Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись
« _____» _____ 2019 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 – «Информатика и вычислительная техника»

Мобильный помощник автовладельца

Руководитель	_____ подпись, дата	ст. преподаватель	К.В. Пушкарев
Выпускник	_____ подпись, дата		Р.У. Исаев
Консультант	_____ подпись, дата	канд. техн. наук, доцент	Л.И. Покидышева
Нормоконтролер	_____ подпись, дата		В.И. Иванов

Красноярск 2019

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Автомобиль уже давно стал частью нашей жизни. Сейчас уже почти каждый имеет свой автомобиль или мечтает о нем. Они стали частью окружающей действительности, сейчас мы многое не сможем делать без машин. То же самое можно сказать о смартфоне. Исходя из этих факторов, есть множество приложений для автолюбителей на любой вкус. Их функционал разнообразен: от простого навигатора до полупрофессионального приложения для считывания ошибок с головного устройства автомобиля.

Примеры существующих приложений:

- навигатор, отслеживающий дорожные события (аварии, дорожные работы, пробки и т.д.) и перестраивающий маршрут, исходя из дорожной обстановки;

- приложение, в котором есть: одометр, компас и спидометр, предупреждающий о превышении скоростного режима;

- приложение, предупреждающее о камерах контроля скорости;

- приложение, позволяющее сделать диагностику автомобиля;

- приложение, позволяющее следить за расходами и своевременно обслуживать свое транспортное средство.

Целью выпускной квалификационной работы является разработка программного обеспечения для смартфонов под управлением операционной системы Android. Оно служит для удобного учёта и анализа расходов на автомобиль. Программа должна обладать следующими особенностями:

- распределение расходов на содержание одного или нескольких автомобилей;

- напоминание о связанных с автомобилем событиях (плановом техническом обслуживании и т. д.);

- работа без доступа в Интернет;

- синхронизация с удаленной базой данных через Интернет.

Для достижения цели в работе решаются следующие задачи:

- анализ задания на выпускную квалификационную работу;
- проектирование;
- реализация приложения;
- составление инструкций.

1 Анализ задания на выпускную квалификационную работу

В соответствии с заданием необходимо разработать приложение «мобильный помощник автовладельца» под операционную систему Android. Чтобы выполнить требования задания, был выполнен анализ аналогов, при этом отмечены их сильные и слабые стороны. В ходе анализа не было обнаружено готовых решений с открытым исходным кодом.

1.1 Обзор существующих решений

На сегодняшний день существует ряд русифицированных, качественных и многофункциональных мобильных помощников для автовладельцев, которые способны вести учет расходов на ремонт, бензин, мойку и т. д., и при этом анализировать полученные данные и выводить их в удобном, читабельном варианте.

Подобные бесплатные системы направлены только на учет расхода топлива, более функциональные системы являются платными. Ниже представлена таблица с основными сведениями бесплатных версий приложений.

Таблица 1 – Основные сведения о приложениях

Название	Поддержка русского языка	Возможность синхронизации с удаленной БД	Визуализация данных (графики, схемы, диаграммы)	Напоминание о плановом ТО	Максимальное кол-во автомобилей
My Car	-	-	-	-	5
My Cars	-	-	+	+	3
Авто Расходы	+	-	-	-	2
Моя Машина	+	-	-	+	3

1.1.1 My Car

My Car – это приложение, которое позволяет управлять расходами ваших автомобилей [1].

My Car распространяется бесплатно и не имеет платного функционала (открытие возможности импорта/экспорта данных, дополнительный функционал).

Данное приложение имеет 3 основных окна:

- traffic – список всех расходов на автомобиль, а также средний расход топлива (рис. 1);

- car – список добавленных автомобилей (рис. 2);

- profile – окно, где указывается общий расход средств на весь автопарк, а также список заправок, где происходила заправка.

Недостатки данного приложения:

- скудный функционал, который дает возможность только добавить пройденное расстояние и количество залитого бензина и добавить прочие расходы, которые никак не фильтруются;

- отсутствие русскоязычного интерфейса;

- отсутствует возможность сортировки расходов (по дате, по среднему расходу, по потраченной сумме);

- отсутствуют техническая характеристики автомобиля (объем двигателя, мощность и т.д.);

- последнее обновление вышло в 2015 году (не исправляются ошибки, интерфейс на современных телефонах отображается неправильно).

Данное приложение предназначено, исключительно, для просмотра среднего расхода топлива на 100 км, но с данной возможностью и так справляется любой бортовой компьютер автомобиля.

