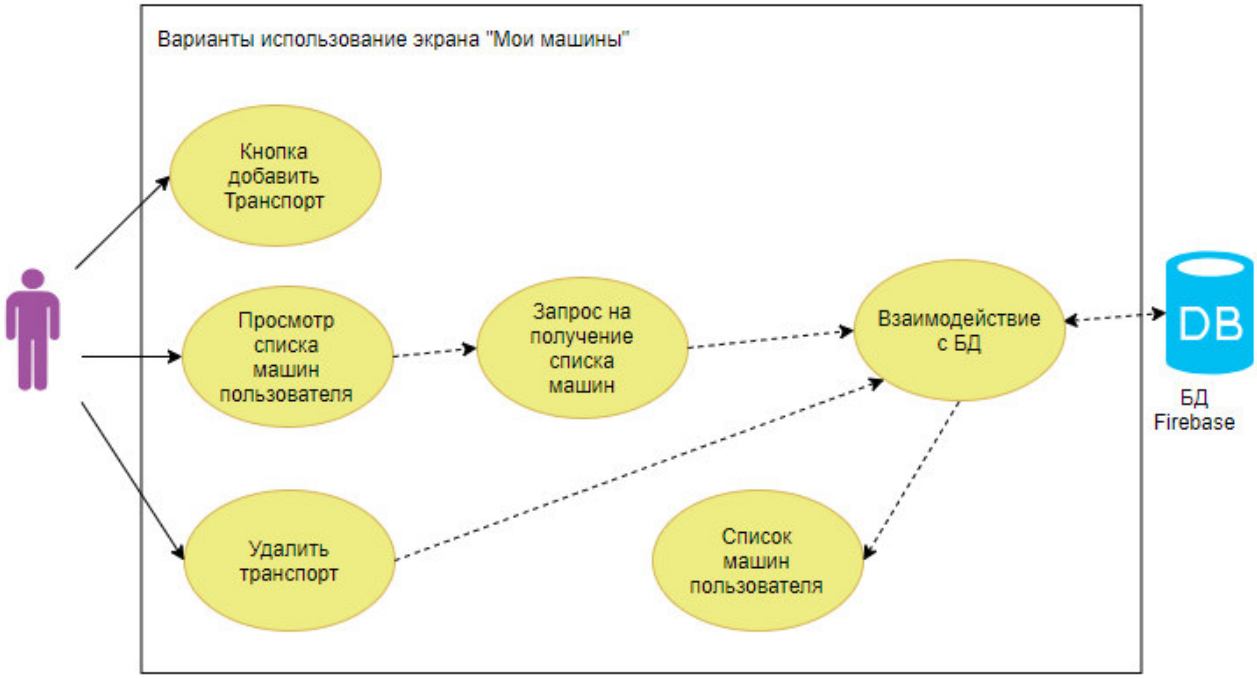

Рисунок 30 – Варианты использования экрана «Мои машины»

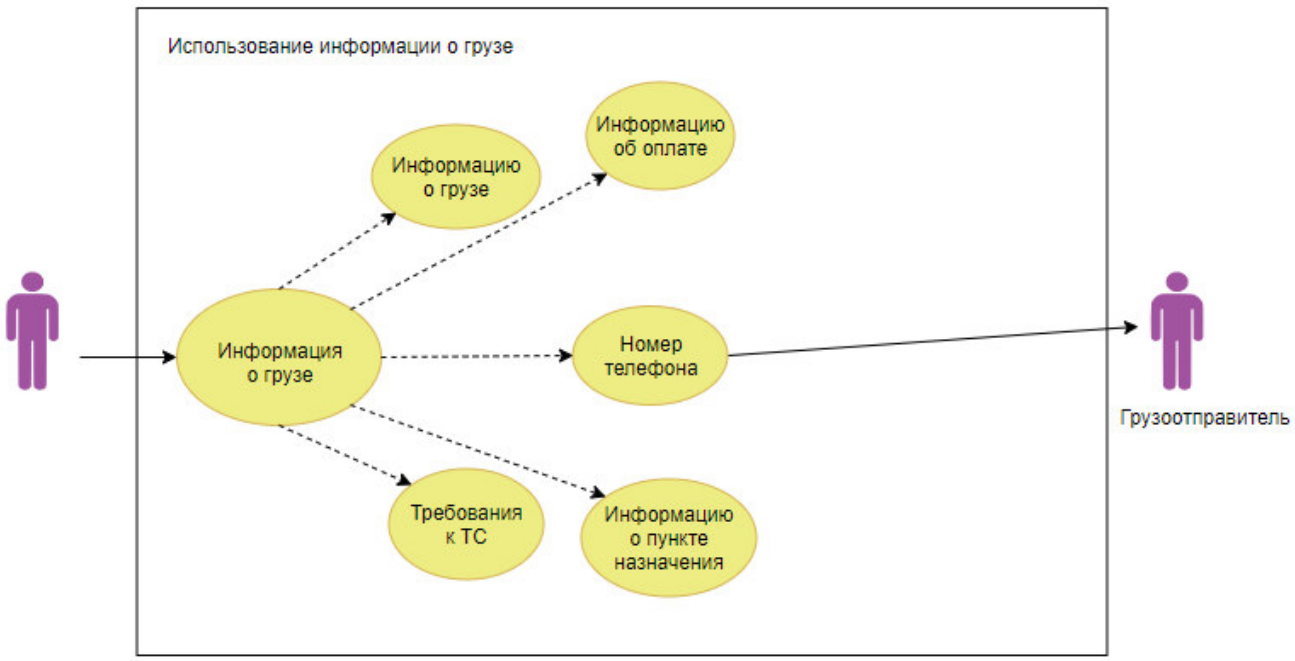

Рисунок 31 – Использование информации о грузе

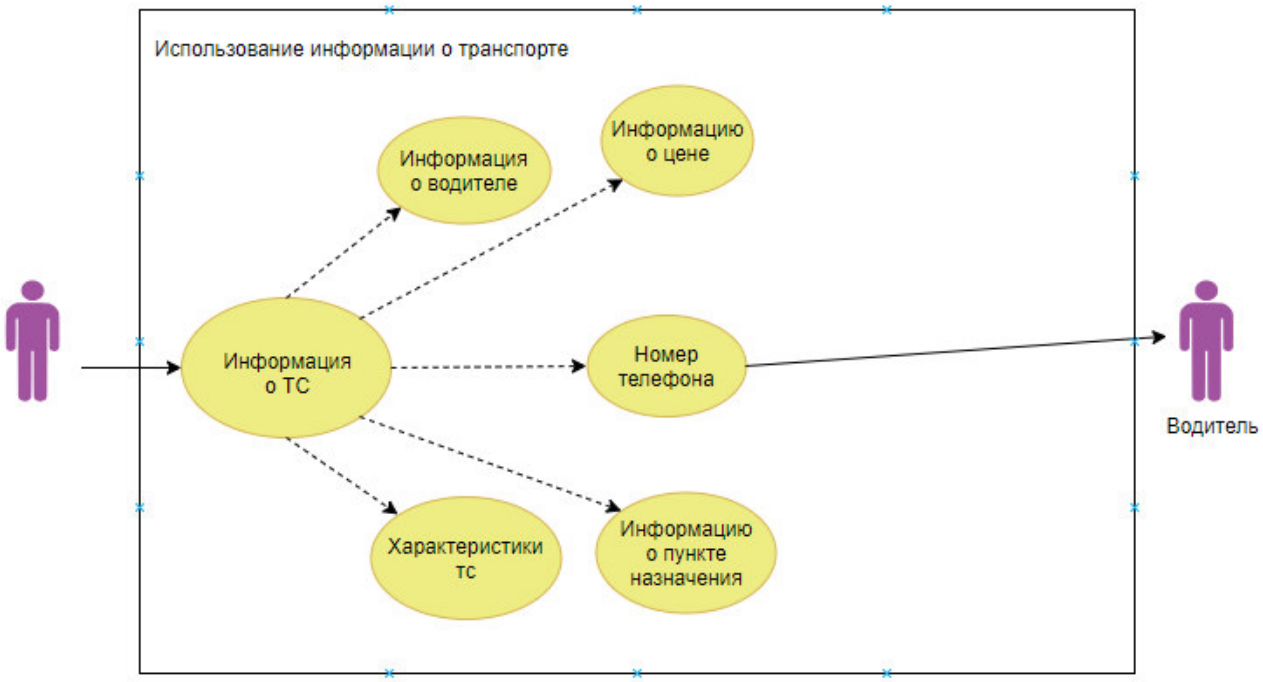

Рисунок 32 – Использование информации о ТС

На рисунке 33 показана диаграмма UML, демонстрирующая пошаговую схему работы всего разрабатываемого приложения.

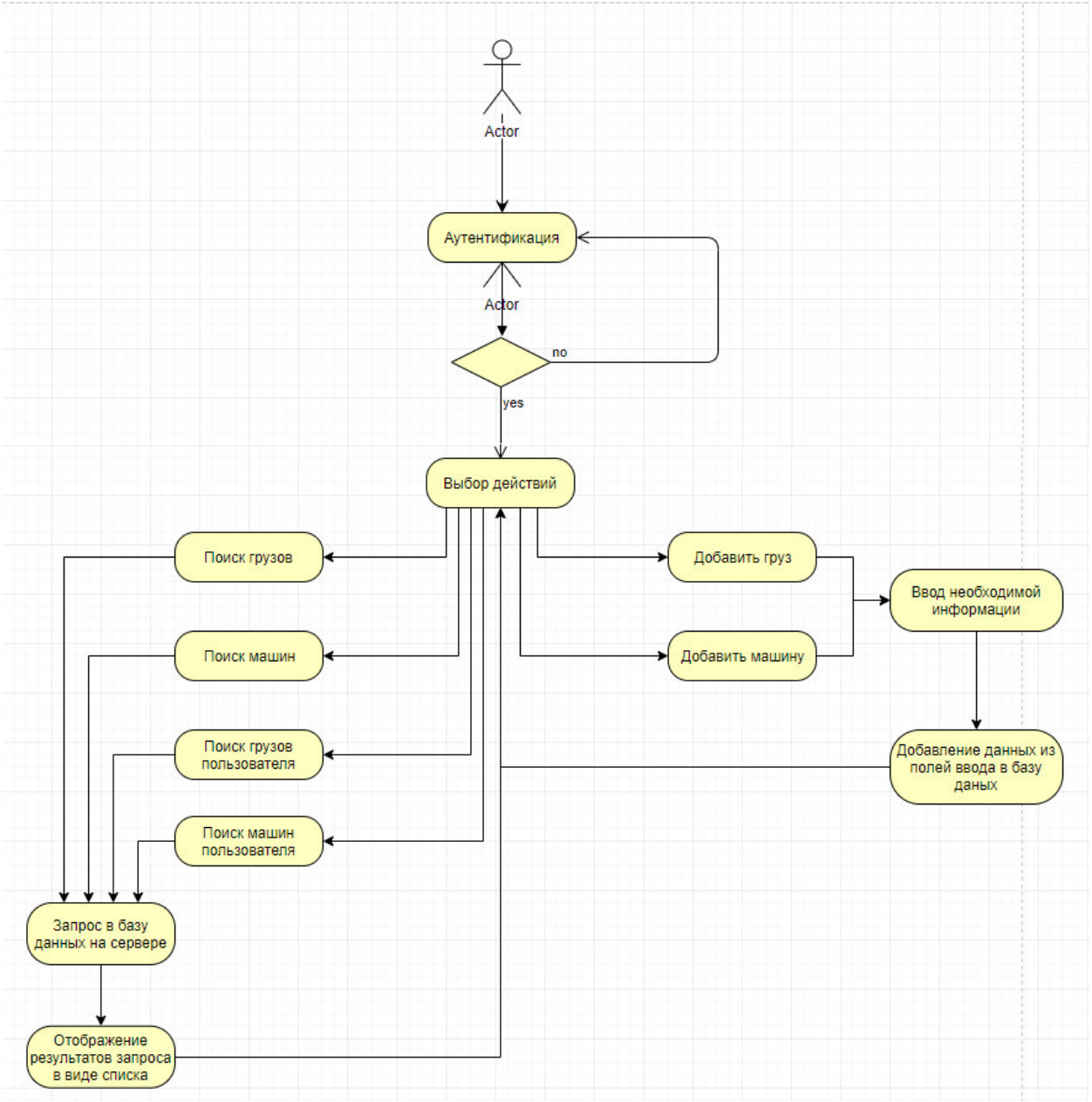

Рисунок 33 – UML диаграмма, демонстрирующая пошаговую работу приложения

3.3 Описание модулей содержащихся в приложении

Activity — это компонент приложения, который выдает экран, и с которым пользователи могут взаимодействовать для выполнения каких-либо действий, например, набрать номер телефона, сделать фото, отправить письмо или просмотреть карту. Каждой операции присваивается окно для прорисовки соответствующего пользовательского интерфейса. Обычно окно отображается во весь экран, однако его размер может быть меньше, и оно может размещаться поверх других окон.

Каждый экран мобильного приложения представлен классом Activity, представляющий собой отдельную форму приложения. Android-приложение

может включать в себе несколько активностей, взаимно общающиеся через механизм Intent-ов.

Функционально, разрабатываемое мною приложение состоит из следующих модулей (активностей):

– Основная активность содержит в себе список грузов, которые храниться в базе данных. Также активность в себе содержит NavigtionView, используемая для навигации пользователя в мобильном приложении;

– активность, подгружаемая при старте приложения, предназначена для аутентификации пользователя с помощью Firebase Authentication.

– активность «Мои грузы» содержит список грузов конкретного пользователя. Также активность содержит кнопку добавления, при нажатии на которой, подгружается новая активность для ввода данных;

– активность «Мои машины» содержит список машин конкретного пользователя. Также в ней присутствует кнопка, при нажатии на которой, подгружается новая активность для ввода данных;

– активность «Все машины» содержит список всех машин, имеющихся в базе данных. Активность для отображения элементов в виде списка использует RecyclerView. При нажатии на конкретной элемент списка, подгружается новая активность, которая содержит в себе более подробную информацию о выбранном объекте;

– активность, вызываемая при нажатии на элемент списка, содержит в себе несколько элементов TextView, для отображения информации о выбранном элементе. Также активность содержит кнопку «Позвонить», при нажатии на которой осуществляется вызов автора данного объекта;

– активность «Поиск», содержит поля ввода для формирования запроса по критериям поиска. Критерии поиска передаются на следующую активность, которая в свою очередь отображает результаты запроса по заданным критериям;

– активность, вызываемая при нажатии на кнопку «Добавить», в свою очередь, содержит поля ввода, необходимые для заполнения информации об объекте.

Схема активностей разработанного мною приложения представлена ниже на рисунке 34 с помощью диаграммы классов UML.

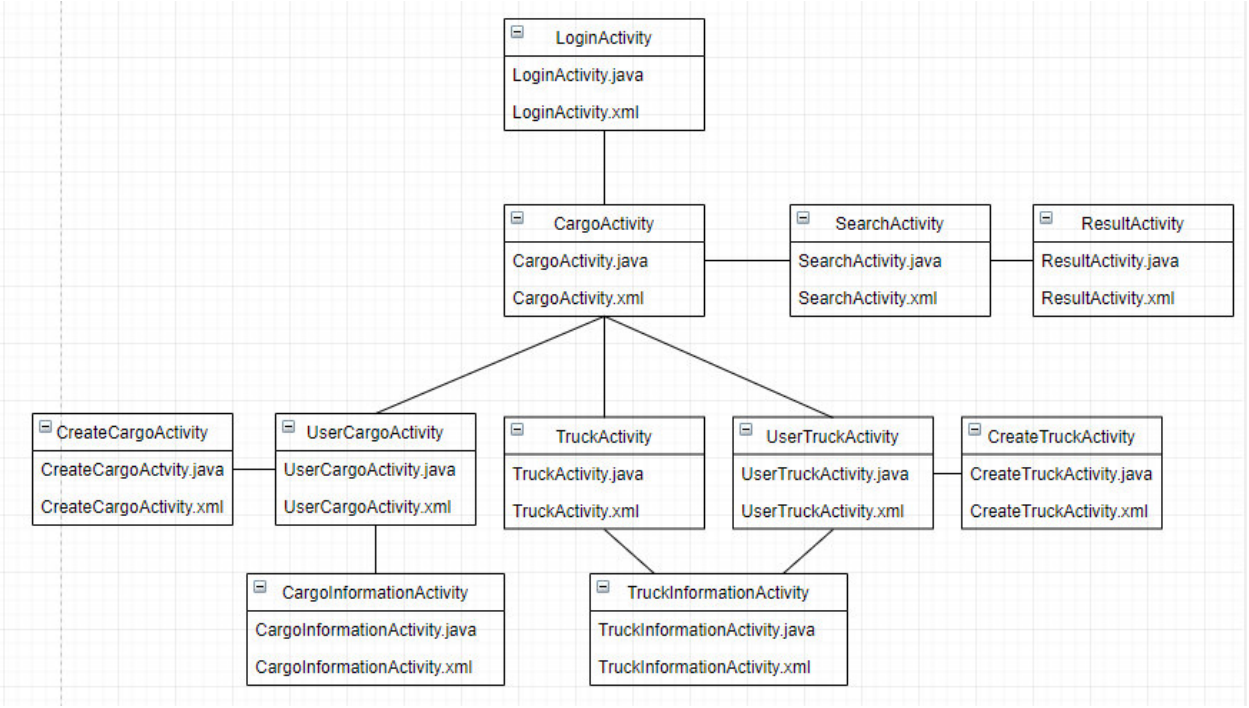

Рисунок 34 – Активности приложения и связи между ними

3.4 Описание функциональности

3.4.1 Стартовая страница

При открытии приложения пользователь видит стартовую страницу, представляющую собой страницу авторизации. Стартовая страница содержит следующие элементы:

– два поля, для ввода почты и пароля соответственно;

– две кнопки для регистрации и авторизации.

При вводе неправильного e-mail или пароля, всплывает сообщение, информирующее пользователя о неправильном вводе. При успешном вводе, осуществляется переход на главную страницу приложения. Кнопка Регистрация служит для регистрации нового пользователя. При попытке регистрации уже созданного пользователя, всплывает сообщение, информирующее об этом. При успешной регистрации пользователя, так же всплывает сообщение, информирующее об успешной авторизации. Весь функционал главной страницы проиллюстрирован на рисунках.