Рисунок 1 – Окно Traffic

Рисунок 2 – Окно Car

1.1.2 My Cars

My Cars – включает в себя:

- журнал расхода топлива;
- учет расходов;
- записи о техобслуживании и многое другое [2].

Интерфейс My Cars изображен на рисунке 3.

Данное приложение предлагает множество возможностей: начиная от полного слежения за расходом топлива, полностью настраиваемыми записями о ТО, заканчивая массой статистик, графиков и записей.

У данного приложения можно выделить несколько проблем: недружелюбный дизайн, отсутствие русскоязычного интерфейса, отсутствие возможности добавить напоминание, о чем-либо, отсутствие экспорта/импорта данных.

Рисунок 3 – Интерфейс My Cars

1.1.3 «Авто Расходы – Менеджер автомобиля»

Это мобильная сервисная книжка автомобиля, содержащая все необходимое: все номера документов, учет расходов, налоги на авто и штрафы ГИБДД [3].

Интерфейс «Авто Расходы» изображен на рисунке 4.

Данное приложение – бесплатная версия платного приложения. В ней присутствует возможность добавлять свои расходы, сортируя по 4 основным категориям («Мойка», «Заправка», «ТО», «Парковка»), а также просмотра графиков по данным. Остальной же функционал сводится к оплате автокредита, оплате штрафов ГИБДД, оформлению страховки и подбору авто.

Рисунок 4 – Интерфейс «Авто Расходы»

1.1.4 «Моя Машина – Авто Расходы»

В приложении «Моя Машина» реализована возможность вести учет расхода топлива, а также учет затрат на автомобиль [4].

Интерфейс «Моя Машина» изображен на рисунке 5.

Из недостатков данного приложения, хотелось бы отметить: расходы на запчасти и услуги сервиса приходится объединять в одном разделе, отсутствие графиков, неудобное окно добавления напоминания, отсутствие напоминания о ключевых событиях.

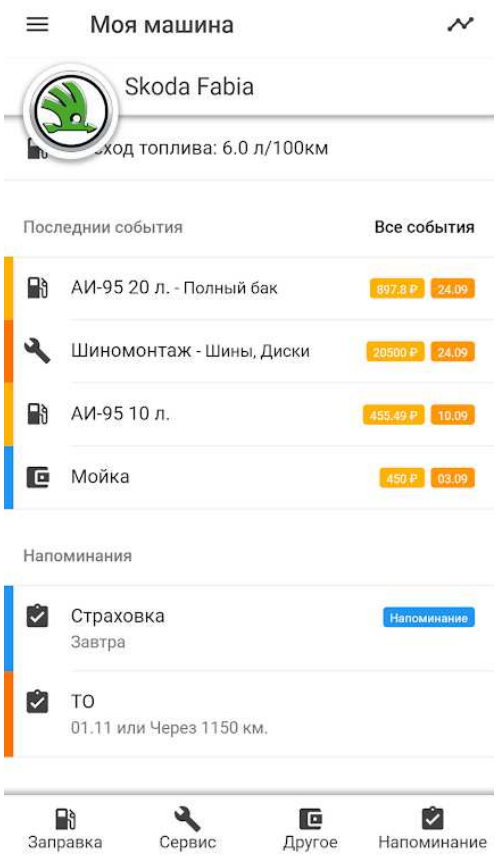

Рисунок 5 – Интерфейс «Моя Машина»

1.1.5 Итоги обзора

Из обзора существующих решений, представленных в пунктах 1.1.1-1.1.4, можно сделать вывод, что все рассмотренные аналоги предназначены для учета расходов автомобиля и упрощения жизни автовладельца.

Каждое из рассмотренных решений позволяет вести учет расхода топлива, а также общий учет расходов. Также во многих решениях присутствует возможность просмотра графиков и диаграмм по существующим данным, кроме My Car.

Из четырех приложений в «Моя Машина» лучше всего выглядит визуальная часть интерфейса, но имеет множество недостатков. А приложение My Cars имеет множество возможностей, но отсутствует экспорт/импорт данных в бесплатной версии приложения.

В My Car, в отличие от остальных программ, отсутствует возможность просмотра графиков и диаграмм, что доставляет неудобство при большом количестве имеющейся информации.

Приложение «Авто Расходы» направлено больше на финансовое удобство в плане оплат автокредитов, страховки, штрафов, но не на удобство контроля своих расходов на автомобиль и своевременных проведений ТО.

Проанализировав приложения, можно сделать вывод, о том, что ни одно из рассмотренных приложений не позволяет обеспечить эффективное выполнение задач автовладельца:

- напоминания о событиях: плановое ТО, ремонт, страховка;
- расчёт расхода топлива и стоимости 1 км пути;
- распределение расходов на заправку, мойку, ремонт, обслуживание и многое другое;
- возможность выгрузки данных, синхронизация с удалённой БД.

Все эти функции необходимы в мобильном помощнике автовладельца.

1.2 Интерфейс приложения

На основе технического задания был сделан макет интерфейса, который должен получиться в конечном приложении. На рисунке 6 изображен этот макет.

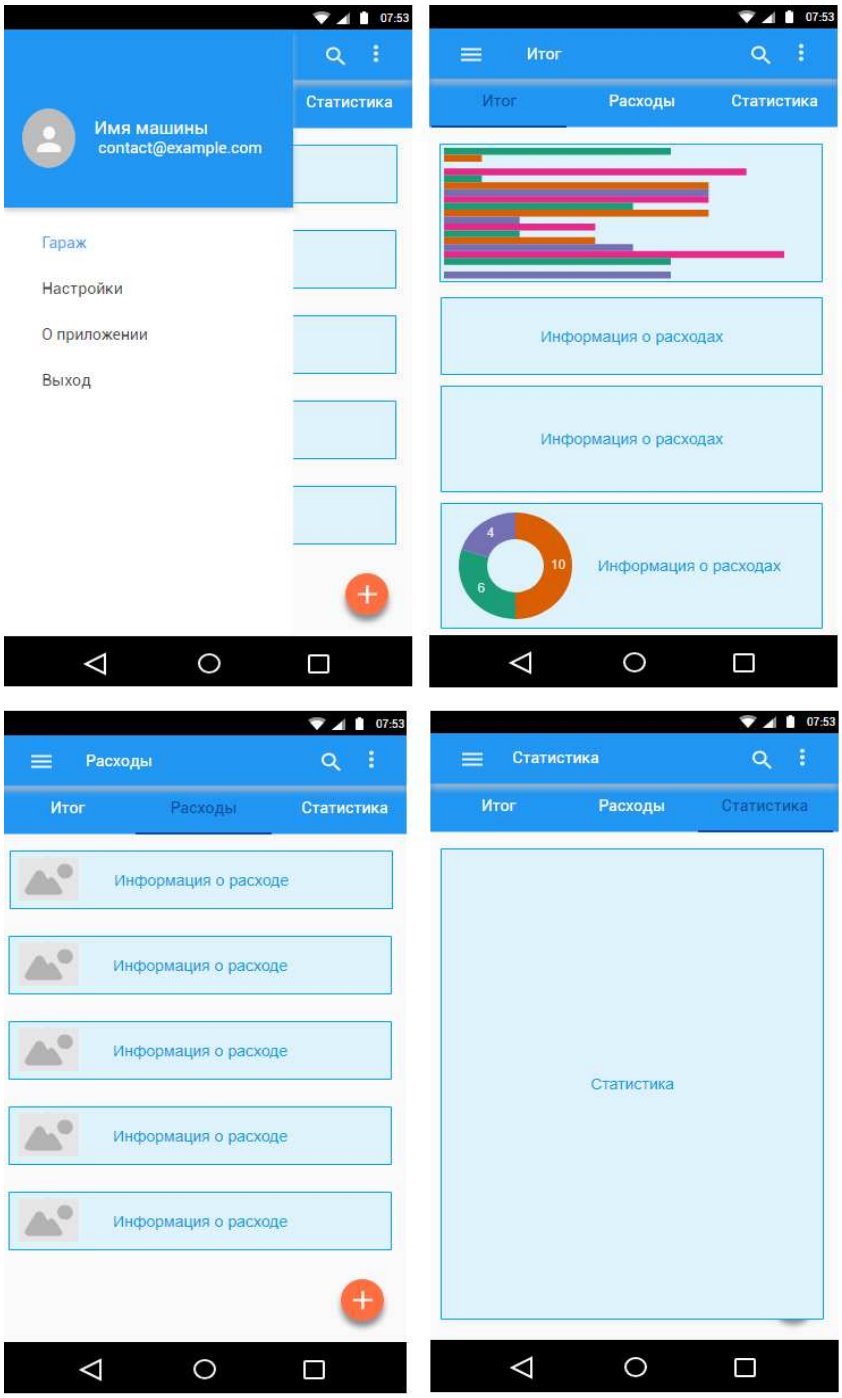

Рисунок 6 – Макет интерфейса

1.2.1 Диаграмма вариантов использования

На рисунке 7 изображена диаграмма вариантов использования.