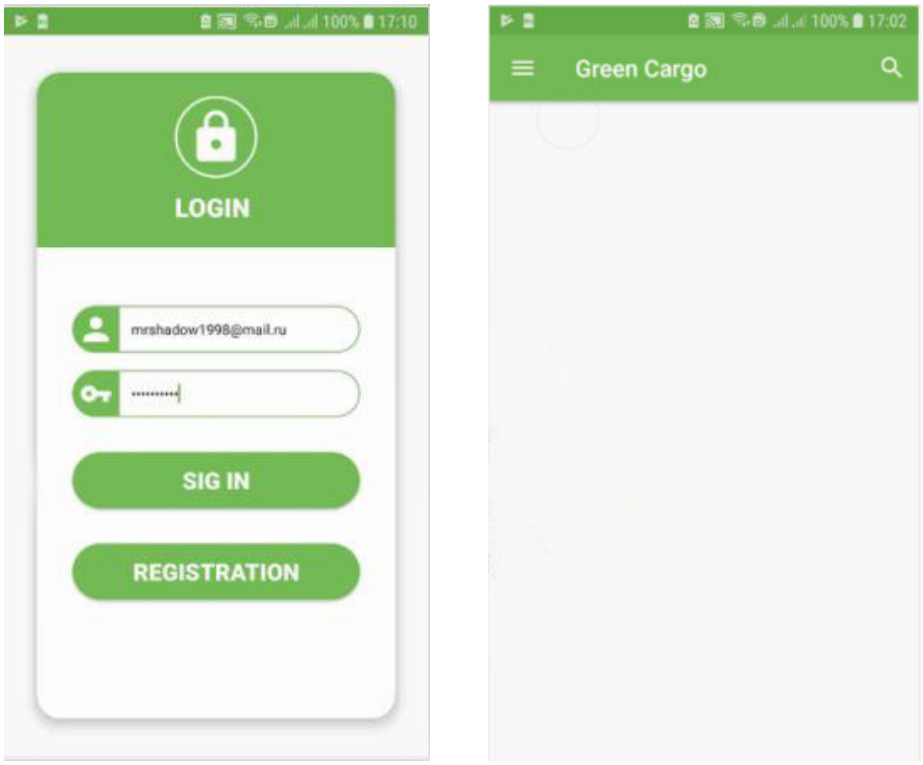

Рисунок 35 – Успешная авторизация

Рисунок 36 – Попытка ввода неправильных данных

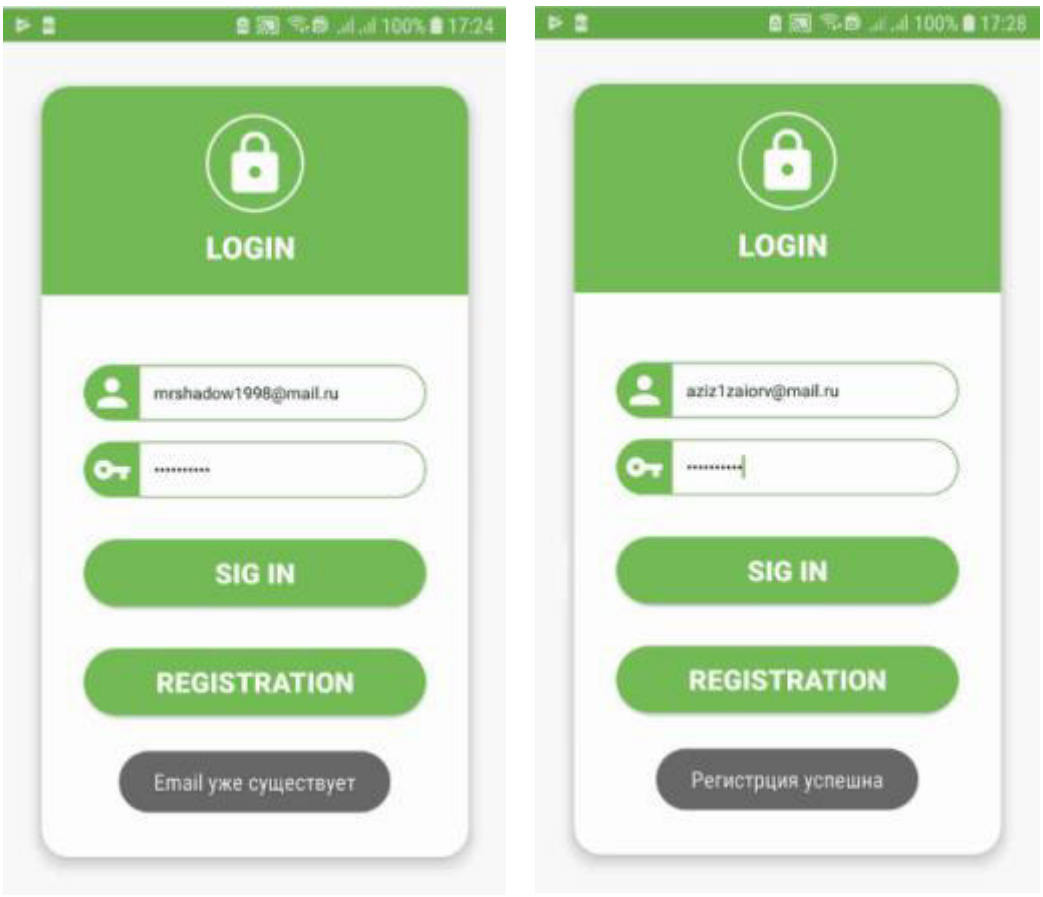

Рисунок 37 – Регистрация пользователя

3.4.2 Главная страница

Главная страница приложения подгружается после успешной авторизации пользователя в приложении. Страница состоит из следующих элементов:

– панель инструментов (ActionBar) , содержащая заголовок экрана, иконку поиска и Burger, при нажатии на которой всплывает навигационная панель;

– список, элементами которого являются объявления о грузах, содержащихся в базе данных.

Для реализации списка использовался RecycleView, так как он обладает рядом преимуществ в отличии ListView:

– сам виджет больше не берёт на себя обязанность по размещению элементов;

– паттерн ViewHolder стал обязательным. Причём виджет умеет заново использовать уже созданные ViewHolder'ы и удалять уже не используемые (отсюда и название), что благоприятно сказывается на быстродействии и размере используемой памяти;

– более удобный способ работы с анимацией.

На рисунке 38 изображен вид главной страницы.

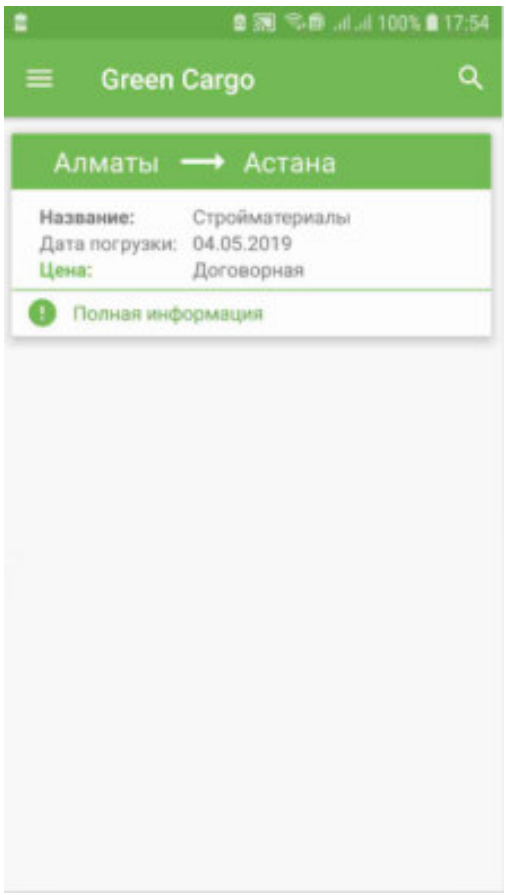

Рисунок 38 – Главная страница

3.4.3 Навигационная панель

Навигационная панель в приложении реализована с помощью элемента NavigationView, которая расположена на главной странице. Панель включает в себе следующие ссылки:

– мои грузы;
– мои машины;
– все машины;
– справка;
– выход.

При нажатии на элемент «Справка», отображается модальное окно, содержащая информацию о приложении. При нажатии на элемент «Выход» осуществляется переход на стартувую страницу приложения. При нажатии на остальные элементы открывается соответствующая страница.

На рисунке 39 изображен интерфейс навигационной панели приложения.

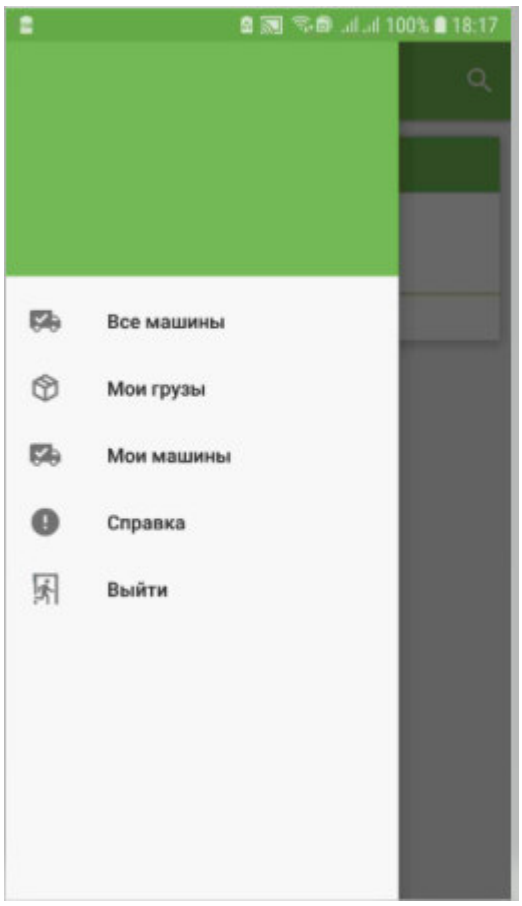

Рисунок 39 – Навигационная панель

3.4.4 Модальное окно

Модальное окно отображается при нажатии на элемент «Справка» навигационной панели. Это окно содержит краткую информацию о приложении, в которую входят:

- информация о разработчике приложения;
- информация о версии приложения;
- наименования приложения;
- год выпуска приложения.

На рисунке 40 изображен интерфейс окна, содержащего краткую информацию о приложении.

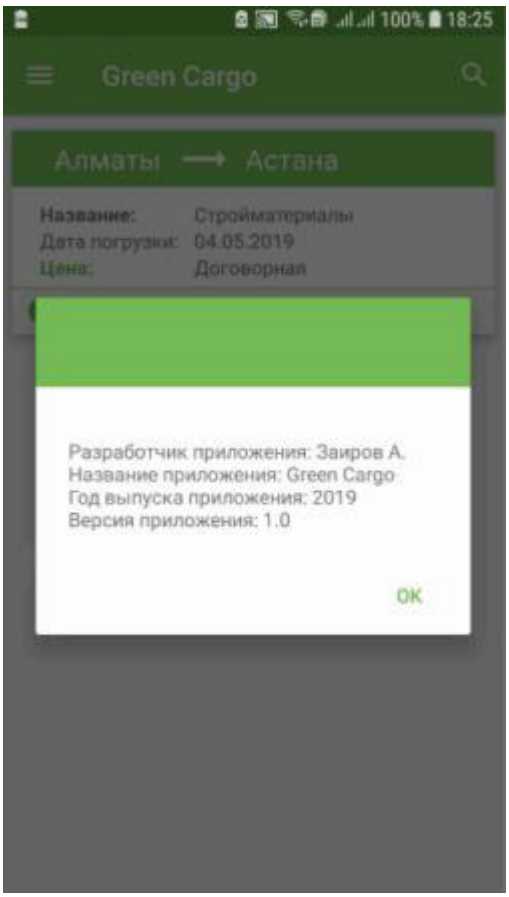

Рисунок 40 – Модальное окно

3.4.5 Создание нового груза

Страница, предназначенная для заполнения информации о грузе: состоит из следующих элементов:

- поле ввода «Откуда»;
- поле ввода «Куда»;
- поле ввода названия;
- поле ввода типа погрузки;
- поле ввода веса;
- поле ввода объема;
- поле ввода «Тип транспорта»;
- поле ввода цены;
- поле ввода даты;
- поле ввода контактных данных;
- кнопка «Добавить груз».

Поле ввода «Откуда». Поле ввода «Откуда» предназначено для ввода информации о пункте отбытия груза.

Поле ввода «Куда». Поле ввода «Куда» предназначено для ввода информации о пункте прибытия груза.

Поле ввода названия. Это поле предназначено для ввода информации о наименовании груза. Например: стройматериалы, овощи, химические вещества и т.д.

Поле ввода «Тип транспорта». Это поле предназначено для ввода информации о требованиях к транспорту. Например: Тентовый полуприцеп, рефрижератор и т.д.

Поле ввода цены. Это поле предназначено для ввода информации о предполагаемом вознаграждении за транспортировку груза. Например: 330 000 тыс. тг или указать сумму как договорную.

Поле ввода даты. Это поле предназначено для ввода информации о дате погрузки.

Поле ввода контактных данных. Это поле предназначено для ввода телефонного номера грузоотправителя.

Примеры интерфейса страницы «Добавить груз» изображен на рисунках 41 – 42.

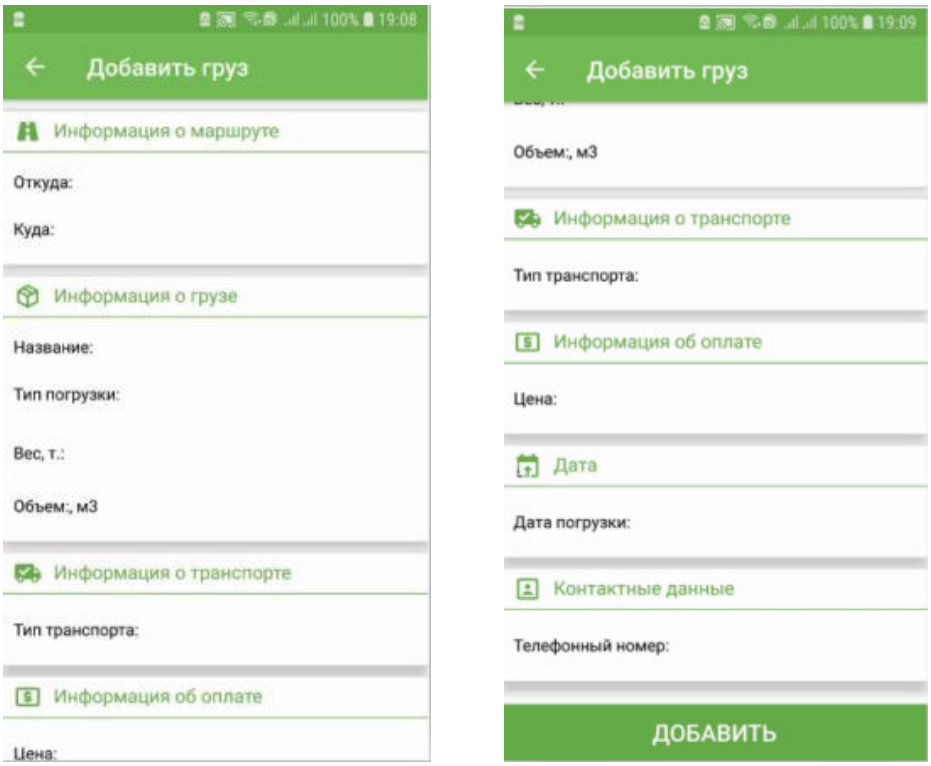

Рисунок 41 – Страница «Добавить груз»

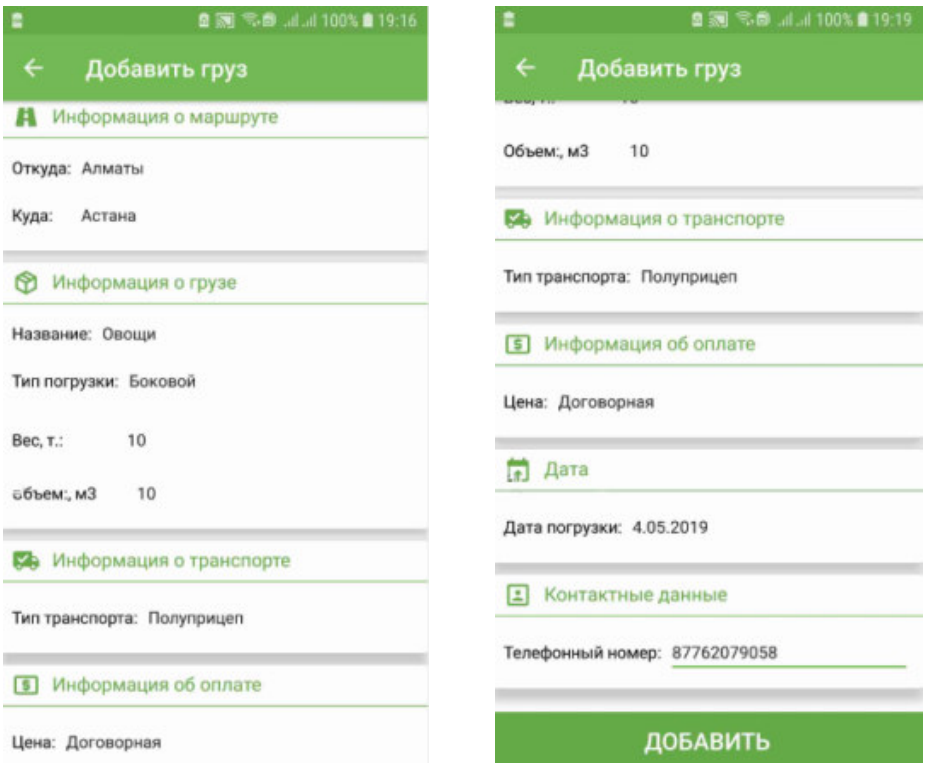

Рисунок 42 – Заполненная страница «Добавить груз»

3.4.6 Добавление нового транспорта

Страница, предназначенная для заполнения информации о транспорте, состоит из следующих элементов:

– поле ввода «Откуда»;
– поле ввода «Куда»;
– поле ввода «Полное имя»;
– поле ввода «Возраст»;
– поле ввода «Стаж работы»;
– поле ввода «Марка»;
– поле ввода «Тип транспорта»;
– поле ввода «Тип кузова»;
– поле ввода «Грузоподемность»;
– поле ввода «Дополнительно»;
– поле ввода контактных данных;
– кнопка «Добавить груз».

Поле ввода «Откуда». Поле ввода «Откуда» предназначено для ввода информации о пункте отбытия груза.

Поле ввода «Куда». Поле ввода «Куда» предназначено для ввода информации о пункте прибытия груза.

Поле ввода «Полное имя». Это поле предназначено для ввода информации о Ф.И.О водителя.

Поле ввода «Стаж работы». Это поле предназначено для ввода информации о стаже работы водителя.

Поле ввода «Тип транспорта». Это поле предназначено для ввода информации о типе транспорта. Например: Полуприцеп и т.д.

Поле ввода «Тип кузова». Это поле предназначено для ввода информации о кузове транспорта. Например: Рефрижератор и т.д.

Поле ввода «Дополнительно». Это поле предназначено для ввода дополнительной информации.

Поле ввода «Грузоподъёмность». Это поле предназначено для ввода информации о максимальной грузоподъёмности транспорта.

Поле ввода контактных данных. Это поле предназначено для ввода телефонного номера водителя.

Примеры интерфейса страницы «Добавить транспорт» изображен на рисунках 43 – 44.

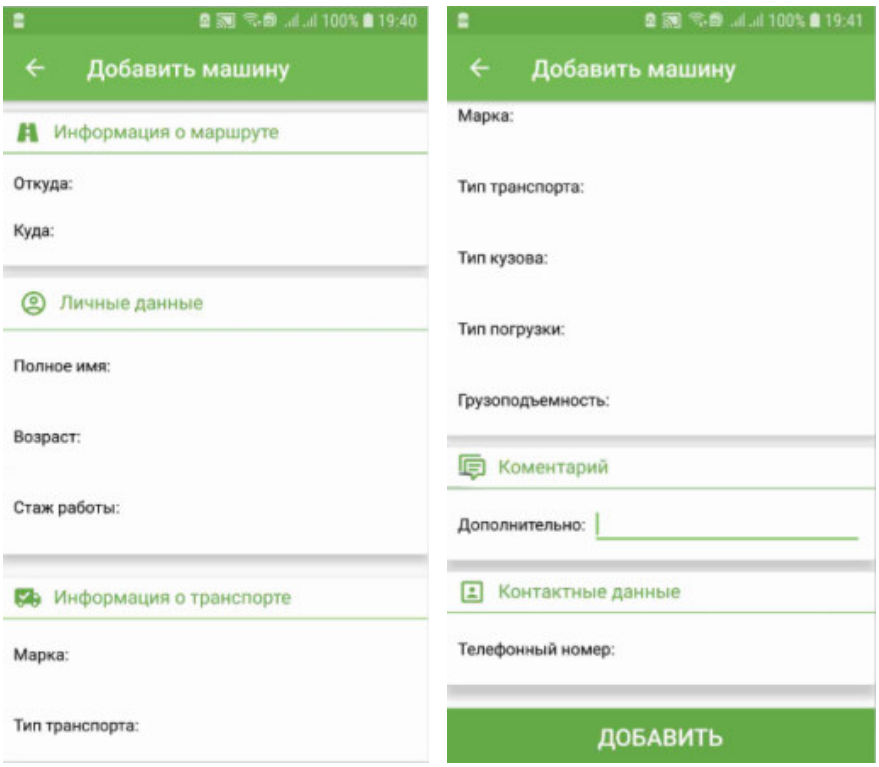

Рисунок 43 – Страница «Добавить транспорт»

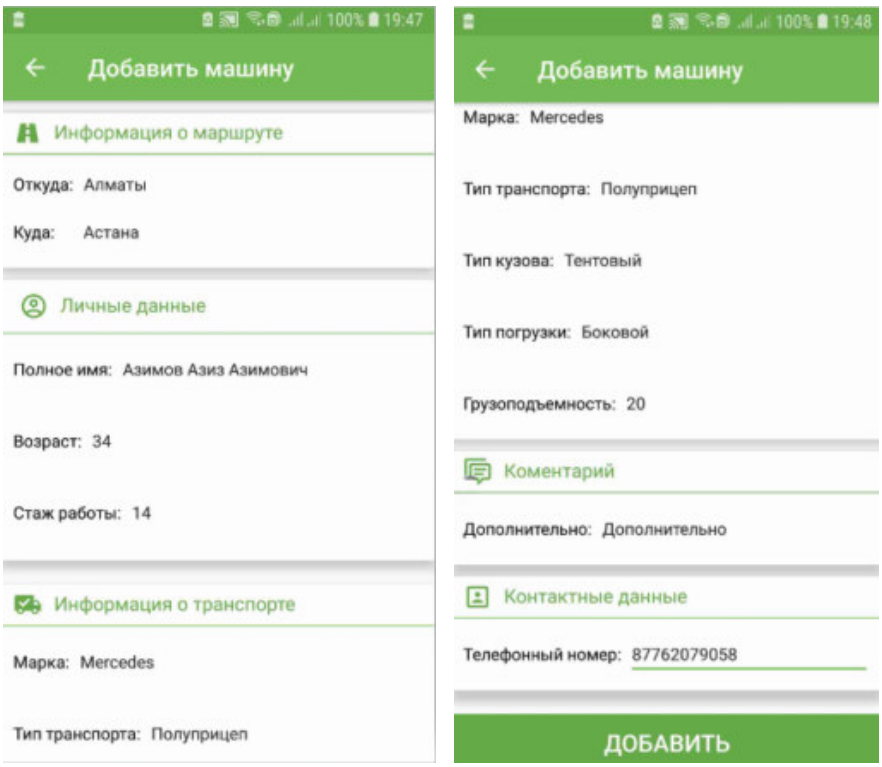

Рисунок 44 – Заполненная страница «Добавить транспорт»

3.4.7 Подробная информация о транспорте

Данная страница подгружается в результате нажатия на элемент списка и предназначена для отображения информации о транспорте. Данная страница разделена на следующие блоки:

– информация о маршруте;
– личные данные;
– информация о транспорте;
– комментарий;
– контактные данные.

Блок «Информация о маршруте» содержит следующую информацию:

– пункт прибытия;
– пункт отправки.

Блок «Личные данные» содержит следующую информацию:

– Ф.И.О водителя;
– возраст;
– стаж работы.

Блок «Информация о транспорте» содержит следующую информацию:

– марка транспорта;
– тип транспорта;
– тип кузова;
– тип погрузки;
– грузоподъемность.

Блок «Комментарий» предназначен для отображения дополнительной информации.

Блок «Контактные данные» содержит следующую информацию:

– телефонный номер водителя.

Также страница содержит кнопку, при нажатии на которой, открывается экран набора номера.

На рисунках 45 – 46 изображен пример интерфейса страницы.

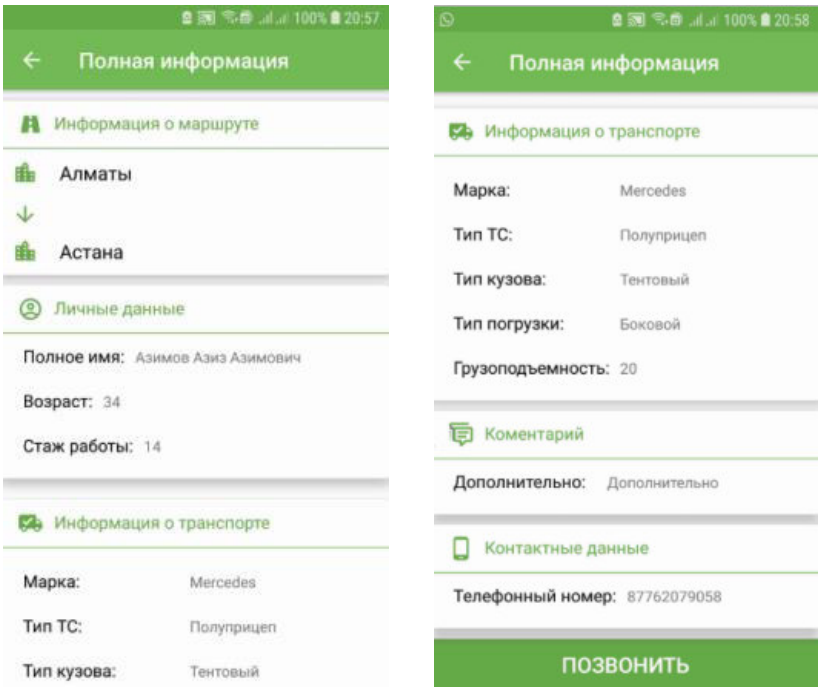

Рисунок 45 – Страница «Полная информация»

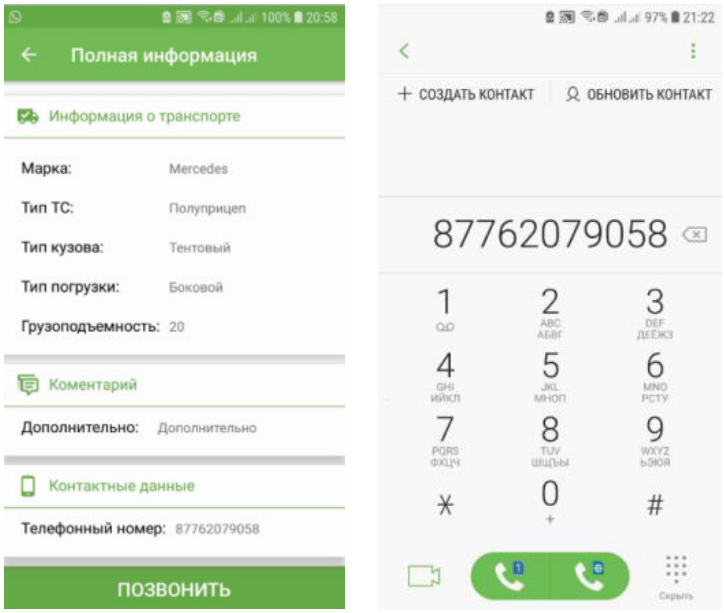

Рисунок 46 – Переход на экран набора номера

3.4.8 Подробная информация о грузе

Данная страница подгружается в результате нажатия на элемент списка и предназначена для отображения информации о ГРУЗЕ. Данная страница разделена на следующие блоки:

– информация о маршруте;
– личные данные;
– информация о грузе;
– информация о транспорте;
– информация об оплате;
– дата;
– контактные данные.

Блок «Информация о маршруте» содержит следующую информацию:

– пункт прибытия;
– пункт отправки.

Блок «Информация о транспорте» содержит следующую информацию:

– тип транспорта.

Блок «Контактные данные» содержит следующую информацию:

– телефонный номер грузоотправителя.

Блок «Информация о грузе» содержит следующую информацию:

– наименование груза
– тип погрузки
– вес груза
– объем груза

Блок «Информация о дате» сдержит информацию о дате погрузки.

Блок «Информация об оплате» содержит информация о цене за транспортировку.

Также страница содержит кнопку, при нажатии на которой, открывается экран набора номера.

На рисунках 47 – 48 изображен пример интерфейса страницы.

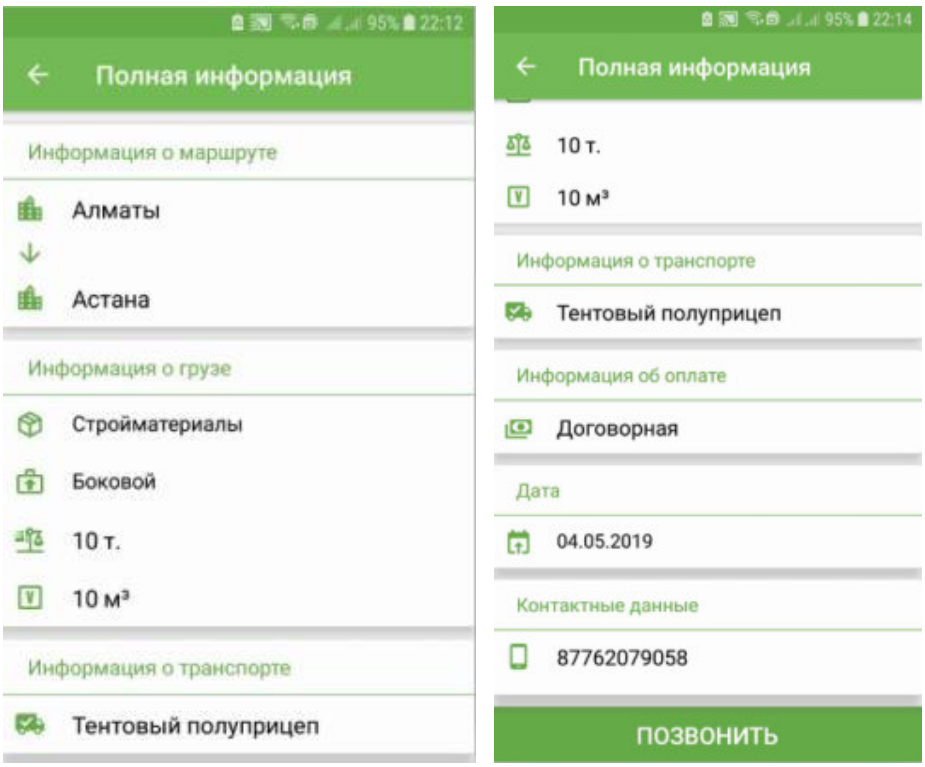

Рисунок 47 – Страница «Полная информация»

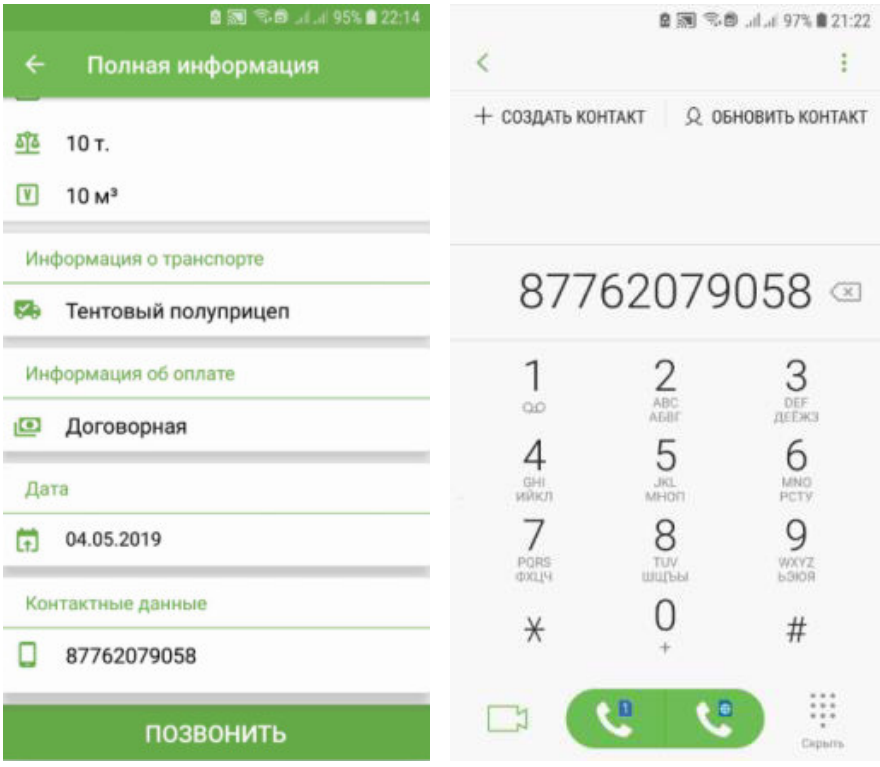

Рисунок 48 – Переход на экран набора номера

3.4.9 Страница «Мои грузы»

Страница «Мои грузы » отображает грузы, принадлежащие конкретному пользователю. Страница состоит из следующих элементов:

– панель инструментов (ActionBar) , содержащая заголовок экрана, иконку, при нажатии на которой осуществляется переход на главную страницу;

– список, элементами которого являются объявления о грузах, принадлежащие данному пользователю.

– кнопка добавления, при нажатии на которой открывается страница добавления груза.

Для реализации списка используется виджет RecycleView, преимущества которого описаны в пункте 3.4.2. Также элементы списка имеют popup меню, с помощью которого пользователь может удалить выбранный груз. Интерфейс страницы «Мои грузы» представлен на рисунке 49.

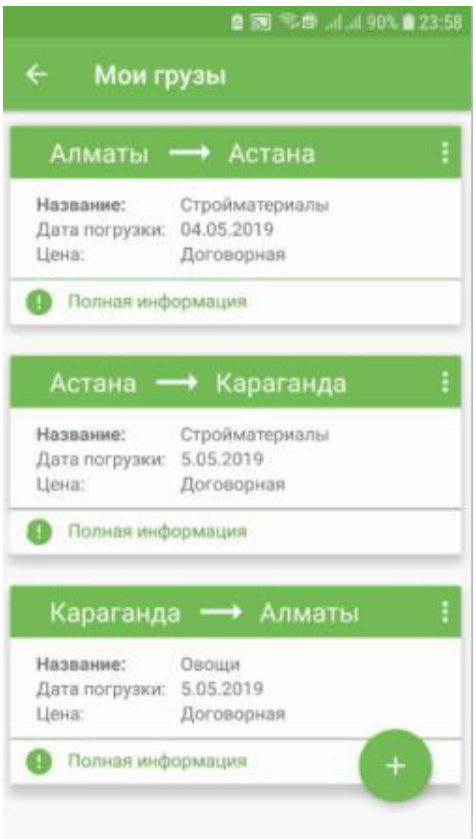

Рисунок 49 – Страница «Мои грузы»

На рисунке 50 изображен процесс удаления груза.

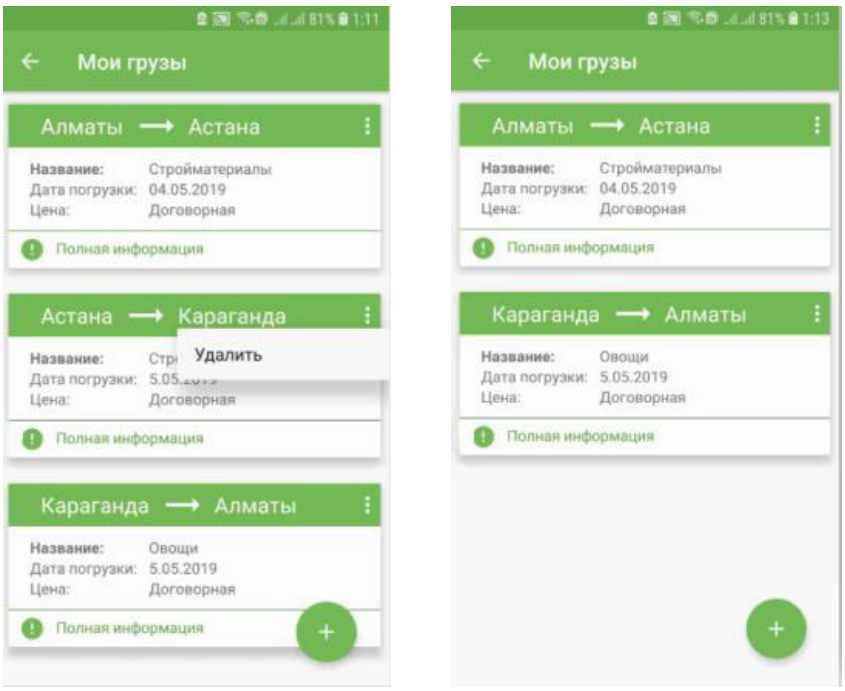

Рисунок 50 – Удаление груза

3.4.10 Страница «Мои машины»

Страница «Мои грузы » отображает машины, принадлежащие конкретному пользователю. Страница состоит из тех же элементов что и страница «Мои грузы». Интерфейс страницы «Мои машины» представлен на рисунке 51.

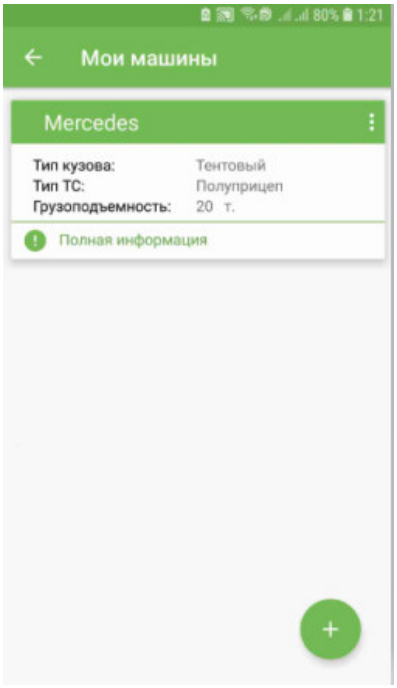

Рисунок 51 – Страница «Мои машины

3.4.11 Страница «Все машины»

Страница «Все машины» отображает список всех машин, хранящихся в базе данных Firebase Database. Панель инструментов страницы содержит иконку поиска, при нажатии на которой подгружается экран поиска. На рисунке 52 представлен интерфейс страницы «Все машины».

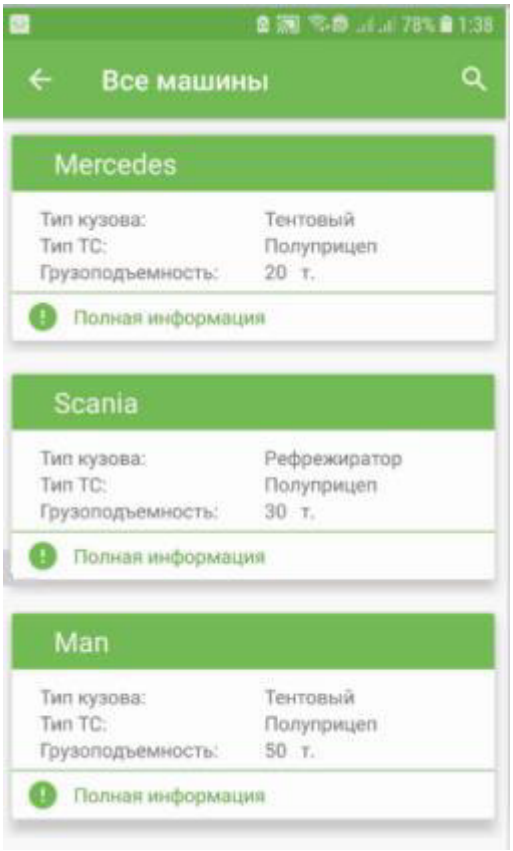

Рисунок 52 – Страница «Все машины»

3.4.12 Страница «Поиск»

Страница поиск предназначена для поиска грузов/машин по месту назначения и месту прибытия. Страница состоит из следующих элементов:

– поле ввода «Откуда»;

– поле ввода «Куда»;

– кнопка «Поиск».

При отправки запроса с пустыми критериями отображается список всех грузов/машин. Список результатов выводится в новом экране.

Интерфейс экрана поиска представлен на рисунке 53.

На рисунке 54 представлен пример поиска грузов.

На рисунке 55 представлен пример поиска машин.