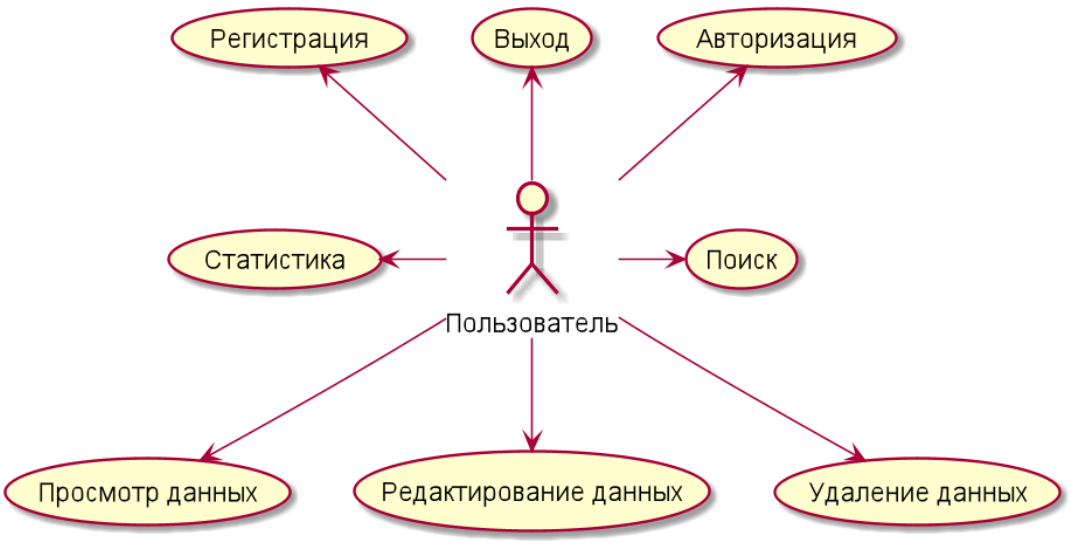

Рисунок 7 – Диаграмма вариантов использования

Текстовое описание вариантов использования:

Название прецедента: Регистрация.

Цель сценария: добавить пользователя в базу данных для дальнейшей синхронизации его расходов.

Предусловия: пользователь находится в окне авторизации и нажимает кнопку регистрация, после чего вводит данные и нажимает кнопку «Зарегистрироваться».

Основной сценарий:

A. Данные проходят проверку на корректность, после чего отправляются на сервер и проходят там проверку на наличие данных в базе.

B. Данные добавляются в базу данных.

Постусловия: пользователь может войти в систему используя регистрационные данные.

Условие ввода в действие альтернативных сценариев

Условие 1. Отсутствует интернет соединение.

А. Приложение выдает сообщение об отсутствии интернет соединения.

Условие 2. Данные введены не корректно.

А. Приложение выдает сообщение об некорректности введенных данных.

Условие 3. Пользователь с такими данными уже зарегистрирован.

А. Приложение выдает сообщение о существовании таких данных в базе.

Название прецедента: Авторизация

Цель сценария: проверить данные на корректность и наличие в базе, и открыть главное окно.

Предусловия: пользователь находится на главном экране, вводит свои данные и нажимает кнопку «Войти».

Основной сценарий:

А. Данные проходят проверку на корректность, после чего отправляются на сервер и проходят там проверку на наличие данных в базе.

Постусловия: пользователь попадает в главное окно.

Условие ввода в действие альтернативных сценариев

Условие 1. Отсутствует интернет соединение.

А. Приложение выдает сообщение об отсутствии интернет соединения.

Условие 2. Данные введены не корректно.

А. Приложение выдает сообщение об некорректности введенных данных.

Условие 3. Пользователь с такими данными не существует.

В. Приложение выдает сообщение отсутствии данных в базе.

Условие 4. Пользователь ввел неправильный пароль.

A. Приложение выдает сообщение о неправильности пароля.

Название прецедента: Выход.

Цель сценария: удалить данные авторизации из памяти и открыть окно авторизации.

Предусловия: пользователь находится на главном экране и нажимает кнопку «Выйти».

Основной сценарий:

A. Данные об авторизации удаляются из памяти, после чего пользователь перенаправляется в окно авторизации.

Постусловия: пользователь попадает в окно авторизации.

Название прецедента: Поиск.

Цель сценария: поиск по базе данных расходов по примечанию, дате или типу.

Предусловия: пользователь находится на главном экране, нажимает на иконку лупы и вводит текст.

Основной сценарий:

A. Происходит поиск по базе данных.

Постусловия: пользователь видит полученные результаты на экране.

Название прецедента: Удаление данных.

Цель сценария: удаление данных.

Предусловия: пользователь, находясь в главном окне, открывает либо гараж, либо список расходов и нажимает кнопку «удалить».