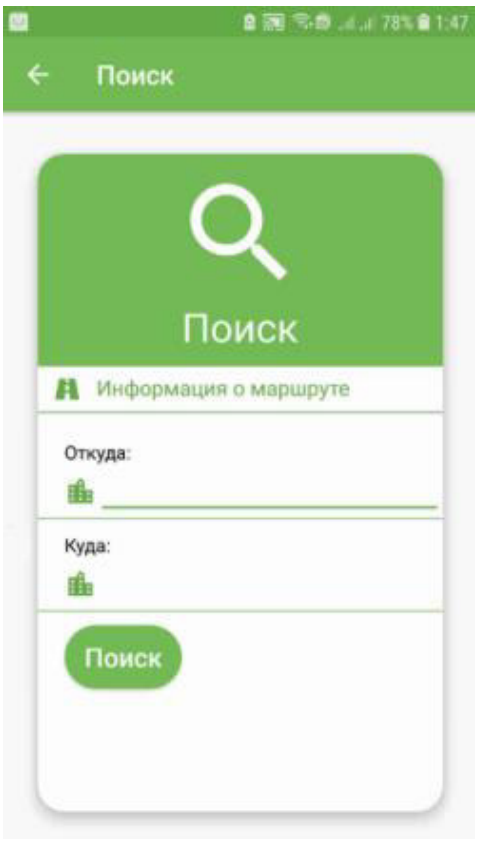

Рисунок 53 – Страница «Поиск»

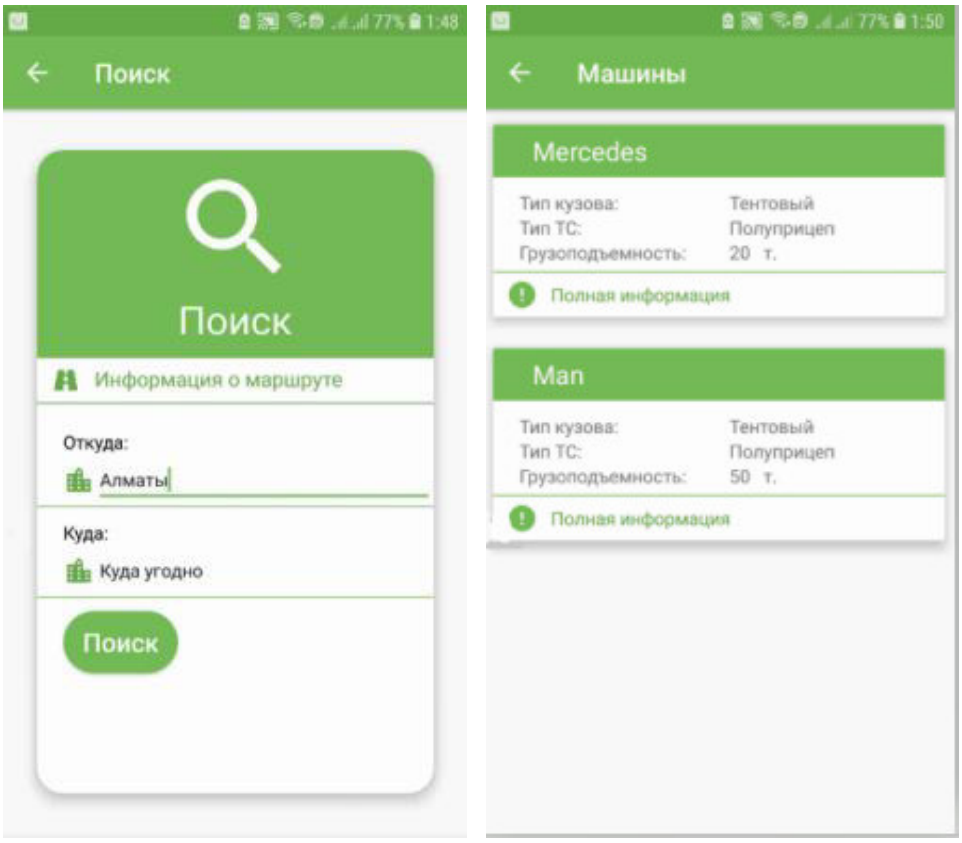

Рисунок 54 – Пример поиска машин

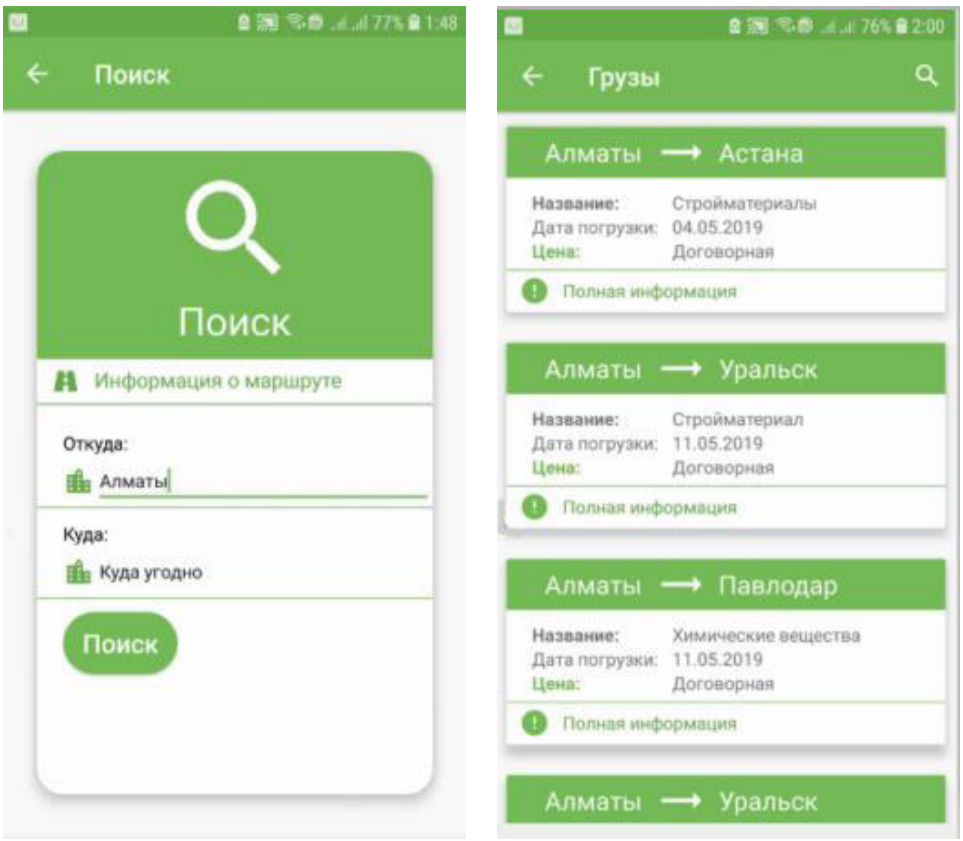

Рисунок 55 – Пример поиска грузов

3.5 Этап тестирования

В процессе разработки приложения производилось поэтапное тестирование с целью выявления программных ошибок и несоответствий ТЗ (техническому заданию). Для этого был созданы эмуляторы смартфона с разными диагоналями экрана для разных версий Android. Тестируемый программный продукт последовательно запускался на этих эмуляторах, его поведение анализировалось, и при необходимости по результатам анализа вносились изменения в код.

Были проведены приведенные ниже тесты:

1. приложение было запущено на устройствах, работающих под управлением разных версий Android с целью выявления особенностей работы приложения, запущенного в разных операционных системах.
2. после завершения цикла разработки, программный продукт тестировался на реальных устройствах.

Тестирование проводилось на различных устройствах, список которых приведёнв таблице 3.15.1.

Таблица 3.15.1 – Устройства для тестирования

Устройства	Nexus 7	Samsung A5	Honor 3C
Процессор	4-ядерный 1.2 ГГц	8-ядерный 1.9 ГГц	4-ядерный 1.3 ГГц
ОЗУ	1.2 Гб	3 Гб	3 Гб
Разрешение	1280x800	1920x1080	720x1280
Диагональ	7'	5.2'	5'
Версия ОС Android	5.1.1	8.0.0	4.2.2
Версия API	22	26	17

Благодаря тестированию были выявлены и исправлены следующие ошибки:

1) обрезание шрифта;
2) отступы, не соответствующие дизайну;
3) ошибки с отображением курсора;
4) неверные размеры полей;
5) неверное отображение сообщений для пользователя;

Средства автоматизации тестирования не использовались.

3.6 Возможные пути развития приложения

Возможными путями развития приложения являются:

1. Интеграция приложения со сторонними базами данных;
2. Реализация двухкомпонентной базы данных (локальной базы данных и базы данных в облаке);
4. Взаимодействие Google приложений с картами;
5. Реализация интерфейса на различных языках.

4 ЭКОНОМИЧЕСКАЯ ЧАСТЬ

Внедрение разрабатываемого мною приложения несет за собой сокращение заработной платы сотруднику, работающему диспетчером в данной компании, а также сокращение потребления электроэнергии, бумаги которые расходуются в процессе работы диспетчера.

Основной задачей проведения экономического исследования внедрения разрабатываемого приложения является определение величины экономической эффективности от внедрения разрабатываемого мобильного приложения.

Раздел технико-экономического обоснования разработки мобильного приложения в основном включает следующие основные этапы:

– определение трудовых затрат на разработку ПП;
– расчет материальных затрат на разработку ПП;
– расчет затрат на оплату труда для разработки ПП;
– определение предполагаемой (договорной) стоимости ПП;
– вычисление себестоимости ПП;
– определение срока окупаемости разрабатываемого ПП.

Именно этому будут посвящены следующие разделы этой главы.

Целью проводимых расчетов является определение сроков окупаемости внедряемого программного продукта.

4.1 Определение трудоемкости разработки программного продукта

Трудоемкость разработки ПП – это зависимость количества задач от объема обрабатываемых данных. Также важно ответить, что трудоемкость разработки программного продукта определятся отдельно для каждого этапа разработки программного обеспечения. Трудоемкость в первую очередь позволяет определить соотношение затрат сил и времени. Также расчет трудоемкости позволяет определить, максимально возможный уровень производительности, с целью его последующего оптимизирования и увеличения эффективности.

Так как для определения трудоемкости ПП нужно выделить все этапы разработки ПП, то следует выделить основные этапы разработки программного продукта. Ниже определены основные этапы разработки ПП согласно каскадной модели:

– формирование требований;
– проектирование;
– реализация;
– тестирование;
– внедрение;
– эксплуатация и сопровождение.

В таблице ниже приведены основные этапы разрабатываемого мобильного приложения:

Таблица 4.1 – Распределение работ по этапам и видам и оценка их трудоемкости

Этапы разработки ПП	Вид работы на данном этапе	Трудоемкость разработки ПП, чел. × ч.
1. Разработка концепции	1.1 Принятие решение о начале этапа «Разработка концепции»; 1.2 Предварительный анализ осуществимости проекта;	32 часа
2. Разработка Т/З	2.1 Составление описания функционала; 2.2 Определение временных рамок и финансовых затрат на работы. 2.3 Подготовка ТЗ	24 часа
3. Этап проектирования	3.1 Создание прототипа интерфейса; 3.2 Разработка схемы всех экранов с указанием функционала; 3.4 Составление связей между экранами.	56 часа
4. Создание концепции дизайна	4.1 детальная проработка и отрисовка всех экранов.	112 часа
5. Этап разработки	5.1 Разработка первой работающей версии приложения; 5.2 Тестирование первой версии приложение; 5.3 Определение недостатков первой версии приложения; 5.4 Разработка конечной версии приложения.	112 часа
6. Тестирование	6.1 Формирование списка всех недоработок, недочетов и ошибок в функционале приложения; 6.2 Определение сроков на доработку.	16 часа

7. Отладка	7.1 начало работ по отладке 7.2 Выявление недокументированного поведения системы 7.3 Устранение небезопасного кода	24 часа
8. Этап реализации	8.1 Согласование договора с клиентом 8.2 Подписания договора с клиентом	2 часа
ИТОГО трудоемкость выполнения проекта		378 часа

Таким образом, в данном разделе была определена общая трудоемкость выполнения проекта. Исходя из анализа, общая трудоемкость составила 378 чел. × часа. Плановый фонд рабочего времени за месяц – 24 дней по 8 часов. Итого 192 часа в месяц. На разработку программного продукта потрачено 2 месяца. Таким образом, общий объем времени, потраченный на разработку программы, составил 378 часов.

4.2 Расчет затрат на разработку ПП

Подсчет требуемых издержек необходимых на разработку ПП определяется с помощью составления сметы, в которую входят ниже приведенные статьи:

– материальные издержки на разработку ПП;
– издержки на оплату труда;
– вычисление социального налога;
– амортизация основных фондов;
– прочие издержки на разработку ПП.

В «Материальные затраты» включаются издержки на основные и дополнительные материалы (бумага, картриджи и другие), энергия, необходимые для разработки программного обеспечения.

Материальные издержки приведение в таблице 4.2.

Таблица 4.2 – Издержки на материальные ресурсы

Наименование материального ресурса	Единица измерения	Количество израсходованного материала	Цена за единицу, тг	Сумма, тг
Бумага офисная	пачка	1	1400	1400
CD-RW	шт.	2	700	1400
Ручка	шт.	1	100	100
Папка	шт.	1	200	200
ИТОГО издержки на материальные ресурсы				3100

Общая сумма издержек на материальные компоненты ($З_м$) были выполнены по формуле:

$$З_м = \sum_{i=1}^{n} Р_i \times Ц_i$$

Где $Р_i$ – расход i-го вида материального ресурса, натуральные единицы;

$Ц_i$ – цена за единицу i-го вида материального ресурса, тг;

I – вид материального ресурса;

n – количество видов материальных ресурсов.

Подставив значения в формулу, получим общую сумму издержек на материальные ресурсы:

$$З_м = 1400 + 1400 + 100 + 200 = 3100, тг.$$

Так как в ходе разработке ПП используется электрооборудование, то необходимо рассчитать затраты на электроэнергию. В таблицы ниже приведены издержки на электроэнергию.

Таблица 4.3 – Затраты на электроэнергию

Наименование оборудования	Паспортная мощность, кВт	Коэффициент использования мощности	Время работы оборудования для разработки ПП, ч	Цена электроэнергии $\frac{тг.}{кВт * ч}$	Сумма, тг
Ноутбук	0,4	1	378	15,9	2 404,08
Принтер	0,4	1	24	15,9	152, 64
ИТОГО затраты на электроэнергию					2 557

Общая сумма издержек на электричество ($З_э$) вычисляется по следующей формуле:

$$З_i = \sum_{i=1}^{n} M_i \times K_i \times T_i \times Ц$$

где M_i – паспортная мощность i-го электрооборудования, кВт;

K_i – коэффициент использования мощности i-го электрооборудования (принимается $K_i = 0.7$) ;

T_i – время работы i-го оборудования за весь период разработки ПП, ч;

i – вид электрооборудования;

n – количество электрооборудования.

Подставив значения в формулу, получим общую сумму издержек на электроэнергию:

$$З_i = 0.4 * 1 * 378 * 15.9 + 0.4 * 1 * 378 * 15.9 = 2\,557, \text{тг.}$$

В «Издержки на оплату труда» входят издержки по оплате труда всех работников, участвующих в разработке ПП. В таблице 4.4 приведены издержки на оплату труда.

Таблица 4.4 – Затраты на оплату труда

Категория работника	Квалификация	Трудоемкость разработки ПП, чел. × ч.	Часовая ставка, тг/ч	Сумма, тг.
Программист	Начинающий	378	600	226 800
ИТОГО издержки на оплату труда				226 800

Общая сумма затрат на оплату труда ($З_т$) вычисляется по формуле:

$$З_{тр} = \sum_{i=1}^{n} ЧС_i \times T_i$$

где $ЧС_i$ – часовая ставка i-го работника, тг;
T_i – трудоемкость разработки ПП, чел. × ч;
i – категория работника
n – количество работников, участвующий в разработке ПП.
Для вычисления часовой ставки воспользуемся следующей формулой:

$$ЧС_i = \frac{ЗП_i}{ФРВ_i}$$

где $ЗП_i$ – месячная заработная плата работника, тг.
$ФРВ_i$ – месячный фонд рабочего времени i–го работника, час.
Подставив значения в формулу, получим часовую ставку:

$$ЧС_i = \frac{115\,200 \text{ тг.}}{192 \text{ ч.}} = 600 \frac{\text{тг}}{\text{ч}}$$

Подставив значения в формулу, получим общую сумму издержек на оплату труда:

$$З_{тр} = 378 * 600 = 226\,800 \text{ тг}$$

В «Социальный налог» входит сумма, которая рассчитывается как 9.5% от затрат на оплату труда работникам.

Социальный налог рассчитывался по следующей формуле:

$$З_{со} = (З_{тр} - З_{по}) * 9.5\%$$

Подставив значения в формулу получим:

$$З_{со} = (226\ 800 - 2680) * 9.5\% = 19\ 391 \text{ тг.}$$

Таким образом «Социальный налог» составляет: 19 391 тг.

В статью «Амортизация основных фондов» входит сумма амортизационных отчислений от стоимости оборудования и программного обеспечения, используемых при разработке ПП. (см. таблицу 4.5).

Таблица 4.5 – Амортизация основных фондов (ОФ)

Наименование оборудования	Стоимость оборудования и ПО, тг	Годовая Норма амортизации, %	Эффективный Фонд времени Работы оборудования и ПО, ч/год	Время работы Оборудования и ПО для разработки ПП, ч	Сумма, тг
Ноутбук	143 000	20	2304	378	4 692
Принтер	74 000	10	2304	24	77
Android Studio	Бесплатно		378	378	0
BPwin 4.0	Бесплатно		378	378	0
Rational Rose	Бесплатно		378	378	0
ИТОГО амортизация основных фондов					4 769

Общую сумму амортизационных отчислений можно вычислить по следующей формуле:

$$З_{АМ} = \sum_{i=1}^{n} \frac{Ф_i \times Н_{Ai} \times Т_{НИРi}}{100 \times T_{Эфi}}$$

где $Ф_i$ – стоимость i–го ОФ, тг

$Н_{Ai}$ – годовая норма амортизации i–го ОФ, %

$Т_{нирi}$ – время работы i–го ОФ за весь период разработки ПП, ч

$Т_{Эфi}$ – эффективный фонд времени работы i–го ОФ за год, ч/год

i – вид ОФ

n– количество ОФ.

Подставив значения в формулу, получим сумму амортизационных отчислений:

$$З_{АМ} = 4692 + 77 + 0 + 0 = 4796 \text{ тг.}$$

Годовые нормы амортизации ОФ принимается по налоговому кодексу РК или определится, исходя из возможного срока полезного использования ОФ:

$$Н_{Ai} = \frac{100}{T_{Ni}}$$

где T_{Ni} – возможный срок использования i–го ОФ, год.

В статью «Прочие затраты» включаются расходы на арендную плату, интернет и т.д.

В таблице №4.6 приведены прочие затраты на разработку программного продукта.

Таблица 4.6 – Прочие затраты

Наименование затрат	Стоимость в месяц, тг	Количество месяцев	Сумма, тг
Арендная плата	40 800	2	81 600
Интернет	4 600	2	9 200
Итого сумма на прочие затраты			90 800

Стоимость ежемесячной арендной платы было высчитано согласно следующей формуле:

$$А_{им} = S \times Ц \times n, \text{ тг}$$

Подставив значения в формулу получим:

$$А_{им} = 24 \times 1\,700 \times 2 = 81\,600 \text{ тг}$$

На основании полученных данных по отдельным статьям составим смету затрат на разработку ПП см. таблицу 4.7.

Таблица 4.7 – Смета затрат на разработку ПП

Статьи затрат	Сумма, тг.
1. Материальные затраты	5 657
2. Затраты на оплату труда	226 800
3. Отчисления на соц. нужды	19 391
4. Амортизация основных фондов	4 769
5. Прочие затраты	90 800
ИТОГО по смете	347 417

4.3 Определение возможной (договорной) цены ПП

Величина возможной цены ПП устанавливается с учетом эффективности, качества и сроков ее выполнения, на уровне, отвечающим экономическим интересам заказчика (потребителя) и исполнителя.

Для вычисления договорной цены ($Ц_Д$) воспользуемся следующей формулой:

$$Ц_Д = З_{НИР} * (1 + \frac{Р}{100})$$

где $З_{нир}$ – затраты на разработку ПП (из таблицы 4.6)

Р – средний уровень рентабельности ПП. В нашем случае средний уровень рентабельности равен 25%.

Подставив значения из таблицы 4.7 в формулу получим:

$$Ц_Д = 347\,417 * \left(1 + \frac{25}{100}\right) = 434\,271 \text{ тг.}$$

Далее определим цену реализации с учетом налога на добавленную стоимость (НДС), ставка НДС согласно Налоговому Кодексу РК равна 12 %.

Для определения цены ПП с учетом НДС воспользуемся следующей формулой:

$$Ц_Р = Ц_Д + Ц_Д * НДС$$

Подставив значения в формулу получим:

$$Ц_д = 434\,271 + 434\,271 * 0{,}12 = 486\,384 \text{ тг.}$$

Таким образом, предполагаемая стоимость разработанного программного продукта составляет 486 384 тенге.

4.4 Расчет основных затрат на внедрение ПП

К основным затратам относится только приобретение самого мобильного приложения.

Основные затраты на внедрение мобильного приложения приведены в таблице 4.8.

Таблица 4.8 – Затраты на внедрение мобильного приложения

№	Статья затрат	Стоимость за единицу	Количество	Сумма, тг
1	Приобретение программы	486 384	1 шт.	486 384
2	Обучение водителей	1000 тг/ч.	6 ч.	6 000
ИТОГО сумма затрат на внедрение				492 384

Основной статьей расходов на внедрение является приобретение программного продукта. Затраты составляют 486 384 тенге.

Обслуживанием программного продукта будут заниматься водители, которые входят в штат сотрудников компании, поэтому затраты ограничиваются обучением сотрудников правильно использовать данный продукт.

Стоимость обучения составляет 6 000 тенге.

Таким образом, стоимость внедрения мобильного приложения составляет 492 384 тенге.

4.5 Эффективность внедрения ПП

Внедрение мобильного приложения внедряет ряд преимуществ:

1. Экономия средств на канцелярские товары и расходные материалы для офисной техники;

2. экономия трудозатрат работников.

Прямую экономию средств, которая заключается в закупке канцелярских товаров, бумаги, расходных материалов для офисной техники, можно подсчитать, исходя из данных по закупке средств на предприятии.

В месяц предприятие закупает:

– канцелярские товары (стержни, карандаши, папки, файлы и т.д.) – 4 000 тг./мес.

– бумага – 12 пачек (Средний месячный расход бумаги 91%, что составляет 10,92 пачек/мес.) – 2730 тг./мес.

1 пачка бумаги стоит 1400 тг., средний месячный расход на покупку бумаги составляет 10,92 * 1400 = 15 288 тг./мес.

– заправка картриджей.

Стоимость заправки одного принтера – 2 500 тг.

Среднее количество принтеров на предприятии, нуждающихся в заправке картриджей– 2 шт./мес.

Затраты на заправку картриджей составляют 5 000 тг./мес.(2 * 2 500 = 5 000 тг./мес.)

Итого в месяц расходуется: 4 000 + 15 288 + 5 000 = 24 288 тг.

Оценка ежемесячных затрат на одного сотрудника

Заработная плата диспетчера 750 тг/час. В среднем он работает 6 часов в день, т.е. 22 дня в месяц (750*22*6=99 000 тг/мес). Также к этому прибавим социальный налог в размере 8 465 тг.

Суммируем стоимость сокращения расходов на покупку расходных материалов в офис (24 288 тг.) и заработную плату сотруднику и получим суммарный объем годовой экономической эффективности от внедрения мобильного приложения: (24 288 + 99 000 + 8 465) * 12 = 1 581 036 тг.

4.6 Расчет срока окупаемости

Рассчитаем срок окупаемости продукта по следующей формуле:

$$Т = К / П, (10),$$

где

– К – единовременные капитальные затраты при внедрении;

– П – годовая экономия, тг.

В нашем случае К = 492 384 тг., П = 1 581 036 тг.

Подставляя данные значения в формулу, получаем:

$$Т = 492\ 384 / 1\ 581\ 036 = 0{,}31$$

Умножим на 365 дней в году и получим срок окупаемости продукта в 113 дней.

Таким образом, внедрение мобильного приложения окупится через 113 дней.

На основании приведенных расчетов можно сделать вывод о том, что за счет снижения трудоемкости, устранения издержек на покупку расходных материалов. Приложение может значительно повысить эффективность деятельности предприятия и обеспечить значительную экономию средств.

5 БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

Темой дипломной работы является «Разработка мобильного приложения для логистической компании». Так как разрабатываемое приложение, будет использоваться широким кругом населения, то в качестве объекта анализа условий труда рассматривается условия труда разработчика – программиста.

5.1 Анализ условий труда на рабочем месте программиста

Объектом анализ является офисное помещение шириной 4м и длиной 6м. В помещение предусматривается одностороннее освещение, площадью светового проема 7,5м^2. Также в помещении имеется 6 светильников типа ПВЛМ - 2 х 40.

В ходе анализа показателей шума в офисном помещении, какие-либо нарушения не были обнаружены, так как единственным источником шума является ноутбук, на котором будет разрабатываться мобильное приложение.

В ходе анализа вентиляции, также не были обнаружены какие-либо нарушения, так как имеющийся кондиционер обеспечивает достаточную вентиляцию для комфортной работы разработчика-программиста.

Также немало важным фактором, препятствующим эффективной работы помимо вентиляции и шума, является освещенность рабочего места. Так как недостаточность освещения приводит к напряжению зрения, ослабляет внимание, приводит к наступлению преждевременной утомленности. Чрезмерно яркое освещение, в свою очередь, вызывает ослепление, раздражение и резь в глазах. Помимо этого неправильное направление света на рабочем месте может создавать резкие тени, блики, дезориентировать работающего.

Исходя их вышесказанного, освещенность на рабочем месте является одним из наиболее важных факторов, которая подлежит более подробному анализу. Расчет освещенности на рабочем месте сводится к определению естественного и искусственного освещения. Искусственное освещение, в свою очередь, определятся количеством и типом светильников на рабочем месте.

Подробная схема помещения представлена на рисунке 56.

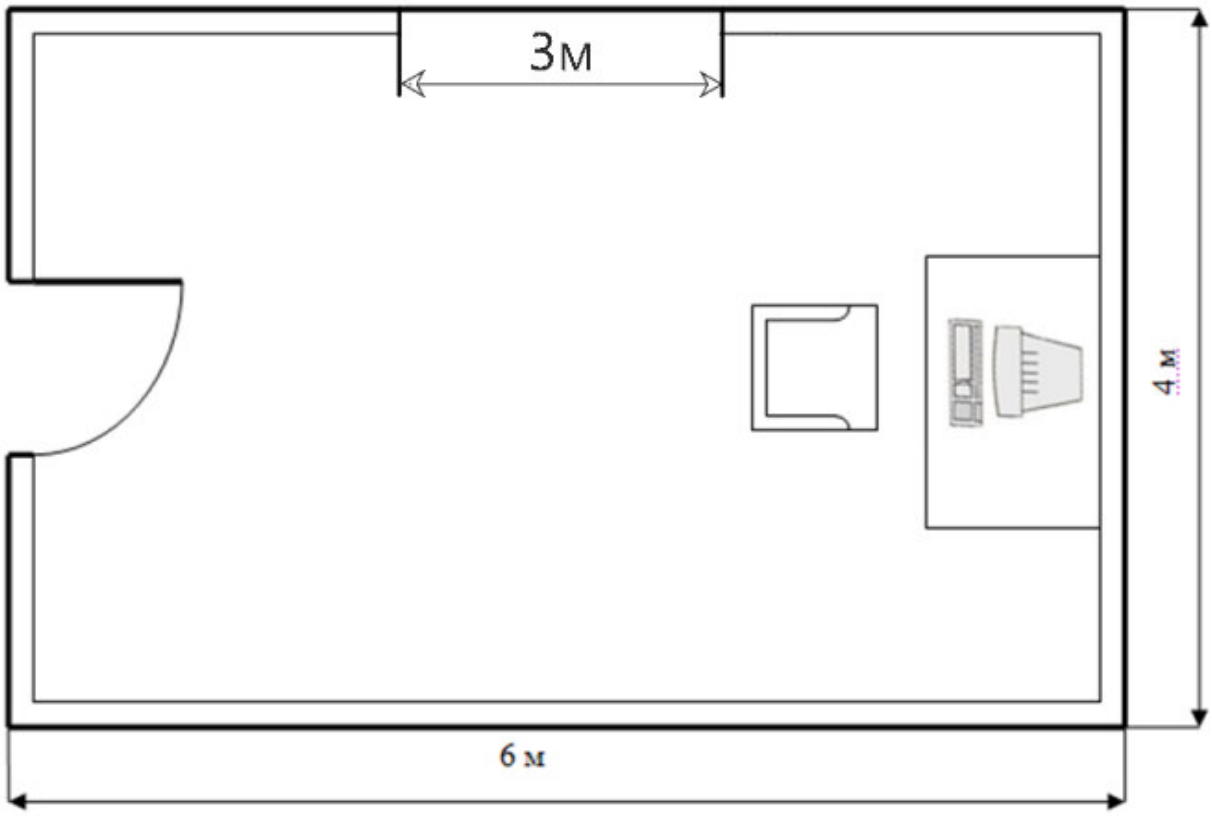

Рисунок 56 – План схема комнаты программиста

5.2 Расчет естественного и искусственного освещения

В ходе анализа условий труда, были выявлены главные факторы, препятствующие эффективной работе сотрудника на рабочем месте. Одним из немаловажных факторов является освещенность рабочего места. Исходя из вышесказанных причин, рассчитаем параметры естественного и искусственного освещения.

1) Расчет естественного освещения

Расчет освещения производится для офисной комнаты. Исходные данные помещения предоставлены ниже:

Параметры помещения:

L = 4 м;

B = 6 м;

H = 4 м;

hok = 2,5 м;

Разряд зрит. работ: IV, в

В помещении предусмотрено одностороннее боковое освещение. Площадь окна составляет 7,5 м^2.

Коэффициенты отражения:

ρпот = 70;

ρстен = 50;

ρпол = 30;

Высота начала окна hнок = 1 м;

Световой пояс: г. Алматы

Нзд = 15;

Расстояние до рядом стоящего здания, Р = 12;

Количество светильников: 6;

Тип светильника: ПВЛМ - 2 х 40;

Расчет естественного освещения заключается в определении площади световых проемов.

Общую площадь окон определяем по формуле (12) [3] для бокового освещения:

$$S_0 = \frac{S_n \cdot e_н \cdot \eta_0 \cdot K_{зд} \cdot K_з}{100 \cdot \tau_0 \cdot r_1}$$

Где Sn– площадь пола помещения, м2:

$$S_n = B \cdot L = 6 \cdot 4 = 24$$

$e_н$ – нормированное значение КЕО:

$$e_н = e_{КЕО} \cdot m$$

$e_н$ – значение КЕО для IV-го разряда В-под разряда зрительной работы:

$e_{КЕО} = 1.5$

m – коэффициент светового климата, определяется для ориентации световых проёмов по ресурсам светового климата с ориентацией окон на СВ m=1

$$e_н = 1.5 * 1 = 1.5$$

Кз – коэффициент запаса по таблице: $K_ç = 1.5$;

τ_0 – общий коэффициент светопропускания $\tau_0 = \tau_1 \cdot \tau_2 \cdot \tau_3 \cdot \tau_4$,

τ_1– коэффициент светопропускания материала по таблице 5.1: для стеклопакета $\tau_1 = 0.8$

τ_2 – коэффициент, учитывающий потери света в переплетах светопроёма по таблице 5.2: $\tau_2 = 0.75$

τ_3 – коэффициент, учитывающий потери света в несущих конструкциях, при боковом освещении равен 1.

τ_4 – коэффициент, учитывающий потери света в солнцезащитных устройствах, см. таблицу 5.3: $\tau_4 = 1$

Таблица 5.1 – коэффициент τ_1

Вид светопропускающего материала	t_1
Стекло оконное листовое:	
одинарное	0,9
двойное	0,8
тройное	0,75
Стекло витринное толщиной 6-8 мм	0,8
Стекло листовое армированное	0,6
Стекло листовое узорчатое	0,65
Стекло листовое со специальными свойствами:	
солнцезащитное	0,65
контрастное	0,75
Стеклопакеты	0,8

Таблица 5.2 – коэффициент τ_2

Вид переплета для окон промышленных зданий	t_2
Переплеты деревянные:	
одинарные	0,75
спаренные	0,7
двойные раздельные	0,6
Переплеты стальные:	
Одинарные открывающиеся	0,75
одинарные глухие	0,9
двойные открывающиеся	0,6
двойные глухие	0,8

Таблица 5.3 – коэффициент τ_4

Солнцезащитные устройства, изделия и материалы	t_4
Убирающиеся регулируемые жалюзи и шторы	1
Стационарные жалюзи и экраны с защитным углом не более 45^0при расположении пластин жалюзи или экранов под углом 90^0 к плоскости окна:	
горизонтальные	0,65
вертикальные	0,75
Горизонтальные козырьки:	
с защитным углом не более 30^0	0,8
с защитным углом от 15 до 45^0	0,9

Тогда

$$\tau_0 = 0.8 * 0.75 * 1 * 1 = 0.6$$

η_0 – световая характеристика окон по таблице 3.2 [3]:
отношения необходимые для определения η_0

$$\frac{L}{B} = \frac{5}{5} = 1$$

h1= hнок + hок = 2.5 + 1 = 3.5 м.

где h_1 – высота от уровня условной рабочей поверхности до верха окна.

$$\frac{B}{\square_1} = \frac{5}{3.5} = 1.43$$

Таким образом, $\eta_0 = 10.5$

r1 – коэффициент, учитывающий повышение КЕО при боковом освещение благодаря свету, отраженному от поверхностей помещения и подстилающего слоя, прилегающего к зданию, см. таблицу 3.9 [3]:

$$\frac{P_{пот} + P_{ст} + P_{пол}}{3} = \frac{70 + 50 + 30}{3} = 50\%$$

$$r_1 = 1.05$$

Кзд – коэффициент, учитывающий затенение окон противостоящими зданиями по таблице 3.8 [3]:

$$\frac{P}{H_{зд}} = \frac{12}{15} = 0.8$$

Кзд= 1.4

Подставим все значения в расчетную формулу:

$$S_0 = \frac{24 \cdot 1.5 \cdot 10.5 \cdot 1.4 \cdot 1.5}{100 \cdot 0.6 \cdot 1.35} \approx 10 \text{ м}^2.$$

Так как в помещении предусмотрено одностороннее боковое освещение, то площадь светового проёма исходя из расчетов должна составлять 10 м^2.

Так как высота оконного проема 2,5 м, то, следовательно, длина его должна составлять 10:2,5 = 4 м.

Таким образом, площадь светового проема должна составлять 10м^2 (4 * 2,5 м). Так как фактическая площадь светового проёма в помещении не соответствует с расчетами, то недостаток естественного освещение компенсируем искусственным освещением.

2) Расчет искусственного освещения

Расчет освещенности рабочего места программиста заключается в выборе системы освещения, определению необходимого числа светильников, их типа и размещения. Именно этому посвящен этот раздел.

Разряд зрительной работы IV (б), поэтому нормируемая освещенность по таблице 1 [7] – 300 лк.

Точечным методом проверим соответствие данного количества и типа светильников нормируемой величине (см. рисунок 57).

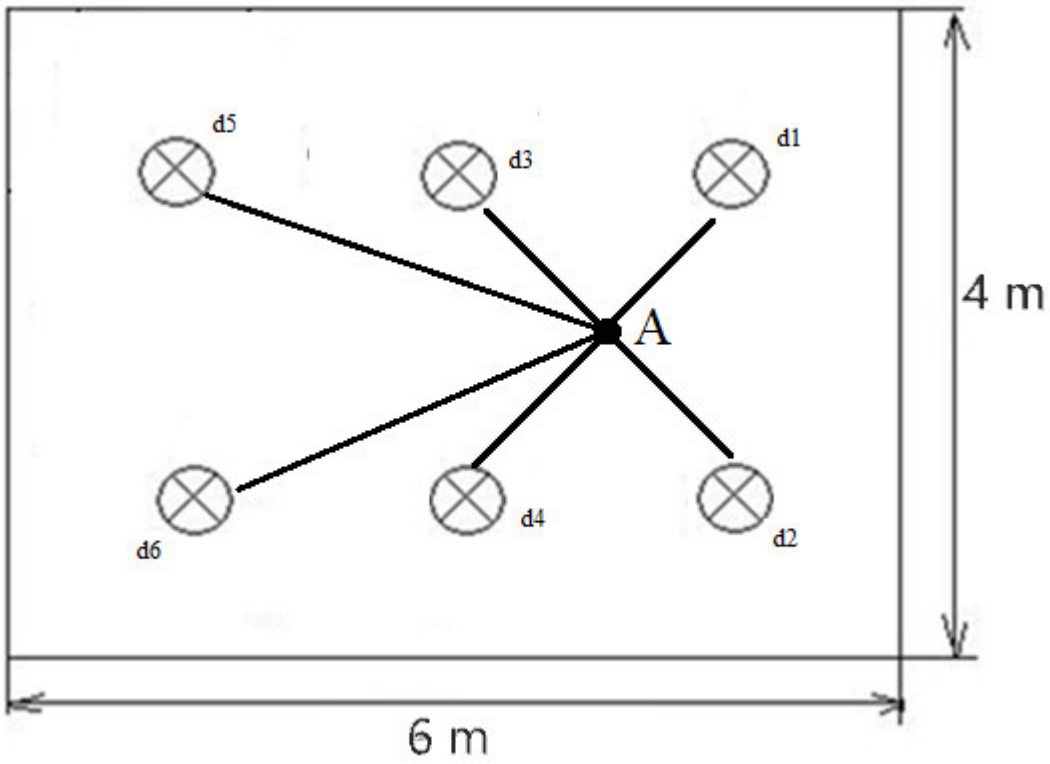

Рисунок 57 – Расположение светильников в офисном помещении

Определение расчетной высоты подвеса:

$$h_{расч}=H - (h_{раб.пов}+ h_{свеса}),$$

$$h_{расч}= 4 - (1,5 + 1) = 1,5 \text{ м}$$

Расстояние между светильниками (Z):

$$L_{A,B}= \lambda \cdot h_{расч}, \text{ где } \lambda =0.6 \div 2$$

1. В длину:

$$L_A=1 \cdot 1,5 = 1,5 \text{ м}.$$

2. В ширину:

$$L_B =1 \cdot 1,5 = 1,5 \text{ м},$$

$$l_{A,B}= L_{A,B}/2,$$

$$l_A= 1,5/2= 0,75 \text{ м}, \quad l_B= 1,5/2= 0,75 \text{ м}.$$

Намечаем контрольную точку А. Для нее определяем суммарную условную освещенность всех светильников следующим образом:

Находим проекцию расстояния на потолок от точки А до светильника-di.

Далее определяем угол между потолком и прямой di. По этому углу находим условную освещенность. Проверим, выполняется ли условие:

$$Ег \geq Енорм$$

Ег рассчитывается по следующей формуле:

$$E_{Г} = F \cdot \mu \cdot \frac{\sum_{i=1}^{m} e_{Гi}}{1000 \cdot K_{з}}$$

Где:

– Коэффициент запаса Кз = 1.1;

– Коэффициент, учитывающий действие равноудаленных светильников $\mu = 1.1$;

– Световой поток лампы F = 2250.

$e_{гi}$ рассчитывается по следующей формуле:

$$e_{Гi} = \frac{I_{\alpha_i} \cos^3(\alpha_i)}{h_{расч}^2}$$

Где:

$I_{\alpha i}$ – сила света в направлении от источника на заданную точку рабочей поверхности, которую будем находить по таблице 5.4;

α_i – угол между нормалью к рабочей поверхности и направлением силы света к расчетной точке, которая рассчитывается по следующей формуле:

$$\alpha_i = \operatorname{arctg}\left(\frac{d_i}{h}\right)$$

Таблица 5.4 – Светораспределение светильника ПВЛМ 2 х 40

Сила света Iα, кд в направлении α, градусов											
Тип свет-ка	0	5	15	25	35	45	55	65	75	85	90
ПВЛМ 2х40	139	135	132	115	104	84	64	44	22	6	0

Если внимательно рассмотреть расположение светильников в помещении, то можно заметить, что расстояние от центральной точки до светильника di делится на группы, то есть расстояние d1 для первого

светильника равно такому же расстоянию для светильников 2, 3, 4. А расстояние d5 равен расстоянию d6. Тогда найдем суммарный уровень освещенности.

$$d_1 = 1{,}42 \text{ м},$$

$$\alpha_1 = arctg\left(\frac{1{,}42}{1.5}\right) = 44$$

по этому значению, берем Iα для ПВЛМ-2×40

$$I\alpha 1 = 84 \text{ кд},$$

$$e_{r1} = \frac{84 * cos^3(44)}{1.5^2} = 19,\ 86 \text{ лк}$$

Вычислим $E_{Г2}$:

$$d_2 = 1{,}42 \text{ м},$$

$$\alpha_2 = arctg\left(\frac{1.42}{1.5}\right) = 44\ ,\quad I_{\alpha 2} = 84 \text{ кд},$$

$$e_{r2} = \frac{84 * cos^3(44)}{1.5^2} = 19{,}86 \text{ лк}$$

Для $E_{Г3}$:

$$d_3 = 1{,}42 \text{ м},$$

$$\alpha_3 = arctg\left(\frac{1.42}{1.5}\right) = 44\ ,\quad ,\ I\alpha 3 = 84 \text{ кд},$$

$$e_{r3} = \frac{84 * cos^3(44)}{1.15^2} = 19{,}86 \text{ лк}.$$

Для $E_{Г4}$:

$$d_4 = 1{,}42 \text{ м},$$

$$\alpha_4 = arctg\left(\frac{1.42}{1.5}\right) = 44\ ,\quad ,\ I\alpha 4 = 84 \text{ кд},$$

$$e_{r4} = \frac{84 * cos^3(44)}{1.15^2} = 19{,}86 \text{ лк}.$$

Для $E_{Г5}$:

$$d_5 = 3{,}16 \text{ м},$$

$$\alpha_5 = arctg\left(\frac{3.16}{1.5}\right) = 57\ ,\quad ,\ I\alpha 5 = 63 \text{ кд},$$

$$e_{r5} = \frac{63 * cos^3(57)}{1.15^2} = 9,93 \text{ лк.}$$

Для $E_{Г6}$:

$$d_6 = 3{,}16 \text{ м},$$

$$\alpha_6 = arctg\left(\frac{3.16}{1.5}\right) = 57\,, \quad \text{, Iα6 = 63 кд,}$$

$$e_{r6} = \frac{63 * cos^3(57)}{1.15^2} = 9{,}93 \text{ лк.}$$

Суммарная условная освещенность равна:

$$\sum \mathrm{e_г} = 19{,}86 * 4 + 9{,}93 * 2 = 99.3$$

Световой поток используемой лампы ПВЛМ – 2 х 40, мощностью 40 Вт $F = 2250$ лм, таблица 2.12 [6].

Суммарная освещенность равна:

$$\mathrm{E_{АГ}} = 2250 * 1.1 * \frac{99.3}{1000 * 1.1} = 223{,}425 \text{ лк}$$

Согласно СНиП, в помещении, где предполагается разряд зрительной работы III,в (высокой точности), уровень освещенности при общем освещении должен быть не ниже 300 лк.

Так как уровень освещенность явно недостаточен, то произведем реконструкцию, применяя метод коэффициента использования.

Определим индекс помещения (i):

$$i = \frac{A \cdot B}{\square_{расч} \cdot (A + B)} = \frac{4 \cdot 6}{1.5 * (4 + 6)} = 1.6$$

Определим коэффициент использования светового потока (η) по таблице 5.12 [6]: η=55%.

Количество ламп при необходимой освещенности Е=300 лк:

$$N = \frac{E_н \cdot S \cdot Z \cdot K_з}{F \cdot \eta}$$

где Z – коэффициент неравномерности освещения, равный 1.1÷1.2;

Кз – коэффициент запаса, принимаемый равным 1.3 для заданного типа помещения.

$$N = \frac{300 \cdot 24 \cdot 1.1 \cdot 1.3}{2250 \cdot 0.55} = 8 \text{ шт.}$$

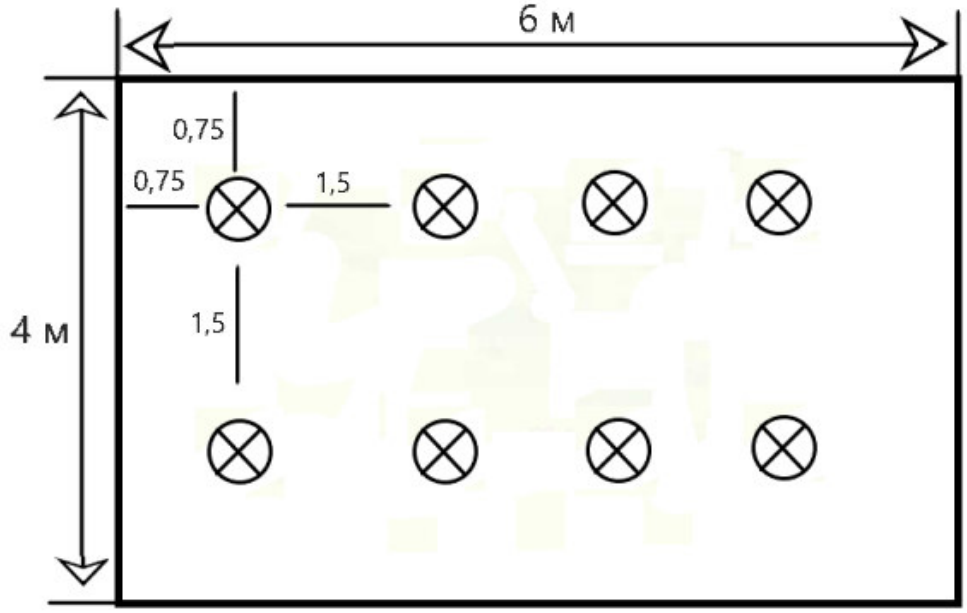

Рисунок 58– Новая схема расположения светильников

Таким образом, в данном разделе был произведен анализ освещенности рабочего места. В ходе анализ были рассчитаны показатели естественного и искусственного освещения. В ходе анализа был обнаружен недостаток в естественном освещении. Поэтому было решено компенсировать этот недостаток искусственным освещением. Исходя из расчетов, было предложено изменить количество светильников с шести на восемь, не меняя при этом тип самого светильника.

Заключение

В результате выполнения дипломной работы были разработаны и спроектированы архитектура и интерфейс мобильного приложения для платформы Android, которое позволяет упростить поиск грузов и грузоперевозчиков, обеспечить быстрый доступ к контактной информации.

В процессе выполнения дипломной работы были решены следующие задачи:

– произведен анализ рынка на аналогичные продукты;

– спроектирована архитектура и интерфейс мобильного приложения, удовлетворяющие общим стандартам для платформы Android;

– разработано приложение, обеспечивающее эффективный поиск грузов и грузоперевозчиков;

– выполнен расчет экономической эффективности от внедрения приложения.

В процессе работы были изучены и использованы архитектурные особенности платформы Android.

При проектировании приложений были использованы подходящие паттерны проектирования.

Тестирование позволило выявить некоторые недочёты приложений и исправить их в процессе разработки

В ходе проектирования информационной системы применялись такие

CASE-средства как ERWin для построения ER – диаграммы, BPWin 4.0 для моделирования бизнес-процессов, также UML-средства – Rational Rose для моделирования функциональных возможностей системы и построения диаграмм взаимодействия бизнес – процессов

Экономическая эффективность от внедрения мобильного приложения составляет 1 581 036 тг.

На данный момент приложение поддерживается версиями операционной системы Android 5.5 и выше. В дальнейшем планируется перенести дизайн интерфейса на настольную и планшетную версию, что позволит значительно расширить потребительскую аудиторию. Дальнейшие пути развития приложения описаны в разделе 3.6.

Список литературы

1. Варакин М.В. Разработка мобильных приложений под Android. УЦ «Специалист» при МГТУ им. Н. Э. Баумана, 2012.
2. Martin Fowler. UML Distilled: A Brief Guide to the Standard Object Modeling Language (Object Technology Series). Addison Wesley, 2003.
3. Абдимуратов Ж. С., Мананбаева С. Е. Безопасность жизнедеятельности. Методические указания к выполнению раздела «Расчет производственного освещения» в выпускных работах для всех специальностей. – Алматы: АУЭС, 2013. – 20 с.
4. Бекишева А. И. Методические указания к выполнению экономической части дипломной работы для бакалавров специальности 5В0703 – Информационные системы – Алматы: АУЭС; 2013. – 24 с.
5. Java. Библиотека профессионала, том 1.10-е издание. : Пер. с англ. – М. : ООО «И. Д. Вильямс», 2016. – 864с.
6. Справочная книга для проектирования электрического освещения/ Под ред. Г. М. Кнорринга. – Л.: Энергия, 1976.
7. СНиП РК 2.04-05-2002 Естественное и искусственное освещение. Государственные нормативы в области архитектуры, градостроительства и строительства.
8. Java. Библиотека профессионала, том 2.10-е издание. : Пер. с англ. – СпБ. : ООО «Альфа-книга», 2017. – 976с.
9. Java: эффективное программирование, 3-е издание. Пер. с англ. СПб. : ООО “Диалектика”, 2019. — 464 с.: ил. — Парал. тит. англ.
10. Пайлон Д. UML 2 для программистов. Издательство Питер, 2012.

Приложение А
Техническое задание

1 Общие сведения

1.1 Полное и краткое наименование приложения

Полное наименование приложения – «Мобильное приложение для логистической компании».

Каткое наименование приложения – «Приложение», «Green Cargo».

2 Назначение и цели создания приложения

2.1 Назначение приложения

Основным назначением мобильного приложения является обеспечение эффективного поиска грузов и грузоперевозчиков.

2.2 Цель создания приложения

Целью создания приложения является обеспечение автоматизированного поиска грузов и грузоотправителей.

2.3 Целевая аудитория приложения

Целевая аудитория приложения представлена следующими группами пользователей:

– Грузоотправители;

– Грузоперевозчики.

2.4 Основные задачи приложения

Приложение должно предоставлять пользователям доступ к информации:

– о грузах, готовых к транспортировке;

– о машинах, готовых предоставить услуги по транспортировке грузов;

– о контактных данных грузоотправителей и грузоперевозчиков;

– о параметрах груза (вес, объем и т. д.);

– о параметрах транспорта (тип транспорта, тип кузова, и т. д.);

3 Требование к приложению

3.1 Требования к функциональным характеристикам

Приложение должно обеспечивать возможность выполнения перечисленных ниже функций:

– авторизацию пользователей в приложении через e-mail;

– поиск объявлений о грузах, подлежащих к транспортировке;

– поиск машин, предоставляющие услуги транспортировки;

– создание объявлений о грузах;

– создание объявлений о транспортах;

– своевременное удаление объявлений о грузах и транспортах;

– возможность получения более подробной информации о грузах, которая включает в себе: информация о пунктах отправки и прибытия, информацию об оплате, информацию о требования к транспортировке, информацию о габаритах груза;

– возможность получения более подробной информации о транспорте, которая включает в себе: информацию о водителе транспорта (Ф. И. О, стаж работы и возраст), информацию о характеристиках транспорта;

– возможность вести поиск грузов по критериям (по пункту отправки и доставки);

– возможность вести поиск машин по аналогичным критериям.

3.2 Требования к интерфейсу приложения

– Дизайн интерфейса приложения должен соответствовать паттернам Material Design;

– дизайн интерфейса приложения должен быть консервативным;

– дизайн интерфейса приложения должен использовать неброские пастельные цвета и тона;

– дизайн интерфейса приложения должен быть общительным;

– дизайн интерфейса приложения должен быть лаконичным и в то же время выглядеть «дорого», стильно, современно.

3.3 Требования к надежности

Для надёжного функционирования приложения необходимо:

– обеспечить жёсткий контроль входных данных;

– проверять корректность вводимых данных.

3.4 Требования к программной совместимости

Программный продукт должен работать под управлением операционной системы семейства Android 5.5 и выше.

3.5 Требования к транспортировке

Носителем информации для хранения программного приложения является компакт-диск и флэш накопители. Требования к транспортировке соответствуют требованиям к транспортировке данных носителей.

3.6 Специальные требования

Не предъявляются.

4 Требования к программной документации

Исходный текст программы по ГОСТ 19.401-78.
Руководство программиста по ГОСТ 19.504-79.

5 Технико-экономические показатели

Ориентировочная экономическая эффективность от внедрения приложения – 1 581 036 тенге.

Предполагаемая стоимость внедрения приложения – 492 384 тг.

6 Стадии и этапы разработки приложения

– Разработка и программная реализация алгоритмов и структур данных;
– отладка и тестирование программных модулей;
– разработка тестов и контрольно-демонстрационной задачи;
– разработка документации по специальной и технологической части;
– разработка документации по БЖД и экономической части дипломного проекта;
– оформление дипломного проекта.

7 Порядок контроля и приемки

Разработать набор тестов и провести с их помощью испытания по требованиям описанных в разделе 3.

Проверить функциональные характеристики: согласно их списку. Интерфейс должен работать в пределах ограничений, описанных в документации.

Приложение Б
Листинг программы

CargoActivity.java

```
package com.example.greencargo;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.design.widget.NavigationView;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageView;
import com.example.greencargo.Adapter.CargoAdapter;
import com.example.greencargo.Fragment.AboutApplicationFragment;
import com.example.greencargo.Model.Cargo;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.ChildEventListener;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import java.util.ArrayList;
import java.util.List;

public class CargoActivity extends AppCompatActivity  implements
NavigationView.OnNavigationItemSelectedListener {
  private DrawerLayout drawer;
  private ImageView search;
  private List<Cargo> cargoList = new ArrayList<>();
  private List<String> cargoKeyList = new ArrayList<>();
  private CargoAdapter cargoAdapter;
  private FirebaseDatabase database = FirebaseDatabase.getInstance();
  private FirebaseUser currentUser = FirebaseAuth.getInstance().getCurrentUser();
  private DatabaseReference databaseReference = database.getReference();
```

```
@Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_cargo);

    final RecyclerView cargoListView = findViewById(R.id.allCargo);
    final LinearLayoutManager linearLayoutManager = new
LinearLayoutManager(this);
    cargoListView.setLayoutManager(linearLayoutManager);
    cargoListView.setHasFixedSize(true);
    cargoAdapter = new CargoAdapter(this, cargoList);
    cargoListView.setAdapter(cargoAdapter);
    initToolbar();
    initNavigationView();
    initSearch();

    databaseReference.child("Cargo").addChildEventListener(new
ChildEventListener() {
      @Override
      public void onChildAdded(@NonNull DataSnapshot dataSnapshot, @Nullable
String s) {
        Cargo cargo = dataSnapshot.getValue(Cargo.class);
        cargo.setUniqueKey(dataSnapshot.getKey());
        cargoList.add(cargo);
        cargoKeyList.add(dataSnapshot.getKey());
        cargoAdapter.notifyDataSetChanged();
      }

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

      @Override
      public void onChildRemoved(@NonNull DataSnapshot dataSnapshot) {
        int index = cargoKeyList.indexOf(dataSnapshot.getKey());
        cargoList.remove(index);
        cargoKeyList.remove(index);
        cargoAdapter.notifyDataSetChanged();
      }
```

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

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

  @Override
  public boolean onNavigationItemSelected(MenuItem item) {
   int id = item.getItemId();
   if (id == R.id.nav_reference) {
    AboutApplicationFragment aboutApplicationFragment = new
AboutApplicationFragment();
    aboutApplicationFragment.show(getSupportFragmentManager(),
"AboutApplication");
   } else if (id == R.id.nav_userCargo){
    if (currentUser != null) {
     Intent intent = new Intent(CargoActivity.this, UserCargoActivity.class);
     startActivity(intent);
    }
   } else if(id == R.id.nav_exit){
    CargoActivity.this.finish();
   } else if (id == R.id.nav_allTrucks) {
    if (currentUser != null) {
     Intent intent = new Intent(CargoActivity.this, TruckActivity.class);
     startActivity(intent);
    }
   }else if (id == R.id.nav_userTrucks) {
    if (currentUser != null) {
     Intent intent = new Intent(CargoActivity.this, UserTruckActivity.class);
     startActivity(intent);
    }
   }
   drawer =  findViewById(R.id.drawer_layout);
   drawer.closeDrawer(GravityCompat.START);
   return true;
  }
```

```
@Override
public void onBackPressed() {
  drawer = findViewById(R.id.drawer_layout);
  if (drawer.isDrawerOpen(GravityCompat.START)) {
    drawer.closeDrawer(GravityCompat.START);
  } else {
    super.onBackPressed();
  }
}

@Override
public boolean onOptionsItemSelected (MenuItem item) {
  switch (item.getItemId()) {
    case android.R.id.home:
      drawer.openDrawer(GravityCompat.START);
      return true;
  }
  return super.onOptionsItemSelected(item);
}

private void initToolbar() {
  final Toolbar toolbar = findViewById(R.id.toolbar);
  toolbar.setTitle(R.string.app_name);
  toolbar.inflateMenu(R.menu.menu_toolbar);
  setSupportActionBar(toolbar);
  ActionBar actionbar = getSupportActionBar();
  actionbar.setDisplayHomeAsUpEnabled(true);
  actionbar.setHomeAsUpIndicator(R.drawable.ic_menu);
}

private void initNavigationView() {
  drawer = findViewById(R.id.drawer_layout);
  NavigationView navigationView =  findViewById(R.id.navigation);
  navigationView.setNavigationItemSelectedListener(this);
}

private void initSearch() {
  search = findViewById(R.id.iv_cargoSearch);
  search.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      Intent intent = new Intent(CargoActivity.this, SearchCargoActivity.class);
      startActivity(intent); }});}}
```

CargoInformationActivity.java

```
package com.example.greencargo;

import android.content.Intent;
import android.net.Uri;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import com.example.greencargo.Firebase.FirebaseHandler;
import com.example.greencargo.Model.Cargo;

public class CargoInformationActivity extends AppCompatActivity {
  private String cargoUniqueKey;
  private TextView textView_startLocation;
  private TextView textView_endLocation;
  private TextView textView_typeOfCharge ;
  private TextView textView_date;
  private TextView textView_price;
  private TextView textView_nameOfCargo;
  private TextView textView_volume;
  private TextView textView_weight;
  private TextView textView_phone;
  private TextView textView_typeCar;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_cargo_information);
    initToolbar();
    initTextViews();
    Button button = findViewById(R.id.callCargo);
    Intent parentIntent = getIntent();
    cargoUniqueKey = parentIntent.getStringExtra("cargoUniqueKey");
    FirebaseHandler.getCargoByKey(new FirebaseHandler.CargoCallback() {
      @Override
      public void onCargoCallback(Cargo cargo) {
        setText(cargo);
      }
```

```
} , cargoUniqueKey);
  button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      String number = textView_phone.getText().toString();
      Intent intent = new Intent(Intent.ACTION_DIAL, Uri.fromParts("tel", number,
null));
      startActivity(intent);
    }
  });
 }

 @Override
 public boolean onOptionsItemSelected (MenuItem item) {
  switch (item.getItemId()) {
   case android.R.id.home:
    this.finish();
    return true;
  }
  return super.onOptionsItemSelected(item);
 }

 private void initToolbar() {
  final Toolbar toolbar = findViewById(R.id.cargoInformationToolbar);
  toolbar.setTitle(R.string.fullInfo);
  toolbar.inflateMenu(R.menu.menu_toolbar);
  setSupportActionBar(toolbar);
  ActionBar actionbar = getSupportActionBar();
  actionbar.setDisplayHomeAsUpEnabled(true);
 }

 private void initTextViews() {
  textView_startLocation = findViewById(R.id.textView_start);
  textView_endLocation = findViewById(R.id.textView_end);
  textView_typeOfCharge = findViewById(R.id.textView_load);
  textView_date = findViewById(R.id.textView_date);
  textView_price = findViewById(R.id.textView_money);
  textView_nameOfCargo = findViewById(R.id.textView_name);
  textView_volume = findViewById(R.id.textView_volume);
  textView_weight = findViewById(R.id.textView_weight);
  textView_phone = findViewById(R.id.textView_phone);
  textView_typeCar = findViewById(R.id.textView_typeCar);
 }
```

```
  private void setText(Cargo cargo) {
    textView_startLocation.setText(cargo.getStartLocation());
    textView_endLocation.setText(cargo.getEndLocation());
    textView_typeOfCharge.setText(cargo.getTypeOfCharge());
    textView_price.setText(cargo.getPrice());
    textView_date.setText(cargo.getDate());
    textView_nameOfCargo.setText(cargo.getName());
    textView_volume.setText(cargo.getVolume());
    textView_weight.setText(cargo.getWeight());
    textView_phone.setText(cargo.getPhoneNumber());
    textView_typeCar.setText(cargo.getTypeCar());
  }
}
```

CreateCargoActivity.java

```
package com.example.greencargo;

import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import com.example.greencargo.Model.Cargo;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

public class CreateCargoActivity extends AppCompatActivity {
  private FirebaseDatabase database = FirebaseDatabase.getInstance();
  private FirebaseUser currentUser = FirebaseAuth.getInstance().getCurrentUser();
  private DatabaseReference databaseReference = database.getReference();
  private EditText editText_startLocation;
  private EditText editText_endLocation;
  private EditText editText_nameOfCargo;
  private EditText editText_weight;
  private EditText editText_volume;
  private EditText editText_price;
  private EditText editText_date;
  private EditText editText_typeOfCharging;
```

```
private EditText editText_phoneNumber;
  private EditText editText_typeCar;
  private Button button_create;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_create_cargo);
    initToolbar();
    initEditTexts();
    button_create = findViewById(R.id.button_create);
    button_create.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        final String startLocation = editText_startLocation.getText().toString().trim();
        final String endLocation = editText_endLocation.getText().toString().trim();
        final String nameOfCargo = editText_nameOfCargo.getText().toString();
        final String weight = editText_weight.getText().toString();
        final String volume = editText_volume.getText().toString();
        final String price = editText_price.getText().toString();
        final String date = editText_date.getText().toString();
        final String typeOfCharging = editText_typeOfCharging.getText().toString();
        final String phoneNumber = editText_phoneNumber.getText().toString();
        final String typeCar = editText_typeCar.getText().toString();
        final String author = currentUser.getUid();
        if (Cargo.isNotNull(nameOfCargo, startLocation, endLocation, date,
phoneNumber, typeOfCharging, price, volume, weight, typeCar)) {
          Cargo cargo = new Cargo(nameOfCargo, startLocation, endLocation, date,
phoneNumber, typeOfCharging, price, volume, weight, author, typeCar);
          cargo.setStart_end(startLocation + "_" + endLocation);
          databaseReference.child("Cargo").push().setValue(cargo);
          CreateCargoActivity.this.finish();
        }
      }
    });
  }

  @Override
  public boolean onOptionsItemSelected (MenuItem item) {
    switch (item.getItemId()) {
      case android.R.id.home:
        this.finish();
        return true;
```

```
        }
    return super.onOptionsItemSelected(item);
  }

  private void initToolbar() {
    final Toolbar toolbar = findViewById(R.id.createCargoToolbar);
    toolbar.setTitle(R.string.createCargoTitle);
    toolbar.inflateMenu(R.menu.menu_toolbar);
    setSupportActionBar(toolbar);
    ActionBar actionbar = getSupportActionBar();
    actionbar.setDisplayHomeAsUpEnabled(true);
  }

  private void initEditTexts() {
    editText_startLocation = findViewById(R.id.editText_startLocation);
    editText_endLocation = findViewById(R.id.editText_endLocation);
    editText_nameOfCargo = findViewById(R.id.editText_nameOfCargo);
    editText_weight = findViewById(R.id.editText_weight);
    editText_volume = findViewById(R.id.editText_volume);
    editText_price = findViewById(R.id.editText_price);
    editText_date = findViewById(R.id.editText_date);
    editText_typeOfCharging = findViewById(R.id.editText_typeOfCharging);
    editText_phoneNumber = findViewById(R.id.editText_phoneNumber);
    editText_typeCar = findViewById(R.id.editText_typeCar);
  }
}
```

CreateTruckActivity.java

```
package com.example.greencargo;

import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import com.example.greencargo.Model.Truck;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
```

```
public class CreateTruckActivity extends AppCompatActivity {
  private FirebaseDatabase database = FirebaseDatabase.getInstance();
  private FirebaseUser currentUser = FirebaseAuth.getInstance().getCurrentUser();
  private DatabaseReference databaseReference = database.getReference();
  private EditText editText_startLocation;
  private EditText editText_endLocation;
  private EditText editText_truckType;
  private EditText editText_truckMark;
  private EditText editText_carrying;
  private EditText editText_comment;
  private EditText editText_carcaseType;
  private EditText editText_loadType;
  private EditText editText_fullName;
  private EditText editText_age;
  private EditText editText_experience;
  private Button button_create;
  private EditText editText_phoneNumber;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_create_truck);
    initToolbar();
    initEditTexts();
    button_create = findViewById(R.id.button_createTruck);
    button_create.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        final String startLocation = editText_startLocation.getText().toString().trim();
        final String endLocation = editText_endLocation.getText().toString().trim();
        final String truckType = editText_truckType.getText().toString();
        final String truckMark = editText_truckMark.getText().toString();
        final String carrying = editText_carrying.getText().toString();
        final String carcaseType = editText_carcaseType.getText().toString();
        final String comment = editText_comment.getText().toString();
        final String loadType = editText_loadType.getText().toString();
        final String phoneNumber = editText_phoneNumber.getText().toString();
        final String fullName = editText_fullName.getText().toString();
        final String age = editText_age.getText().toString();
        final String experience = editText_experience.getText().toString();
```

```
        if (Truck.isNotNull(carrying, carcaseType, truckType, loadType,
startLocation, endLocation, truckMark, phoneNumber, comment, fullName, age,
experience)) {
        Truck truck = new Truck(currentUser.getUid(), carrying, carcaseType,
truckType, loadType, startLocation, endLocation,
                truckMark, phoneNumber, comment, fullName, age, experience);
        truck.setStart_end(startLocation + "_" + endLocation);
        databaseReference.child("Trucks").push().setValue(truck);
        CreateTruckActivity.this.finish();
       }
     }
   });
 }

 @Override
 public boolean onOptionsItemSelected (MenuItem item) {
  switch (item.getItemId()) {
    case android.R.id.home:
      this.finish();
      return true;
  }
  return super.onOptionsItemSelected(item);
 }

 private void initEditTexts() {
  editText_startLocation = findViewById(R.id.editText_truckStartLocation);
  editText_endLocation = findViewById(R.id.editText_truckEndLocation);
  editText_truckMark = findViewById(R.id.editText_truckMark);
  editText_truckType = findViewById(R.id.editText_typeTruck);
  editText_carrying = findViewById(R.id.editText_carrying);
  editText_carcaseType = findViewById(R.id.editText_carcaseType);
  editText_comment = findViewById(R.id.editText_comment);
  editText_loadType = findViewById(R.id.editText_loadType);
  editText_phoneNumber = findViewById(R.id.editText_truckPhoneNumber);
  editText_fullName = findViewById(R.id.editText_fullName);
  editText_age = findViewById(R.id.editText_age);
  editText_experience = findViewById(R.id.editText_experience);
 }

 private void initToolbar() {
  final Toolbar toolbar = findViewById(R.id.createTruckToolbar);
  toolbar.setTitle(R.string.createTruckTitle);
  toolbar.inflateMenu(R.menu.menu_toolbar);
```

```
    setSupportActionBar(toolbar);
    ActionBar actionbar = getSupportActionBar();
    actionbar.setDisplayHomeAsUpEnabled(true);
  }
}
```

LoginActivity.java

```
package com.example.greencargo;

import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

public class LoginActivity extends AppCompatActivity {
  private Button btn_registration;
  private Button btn_sig_in;
  private FirebaseAuth mAuth;
  private FirebaseAuth.AuthStateListener mAuthListener;
  private EditText ETemail;
  private EditText ETpassword;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);
    mAuth = FirebaseAuth.getInstance();
    mAuthListener = new FirebaseAuth.AuthStateListener() {
      @Override
      public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
        FirebaseUser user  = firebaseAuth.getCurrentUser();
        if (user != null) {
          Intent intent = new Intent(LoginActivity.this, CargoActivity.class);
          startActivity(intent);
```

```
        }else {
       }
     }
   };
   ETemail = findViewById(R.id.EditText_Email);
   ETpassword = findViewById(R.id.EditText_Password);
   btn_registration = findViewById(R.id.registration_button);
   btn_sig_in = findViewById(R.id.sig_in_button);
   btn_registration.setOnClickListener(new View.OnClickListener() {
     @Override
     public void onClick(View v) {
       registration(ETemail.getText().toString(), ETpassword.getText().toString());
     }
   });
   btn_sig_in.setOnClickListener(new View.OnClickListener() {
     @Override
     public void onClick(View v) {
       sign(ETemail.getText().toString(), ETpassword.getText().toString());
     }
   });
  }

  public void sign(String email, String password) {
    mAuth.signInWithEmailAndPassword(email,
password).addOnCompleteListener(this, new OnCompleteListener<AuthResult>()
{
      @Override
      public void onComplete(@NonNull Task<AuthResult> task) {
        if (task.isSuccessful()) {
          FirebaseUser user  = mAuth.getCurrentUser();
          if (user != null) {
            Intent intent = new Intent(LoginActivity.this, CargoActivity.class);
            startActivity(intent);
          }
        } else {
          Toast.makeText(LoginActivity.this, "Неверный Email или пароль",
Toast.LENGTH_SHORT).show();
        }
      }
    });
  }
```

```
  public void registration(String email, String password) {
    mAuth.createUserWithEmailAndPassword(email,
password).addOnCompleteListener(this, new OnCompleteListener<AuthResult>()
{
      @Override
      public void onComplete(@NonNull Task<AuthResult> task) {
        if (task.isSuccessful()) {
          Toast.makeText(LoginActivity.this, "Регистрция успешна",
Toast.LENGTH_SHORT).show();
        } else {
          Toast.makeText(LoginActivity.this, "Email уже существует",
Toast.LENGTH_SHORT).show();
        }
      }
    });
  }
}
```

ResultCargoActivity.java

```
package com.example.greencargo;

import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import com.example.greencargo.Adapter.CargoAdapter;
import com.example.greencargo.Model.Cargo;
import com.google.firebase.database.ChildEventListener;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.Query;
import java.util.ArrayList;
import java.util.List;
```

```
public class ResultCargoActivity extends AppCompatActivity {
  private String start;
  private String end;
  private List<Cargo> cargoList = new ArrayList<>();
  private List<String> cargoKeyList = new ArrayList<>();
  private CargoAdapter cargoAdapter;
  private FirebaseDatabase database = FirebaseDatabase.getInstance();
  private DatabaseReference databaseReference = database.getReference();
  private Query querySearch;

  @Override
  protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_cargo);

    final RecyclerView cargoListView = findViewById(R.id.allCargo);
    final LinearLayoutManager linearLayoutManager = new
LinearLayoutManager(this);
    cargoListView.setLayoutManager(linearLayoutManager);
    cargoListView.setHasFixedSize(true);
    cargoAdapter = new CargoAdapter(this, cargoList);
    cargoListView.setAdapter(cargoAdapter);
    Intent parentIntent = getIntent();
    start = parentIntent.getStringExtra("start");
    end = parentIntent.getStringExtra("end");
    if ((start.isEmpty() && end.isEmpty()) || (start.equalsIgnoreCase("откуда
угодно") && end.equalsIgnoreCase("куда угодно"))) {
      querySearch = databaseReference.child("Cargo");
    } else if ((end.isEmpty() || end.equalsIgnoreCase("куда угодно")) &&
(start.length() > 0)) {
      querySearch =
databaseReference.child("Cargo").orderByChild("startLocation").equalTo(start);
    } else if ((start.isEmpty() || start.equalsIgnoreCase("откуда угодно")) &&
(end.length() > 0)) {
      querySearch =
databaseReference.child("Cargo").orderByChild("endLocation").equalTo(end);
    } else {
      querySearch =
databaseReference.child("Cargo").orderByChild("start_end").equalTo(start + "_" +
end);
    }
```

```
    querySearch.addChildEventListener(new ChildEventListener() {
      @Override
      public void onChildAdded(@NonNull DataSnapshot dataSnapshot, @Nullable
String s) {
        Cargo cargo = dataSnapshot.getValue(Cargo.class);
        cargo.setUniqueKey(dataSnapshot.getKey());
        cargoList.add(cargo);
        cargoKeyList.add(dataSnapshot.getKey());
        cargoAdapter.notifyDataSetChanged();
      }

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

      @Override
      public void onChildRemoved(@NonNull DataSnapshot dataSnapshot) {
        int index = cargoKeyList.indexOf(dataSnapshot.getKey());
        cargoList.remove(index);
        cargoKeyList.remove(index);
        cargoAdapter.notifyDataSetChanged();
      }

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

      @Override
      public void onCancelled(@NonNull DatabaseError databaseError) {
      }
    });
    initToolbar();
  }

  @Override
  public boolean onOptionsItemSelected (MenuItem item) {
    switch (item.getItemId()) {
      case android.R.id.home:
        this.finish();
        return true;
    } return super.onOptionsItemSelected(item);}
```

```
  private void initToolbar() {
    final Toolbar toolbar = findViewById(R.id.toolbar);
    toolbar.setTitle("Грузы");
    toolbar.inflateMenu(R.menu.menu_toolbar);
    setSupportActionBar(toolbar);
    ActionBar actionbar = getSupportActionBar();
    actionbar.setDisplayHomeAsUpEnabled(true);
  }
}
```

ResultTrukActivity.java

```
package com.example.greencargo;

import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import com.example.greencargo.Adapter.TruckAdapter;
import com.example.greencargo.Model.Truck;
import com.google.firebase.database.ChildEventListener;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.Query;
import java.util.ArrayList;
import java.util.List;

public class ResultTruckActivity extends AppCompatActivity {
  private String start;
  private String end;
  private List<Truck> truckList = new ArrayList<>();
  private List<String> truckKeyList = new ArrayList<>();
  private TruckAdapter truckAdapter;
  private FirebaseDatabase database = FirebaseDatabase.getInstance();
  private DatabaseReference databaseReference = database.getReference();
  private Query querySearch;
```

```
  @Override
  protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_truck);
    final RecyclerView truckLisView = findViewById(R.id.trucks);
    final LinearLayoutManager linearLayoutManager = new
LinearLayoutManager(this);
    truckLisView.setLayoutManager(linearLayoutManager);
    truckLisView.setHasFixedSize(true);
    truckAdapter = new TruckAdapter(this, truckList);
    truckLisView.setAdapter(truckAdapter);
    Intent parentIntent = getIntent();
    start = parentIntent.getStringExtra("start");
    end = parentIntent.getStringExtra("end");
    if ((start.isEmpty() && end.isEmpty()) || (start.equalsIgnoreCase("откуда
угодно") && end.equalsIgnoreCase("куда угодно"))) {
      querySearch = databaseReference.child("Trucks");
    } else if ((end.isEmpty() || end.equalsIgnoreCase("куда угодно")) &&
(start.length() > 0)) {
      querySearch =
databaseReference.child("Trucks").orderByChild("startLocation").equalTo(start);
    } else if ((start.isEmpty() || start.equalsIgnoreCase("откуда угодно")) &&
(end.length() > 0)) {
      querySearch =
databaseReference.child("Trucks").orderByChild("endLocation").equalTo(end);
    } else {
      querySearch =
databaseReference.child("Trucks").orderByChild("start_end").equalTo(start + "_" +
end);
    }
    querySearch.addChildEventListener(new ChildEventListener() {
      @Override
      public void onChildAdded(@NonNull DataSnapshot dataSnapshot, @Nullable
String s) {
        Truck truck = dataSnapshot.getValue(Truck.class);
        truck.setUniqueKey(dataSnapshot.getKey());
        truckList.add(truck);
        truckKeyList.add(dataSnapshot.getKey());
        truckAdapter.notifyDataSetChanged();
      }
```

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

    @Override
    public void onChildRemoved(@NonNull DataSnapshot dataSnapshot) {
      int index = truckKeyList.indexOf(dataSnapshot.getKey());
      truckList.remove(index);
      truckKeyList.remove(index);
      truckAdapter.notifyDataSetChanged();
    }

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

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {
    }
   });
   initToolbar();
  }

  @Override
  public boolean onOptionsItemSelected (MenuItem item) {
   switch (item.getItemId()) {
    case android.R.id.home:
      this.finish();
      return true;
   }
   return super.onOptionsItemSelected(item);
  }

  private void initToolbar() {
   final Toolbar toolbar = findViewById(R.id.truckToolbar);
   toolbar.setTitle("Машины");
   toolbar.inflateMenu(R.menu.menu_toolbar);
   setSupportActionBar(toolbar);
   ActionBar actionbar = getSupportActionBar();
   actionbar.setDisplayHomeAsUpEnabled(true);
  }}
```

SearchCargoActivity.java

```
package com.example.greencargo;

import android.content.Intent;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class SearchCargoActivity extends AppCompatActivity {
  private EditText editText_start;
  private EditText editText_end;
  private Button button_search;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_search);
    initToolbar();
    editText_start = findViewById(R.id.editText_searchStart);
    editText_end = findViewById(R.id.editText_searchEnd);
    button_search = findViewById(R.id.search_btn);
    button_search.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        Intent intent = new Intent(SearchCargoActivity.this,
ResultCargoActivity.class);
        intent.putExtra("start", editText_start.getText().toString().trim());
        intent.putExtra("end", editText_end.getText().toString().trim());
        startActivity(intent);
      }
    });
  }

  @Override
  public boolean onOptionsItemSelected (MenuItem item) {
    switch (item.getItemId()) {
      case android.R.id.home:
        this.finish();
```

```
        return true;
    }
    return super.onOptionsItemSelected(item);
  }

  private void initToolbar() {
    final Toolbar toolbar = findViewById(R.id.searchToolbar);
    toolbar.setTitle(R.string.search);
    toolbar.inflateMenu(R.menu.menu_toolbar);
    setSupportActionBar(toolbar);
    ActionBar actionbar = getSupportActionBar();
    actionbar.setDisplayHomeAsUpEnabled(true);
  }
}
```

SearchTruckActivity.java

```
package com.example.greencargo;

import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class SearchTruckActivity extends AppCompatActivity {
  private EditText editText_start;
  private EditText editText_end;
  private Button button_search;

  @Override
  protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_search);
    initToolbar();
    editText_start = findViewById(R.id.editText_searchStart);
    editText_end = findViewById(R.id.editText_searchEnd);
    button_search = findViewById(R.id.search_btn);
    button_search.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
    public void onClick(View v) {
      Intent intent = new Intent(SearchTruckActivity.this,
ResultTruckActivity.class);
      intent.putExtra("start", editText_start.getText().toString().trim());
      intent.putExtra("end", editText_end.getText().toString().trim());
      startActivity(intent);
    }
   });
  }

  @Override
  public boolean onOptionsItemSelected (MenuItem item) {
   switch (item.getItemId()) {
    case android.R.id.home:
      this.finish();
      return true;
   }
   return super.onOptionsItemSelected(item);
  }

  private void initToolbar() {
   final Toolbar toolbar = findViewById(R.id.searchToolbar);
   toolbar.setTitle(R.string.search);
   toolbar.inflateMenu(R.menu.menu_toolbar);
   setSupportActionBar(toolbar);
   ActionBar actionbar = getSupportActionBar();
   actionbar.setDisplayHomeAsUpEnabled(true);
  }
}
```

TruckInformationActivity.java

```
package com.example.greencargo;

import android.content.Intent;
import android.net.Uri;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
```

```
import android.widget.TextView;
import com.example.greencargo.Firebase.FirebaseHandler;
import com.example.greencargo.Model.Truck;

public class TruckInformationActivity extends AppCompatActivity {
  private String truckUniqueKey;
  private Button button;
  private TextView textView_startLocation;
  private TextView textView_endLocation;
  private TextView textView_fullName;
  private TextView textView_age;
  private TextView textView_experience;
  private TextView textView_truckMark;
  private TextView textView_truckType;
  private TextView textView_carrying;
  private TextView textView_comment;
  private TextView textView_carcaseType;
  private TextView textView_loadType;
  private TextView textView_phoneNumber;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_truck_information);
    Intent parentIntent = getIntent();
    truckUniqueKey = parentIntent.getStringExtra("truckUniqueKey");
    initToolbar();
    initTextViews();
    button = findViewById(R.id.callTruck);
    FirebaseHandler.getTruckByKey(new FirebaseHandler.TruckCallback() {
      @Override
      public void onTruckCallback(Truck truck) {
        setText(truck);
      }
    }, truckUniqueKey);
    button.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        String number = textView_phoneNumber.getText().toString();
        Intent intent = new Intent(Intent.ACTION_DIAL, Uri.fromParts("tel", number,
null));
        startActivity(intent);
      }});}
```

```
@Override
public boolean onOptionsItemSelected (MenuItem item) {
  switch (item.getItemId()) {
    case android.R.id.home:
      this.finish();
      return true;
  }
  return super.onOptionsItemSelected(item);
}

private void initTextViews() {
  textView_startLocation = findViewById(R.id.truckTextView_start);
  textView_endLocation = findViewById(R.id.truckTextView_end);
  textView_fullName = findViewById(R.id.truckTextView_fullName);
  textView_age = findViewById(R.id.truckTextView_age);
  textView_experience = findViewById(R.id.truckTextView_experience);
  textView_truckMark = findViewById(R.id.truckTextView_truckMark);
  textView_truckType = findViewById(R.id.truckTextView_typeTruck);
  textView_carrying = findViewById(R.id.truckTextView_carrying);
  textView_comment = findViewById(R.id.truckTextView_comment);
  textView_carcaseType = findViewById(R.id.truckTextView_carcaseType);
  textView_loadType = findViewById(R.id.truckTextView_typeOfCharging);
  textView_phoneNumber = findViewById(R.id.truckTextView_phone);
}

private void initToolbar() {
  final Toolbar toolbar = findViewById(R.id.truckInformationToolbar);
  toolbar.setTitle(R.string.fullInfo);
  toolbar.inflateMenu(R.menu.menu_toolbar);
  setSupportActionBar(toolbar);
  ActionBar actionbar = getSupportActionBar();
  actionbar.setDisplayHomeAsUpEnabled(true);
}

private void setText(Truck truck) {
  textView_startLocation.setText(truck.getStartLocation());
  textView_endLocation.setText(truck.getEndLocation());
  textView_fullName.setText(truck.getFullName());
  textView_age.setText(truck.getAge());
  textView_experience.setText(truck.getExperience());
  textView_truckMark.setText(truck.getTruckMark());
  textView_truckType.setText(truck.getTruckType());
  textView_carrying.setText(truck.getCarrying());
```

```
    textView_comment.setText(truck.getComment());
    textView_carcaseType.setText(truck.getCarcaseType());
    textView_loadType.setText(truck.getLoadType());
    textView_phoneNumber.setText(truck.getPhoneNumber());
  }
}
```

UserCargoActivity.java

```
package com.example.greencargo;

import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.design.widget.FloatingActionButton;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import com.example.greencargo.Adapter.UserCargoAdapter;
import com.example.greencargo.Model.Cargo;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.ChildEventListener;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.Query;
import java.util.ArrayList;
import java.util.List;

public class UserCargoActivity extends AppCompatActivity {
  private List<Cargo> cargoList = new ArrayList<>();
  private List<String> userCargoKeyList = new ArrayList<>();
  private UserCargoAdapter userCargoAdapter;
  private FirebaseDatabase database = FirebaseDatabase.getInstance();
  private FirebaseUser currentUser = FirebaseAuth.getInstance().getCurrentUser();
  private DatabaseReference databaseReference = database.getReference();
```

```
 @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_user_cargo);
    final RecyclerView cargoListView = findViewById(R.id.allUserCargo);
    final LinearLayoutManager linearLayoutManager = new
LinearLayoutManager(this);
    cargoListView.setLayoutManager(linearLayoutManager);
    cargoListView.setHasFixedSize(true);
    userCargoAdapter = new UserCargoAdapter(this, cargoList);
    cargoListView.setAdapter(userCargoAdapter);
    Query query =
databaseReference.child("Cargo").orderByChild("author").equalTo(currentUser.get
Uid());
    query.addChildEventListener(new ChildEventListener()  {
      @Override
      public void onChildAdded(@NonNull DataSnapshot dataSnapshot, @Nullable
String s) {
        Cargo cargo = dataSnapshot.getValue(Cargo.class);
        cargo.setUniqueKey(dataSnapshot.getKey());
        cargoList.add(cargo);
        userCargoKeyList.add(dataSnapshot.getKey());
        userCargoAdapter.notifyDataSetChanged();
      }

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

      @Override
      public void onChildRemoved(@NonNull DataSnapshot dataSnapshot) {
        int index = userCargoKeyList.indexOf(dataSnapshot.getKey());
        cargoList.remove(index);
        userCargoKeyList.remove(index);
        userCargoAdapter.notifyDataSetChanged();
      }

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

```
    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {
    }
   });
   initToolbar();
   initFloatingActionButton();
  }

  @Override
  public boolean onOptionsItemSelected (MenuItem item) {
   switch (item.getItemId()) {
    case android.R.id.home:
     this.finish();
     return true;
   }
   return super.onOptionsItemSelected(item);
  }

  private void initToolbar() {
   final Toolbar toolbar = findViewById(R.id.cargoUserToolbar);
   toolbar.setTitle(R.string.userCargoTitle);
   toolbar.inflateMenu(R.menu.menu_toolbar);
   setSupportActionBar(toolbar);
   ActionBar actionbar = getSupportActionBar();
   actionbar.setDisplayHomeAsUpEnabled(true);
  }

  private void initFloatingActionButton () {
   final FloatingActionButton addCargoButton =
findViewById(R.id.floatButtonAdd);

   addCargoButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
     if (currentUser != null) {
      Intent intent = new Intent(UserCargoActivity.this,
CreateCargoActivity.class);
      startActivity(intent);
     }
    }
   });
  }
}
```

UserTruckActivity.java

```
package com.example.greencargo;

import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.design.widget.FloatingActionButton;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import com.example.greencargo.Adapter.UserTruckAdapter;
import com.example.greencargo.Model.Truck;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.ChildEventListener;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.Query;
import java.util.ArrayList;
import java.util.List;

public class UserTruckActivity extends AppCompatActivity {
  private FirebaseDatabase database = FirebaseDatabase.getInstance();
  private DatabaseReference databaseReference = database.getReference();
  private FirebaseUser currentUser = FirebaseAuth.getInstance().getCurrentUser();
  private List<Truck> userTruckList = new ArrayList<>();
  private List<String> userTruckKeyList = new ArrayList<>();

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_user_truck);
    final RecyclerView userTruckListView = findViewById(R.id.userTrucks);
    final LinearLayoutManager linearLayoutManager = new
LinearLayoutManager(this);
    userTruckListView.setLayoutManager(linearLayoutManager);
```

```
    userTruckListView.setHasFixedSize(true);
    final UserTruckAdapter userTruckAdapter = new UserTruckAdapter(this,
userTruckList);
    userTruckListView.setAdapter(userTruckAdapter);
    initToolbar();
    initFloatingActionButton();
    Query query =
databaseReference.child("Trucks").orderByChild("author").equalTo(currentUser.ge
tUid());
    query.addChildEventListener(new ChildEventListener()  {
      @Override
      public void onChildAdded(@NonNull DataSnapshot dataSnapshot, @Nullable
String s) {
        Truck truck = dataSnapshot.getValue(Truck.class);
        truck.setUniqueKey(dataSnapshot.getKey());
        userTruckList.add(truck);
        userTruckKeyList.add(dataSnapshot.getKey());
        userTruckAdapter.notifyDataSetChanged();
      }

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

      @Override
      public void onChildRemoved(@NonNull DataSnapshot dataSnapshot) {
        int index = userTruckKeyList.indexOf(dataSnapshot.getKey());
        userTruckList.remove(index);
        userTruckKeyList.remove(index);
        userTruckAdapter.notifyDataSetChanged();
      }

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

      @Override
      public void onCancelled(@NonNull DatabaseError databaseError) {
      }
    });
  }
```

```
  @Override
  public boolean onOptionsItemSelected (MenuItem item) {
    switch (item.getItemId()) {
      case android.R.id.home:
        this.finish();
        return true;
    }
    return super.onOptionsItemSelected(item);
  }

  private void initFloatingActionButton() {
    final FloatingActionButton addCargoButton =
findViewById(R.id.floatButtonAddTruck);

    addCargoButton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        if (currentUser != null) {
          Intent intent = new Intent(UserTruckActivity.this,
CreateTruckActivity.class);
          startActivity(intent);
        }
      }
    });
  }

  private void initToolbar() {
    final Toolbar toolbar = findViewById(R.id.userTruckToolbar);
    toolbar.setTitle(R.string.userTruckTitle);
    toolbar.inflateMenu(R.menu.menu_toolbar);
    setSupportActionBar(toolbar);
    ActionBar actionbar = getSupportActionBar();
    actionbar.setDisplayHomeAsUpEnabled(true);
  }
}
```

AboutApplicationFragment.java

```
package com.example.greencargo.Fragment;

import android.app.AlertDialog;
import android.app.Dialog;
import android.os.Bundle;
import android.support.annotation.NonNull;
```

```
import android.support.annotation.Nullable;
import android.support.v4.app.DialogFragment;
import android.view.LayoutInflater;
import com.example.greencargo.R;

public class AboutApplicationFragment extends DialogFragment {
    @NonNull
    @Override
    public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
        LayoutInflater layoutInflater = getActivity().getLayoutInflater();
        builder.setView(layoutInflater.inflate(R.layout.dialog_about_application,
null))
                .setPositiveButton(R.string.ok,null);
        return builder.create();
    }
}
```

FirebaseHandler.java

```
package com.example.greencargo.Firebase;

import android.support.annotation.NonNull;
import com.example.greencargo.Model.Cargo;
import com.example.greencargo.Model.Truck;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class FirebaseHandler {
  private static FirebaseDatabase database = FirebaseDatabase.getInstance();
  private static DatabaseReference cargoReference =
database.getReference().child("Cargo");
  private static DatabaseReference truckReference =
database.getReference().child("Trucks");

  public static void getCargoByKey(final CargoCallback firebaseCallback, final
String key) {
    ValueEventListener valueEventListener= new ValueEventListener() {
      @Override
      public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        Cargo cargo = dataSnapshot.child(key).getValue(Cargo.class);
```

```
      firebaseCallback.onCargoCallback(cargo);
     }

     @Override
     public void onCancelled(@NonNull DatabaseError databaseError) {
     }
    };
   cargoReference.addListenerForSingleValueEvent(valueEventListener);
  }

  public static void getTruckByKey(final TruckCallback truckCallback, final String
key) {
   ValueEventListener valueEventListener = new ValueEventListener() {
     @Override
     public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
      Truck truck = dataSnapshot.child(key).getValue(Truck.class);
      truckCallback.onTruckCallback(truck);
     }

     @Override
     public void onCancelled(@NonNull DatabaseError databaseError) {
     }
    };
   truckReference.addListenerForSingleValueEvent(valueEventListener);
  }

  public interface CargoCallback {
   void onCargoCallback(Cargo cargo);
  }

  public interface TruckCallback {
   void onTruckCallback(Truck truck);
  }
}
```

CargoAdapter.java

```
package com.example.greencargo.Adapter;

import android.content.Context;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
```

```
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import com.example.greencargo.CargoInformationActivity;
import com.example.greencargo.Model.Cargo;
import com.example.greencargo.R;
import java.util.List;

public class CargoAdapter extends
RecyclerView.Adapter<CargoAdapter.CargoViewHolder>{
   private Context context;
   private List<Cargo> cargoArrayList;

   public  CargoAdapter(Context context, List<Cargo> cargoArrayList) {
      this.context = context;
      this.cargoArrayList = cargoArrayList;
   }

   @NonNull
   @Override
   public CargoViewHolder onCreateViewHolder(@NonNull ViewGroup
viewGroup, int i) {
      View view = LayoutInflater.from(context).inflate(R.layout.cargo_item,
viewGroup, false);
      return new CargoViewHolder(view);
   }

   @Override
   public void onBindViewHolder(@NonNull final CargoViewHolder
cargoViewHolder, int i) {
      cargoViewHolder.name.setText(cargoArrayList.get(i).getName());
      cargoViewHolder.price.setText(cargoArrayList.get(i).getPrice());
cargoViewHolder.endLocation.setText(cargoArrayList.get(i).getEndLocation());

cargoViewHolder.startLocation.setText(cargoArrayList.get(i).getStartLocation());
      cargoViewHolder.date.setText(cargoArrayList.get(i).getDate());
      cargoViewHolder.itemView.setOnClickListener(new View.OnClickListener()
{
         @Override
         public void onClick(View v) {
            Cargo cargo =
cargoArrayList.get(cargoViewHolder.getAdapterPosition());
            String uniqueKey = cargo.getUniqueKey();
```

```
            Intent intent = new Intent(context, CargoInformationActivity.class);
            intent.putExtra("cargoUniqueKey", uniqueKey);
            context.startActivity(intent);
        }
    });
}

@Override
public int getItemCount() {
    return cargoArrayList.size();
}

public class CargoViewHolder extends RecyclerView.ViewHolder {
    private TextView startLocation;
    private TextView endLocation;
    private TextView date;
    private TextView name;
    private TextView price;

    public CargoViewHolder(@NonNull View itemView) {
        super(itemView);
        startLocation = itemView.findViewById(R.id.startLocation);
        endLocation = itemView.findViewById(R.id.endLocation);
        name = itemView.findViewById(R.id.name);
        date = itemView.findViewById(R.id.date);
        price = itemView.findViewById(R.id.price);
    }
}
}
```

TruckAdapter.java

```
package com.example.greencargo.Adapter;

import android.content.Context;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import com.example.greencargo.TruckInformationActivity;
import com.example.greencargo.Model.Truck;
```

```
import com.example.greencargo.R;
import java.util.List;

public class TruckAdapter extends
RecyclerView.Adapter<TruckAdapter.TruckViewHolder>{
  private Context context;
  private List<Truck> truckList;

  public TruckAdapter(Context context, List<Truck> truckList) {
    this.context = context;
    this.truckList = truckList;
  }

  @NonNull
  @Override
  public TruckViewHolder onCreateViewHolder(@NonNull ViewGroup
viewGroup, int i) {
    View view = LayoutInflater.from(context).inflate(R.layout.truck_item,
viewGroup, false);
    return new TruckViewHolder(view);
  }

  @Override
  public void onBindViewHolder(@NonNull final TruckViewHolder
truckViewHolder, int i) {
    truckViewHolder.carcaseType.setText(truckList.get(i).getCarcaseType());
    truckViewHolder.carrying.setText(truckList.get(i).getCarrying());
    truckViewHolder.truckType.setText(truckList.get(i).getTruckType());
    truckViewHolder.truckMark.setText(truckList.get(i).getTruckMark());
    truckViewHolder.itemView.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        Truck truck = truckList.get(truckViewHolder.getAdapterPosition());
        String uniqueKey = truck.getUniqueKey();
        Intent intent = new Intent(context, TruckInformationActivity.class);
        intent.putExtra("truckUniqueKey", uniqueKey);
        context.startActivity(intent);
      }
    });
  }
```

```
 @Override
  public int getItemCount() {
    return truckList.size();}

  public class TruckViewHolder extends RecyclerView.ViewHolder {
    private TextView truckMark;
    private TextView truckType;
    private TextView carrying;
    private TextView carcaseType;

    public TruckViewHolder(@NonNull View itemView) {
      super(itemView);
      truckMark = itemView.findViewById(R.id.textView_truckMark);
      truckType = itemView.findViewById(R.id.textView_truckType);
      carrying = itemView.findViewById(R.id.textView_carrying);
      carcaseType = itemView.findViewById(R.id.textView_carcaseType);
    }
  }
}
```

UserCargoAdapter.java

```
package com.example.greencargo.Adapter;

import android.content.Context;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.PopupMenu;
import android.widget.TextView;
import com.example.greencargo.CargoInformationActivity;
import com.example.greencargo.Model.Cargo;
import com.example.greencargo.R;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import java.util.List;

public class UserCargoAdapter extends
RecyclerView.Adapter<UserCargoAdapter.UserCargoViewHolder>{
```

```
  private FirebaseDatabase database = FirebaseDatabase.getInstance();
  private DatabaseReference databaseReference = database.getReference();
  private Context context;
  private List<Cargo> cargoList;

  public UserCargoAdapter (Context context, List<Cargo> cargoList) {
    this.context = context;
    this.cargoList = cargoList;
  }

  @NonNull
  @Override
  public UserCargoViewHolder onCreateViewHolder(@NonNull ViewGroup
viewGroup, int i) {
    View view = LayoutInflater.from(context).inflate(R.layout.user_cargo_item,
viewGroup, false);
    return new UserCargoAdapter.UserCargoViewHolder(view);
  }

  @Override
  public void onBindViewHolder(@NonNull final UserCargoViewHolder
userCargoViewHolder, int i) {
    userCargoViewHolder.name.setText(cargoList.get(i).getName());
    userCargoViewHolder.price.setText(cargoList.get(i).getPrice());
    userCargoViewHolder.endLocation.setText(cargoList.get(i).getEndLocation());
    userCargoViewHolder.startLocation.setText(cargoList.get(i).getStartLocation());
    userCargoViewHolder.date.setText(cargoList.get(i).getDate());
    userCargoViewHolder.itemView.setOnClickListener(new
View.OnClickListener() {
      @Override
      public void onClick(View v) {
        Cargo cargo = cargoList.get(userCargoViewHolder.getAdapterPosition());
        String uniqueKey = cargo.getUniqueKey();
        Intent intent = new Intent(context, CargoInformationActivity.class);
        intent.putExtra("cargoUniqueKey", uniqueKey);
        context.startActivity(intent);
      }
    });
    userCargoViewHolder.popupMenu.setOnClickListener(new
View.OnClickListener() {
      @Override
      public void onClick(View v) {
```

```
        final Cargo cargo =
cargoList.get(userCargoViewHolder.getAdapterPosition());
        final String uniqueKey = cargo.getUniqueKey();
        PopupMenu popupMenu = new PopupMenu(context,
userCargoViewHolder.popupMenu);
        popupMenu.inflate(R.menu.menu_popup);
        popupMenu.setOnMenuItemClickListener(new
PopupMenu.OnMenuItemClickListener() {
          @Override
          public boolean onMenuItemClick(MenuItem item) {
            switch (item.getItemId()) {
              case (R.id.delete):
                databaseReference.child("Cargo").child(uniqueKey).removeValue();
            }
            return false;
          }
        });
        popupMenu.show();
      }
    });
  }

  @Override
  public int getItemCount() {
    return cargoList.size();
  }

  public class UserCargoViewHolder extends RecyclerView.ViewHolder {
    private TextView startLocation;
    private TextView endLocation;
    private TextView date;
    private TextView name;
    private TextView price;
    private ImageView popupMenu;

    public UserCargoViewHolder(@NonNull View itemView) {
      super(itemView);
      startLocation = itemView.findViewById(R.id.userStartLocation);
      endLocation = itemView.findViewById(R.id.userEndLocation);
      date = itemView.findViewById(R.id.userDate0fCharging);
      name = itemView.findViewById(R.id.userName);
      price = itemView.findViewById(R.id.userPrice);
      popupMenu = itemView.findViewById(R.id.iv_cargoPopupMenu); }}}
```

UserTruckAdapter.java

```
package com.example.greencargo.Adapter;

import android.content.Context;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.PopupMenu;
import android.widget.TextView;
import com.example.greencargo.TruckInformationActivity;
import com.example.greencargo.Model.Truck;
import com.example.greencargo.R;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import java.util.List;

public class UserTruckAdapter extends
RecyclerView.Adapter<UserTruckAdapter.UserTruckViewHolder>{
  private FirebaseDatabase database = FirebaseDatabase.getInstance();
  private DatabaseReference databaseReference = database.getReference();
  private Context context;
  private List<Truck> truckList;

  public UserTruckAdapter(Context context, List<Truck> truckList) {
    this.context = context;
    this.truckList = truckList;
  }

  @NonNull
  @Override
  public UserTruckViewHolder onCreateViewHolder(@NonNull ViewGroup
viewGroup, int i) {
    View view = LayoutInflater.from(context).inflate(R.layout.user_truck_item,
viewGroup, false);
    return new UserTruckAdapter.UserTruckViewHolder(view);
  }
```

```
  @Override
  public void onBindViewHolder(@NonNull final UserTruckViewHolder
userTruckViewHolder, int i) {
    userTruckViewHolder.carcaseType.setText(truckList.get(i).getCarcaseType());
    userTruckViewHolder.carrying.setText(truckList.get(i).getCarrying());
    userTruckViewHolder.truckType.setText(truckList.get(i).getTruckType());
    userTruckViewHolder.truckMark.setText(truckList.get(i).getTruckMark());
    userTruckViewHolder.itemView.setOnClickListener(new
View.OnClickListener() {
      @Override
      public void onClick(View v) {
        final Truck truck = truckList.get(userTruckViewHolder.getAdapterPosition());
        String uniqueKey = truck.getUniqueKey();
        Intent intent = new Intent(context, TruckInformationActivity.class);
        intent.putExtra("truckUniqueKey", uniqueKey);
        context.startActivity(intent);
      }
    });
    userTruckViewHolder.popupMenu.setOnClickListener(new
View.OnClickListener() {
      @Override
      public void onClick(View v) {
        final Truck truck = truckList.get(userTruckViewHolder.getAdapterPosition());
        final String uniqueKey = truck.getUniqueKey();
        PopupMenu popupMenu = new PopupMenu(context,
userTruckViewHolder.popupMenu);
        popupMenu.inflate(R.menu.menu_popup);
        popupMenu.setOnMenuItemClickListener(new
PopupMenu.OnMenuItemClickListener() {
          @Override
          public boolean onMenuItemClick(MenuItem item) {
            switch (item.getItemId()) {
              case  R.id.delete:
              databaseReference.child("Trucks").child(uniqueKey).removeValue();
              break;
            }
            return false;
          }
        });
        popupMenu.show();
      }
    });
  }
```

```
 @Override
 public int getItemCount() {
  return truckList.size();
 }

 public class UserTruckViewHolder extends RecyclerView.ViewHolder {
  private TextView truckMark;
  private TextView truckType;
  private TextView carrying;
  private TextView carcaseType;
  private ImageView popupMenu;

  public UserTruckViewHolder(@NonNull View itemView) {
   super(itemView);
   truckMark = itemView.findViewById(R.id.textView_userTruckMark);
   truckType = itemView.findViewById(R.id.textView_userTruckType);
   carrying = itemView.findViewById(R.id.textView_userCarrying);
   carcaseType = itemView.findViewById(R.id.textView_userCarcaseType);
   popupMenu = itemView.findViewById(R.id.iv_popupMenu);
  }
 }
}
```