Основной сценарий:

A. Происходит обновление списка исходя из изменений.

B. Происходит синхронизация с онлайн базой данных.

Постусловия: данные обновляются в базе данных исходя из полученных изменений.

Название прецедента: Редактирование данных.

Цель сценария: редактирование.

Предусловия: пользователь, находясь в главном окне, открывает либо гараж, либо список расходов и нажимает кнопку «редактировать».

Основной сценарий:

A. Происходит обновление списка исходя из изменений.

B. Происходит синхронизация с онлайн базой данных.

Постусловия: данные обновляются в базе данных исходя из полученных изменений.

Название прецедента: Просмотр данных.

Цель сценария: просмотр данных об авто или расходе.

Предусловия: пользователь, находясь в главном окне, открывает либо гараж, либо список расходов и нажимает на элемент списка.

Основной сценарий:

A. Регистрируется нажатие на элемент из списка и указывается позиция.

B. Происходит поиск информации в базе данных по данной позиции.

C. Выводится информация.

Постусловия: вывод информации.

Название прецедента: Статистика.

Цель сценария: просмотр результата обработки имеющихся данных по расходам в базе.

Предусловия: пользователь, находясь в главном окне нажимает кнопку «Статистика».

Основной сценарий:

A. Происходит анализ имеющихся данных.

B. Выводится статистика на экран.

Постусловия: просмотр результата обработки имеющихся данных.

1.3 Выбор инструментов

1.3.1 Выбор языка программирования

Хотя Java является официальным языком для Android-приложений, существует множество других языков программирования для Andorid-приложений. С каждым языком связаны свои сложности и нюансы, достоинства и недостатки. Рассмотрим основные языки для написания Android-приложений.

1.3.1.1 Java

Язык Java впервые появился 23 мая 1995 года. Java является официальным языком для разработки приложений для Android и, следовательно, он также является наиболее используемым языком [5]. Многие приложения в Play Store написаны на Java, а также это самый поддерживаемый язык Google.

Тем не менее, Java является сложным языком для новичка, поскольку он содержит сложные темы, такие как конструкторы классов, исключения нулевого указателя, параллелизм, проверенные исключения и так далее. Кроме того, пакет разработчика программного обеспечения Android (SDK) повышает сложность программирования на нем, за счет добавления нового материала. При соблюдении принятых стандартов оформления код на Java легко читается и структурируется.

В общем, Java – отличный язык, для разработки приложений под ОС Android. Благодаря своим надежным функциям, таким как объектно-ориентированное программирование (ООП), независимость от платформы, многопоточность, сборка мусора и обработка исключений, язык завоевал высокую популярность.

1.3.1.2 Kotlin

Kotlin – это кроссплатформенный язык программирования, который можно использовать в качестве альтернативы Java для разработки приложений для Android. Он также был представлен как дополнительный «официальный» язык Java в 2017 году [6]. Kotlin может взаимодействовать с Java и работает на виртуальной машине Java.

Одно из существенных отличий состоит в том, что Kotlin удаляет лишние функции Java, такие, как исключения нулевого указателя. Это также устраняет необходимость заканчивать каждую строку точкой с запятой. Kotlin гораздо проще для начинающих по сравнению с Java, и его также можно использовать в качестве «точки входа» для разработки приложений для Android. Также Kotlin совместим с Java и не вызывает снижения производительности.

1.3.1.3 C/C++

C++ может быть использован для разработки приложений для Android с использованием Native Development Kit (NDK) для Android [7]. Однако приложение не может быть полностью создано с использованием C++, а NDK используется для реализации частей приложения на C++. Это помогает в использовании библиотек, написанных на C++ для приложения.

Хотя C++ полезен для разработки приложений для Android в некоторых случаях, он гораздо менее гибок. Это также может привести к большему количеству ошибок из-за повышенной сложности. Таким образом, лучше использовать Java по сравнению с C++.

1.3.1.4 C#

C# очень похож на Java и поэтому идеально подходит для разработки приложений для Android. Как и Java, C# также реализует сборку мусора, поэтому вероятность утечки памяти меньше. И C# также имеет более чистый и простой синтаксис, по сравнению с Java, что делает программирование на нем значительно проще.

Ранее самым большим недостатком C# было то, что он мог работать только в системах Windows, поскольку он использовал .NET Framework [8]. Однако эта проблема была решена Xamarin.Android (ранее Mono для Android), который является кроссплатформенной реализацией Common Language Infrastructure. Теперь инструменты Xamarin.Android можно использовать для написания собственных приложений для Android и совместного использования кода на нескольких платформах.

1.3.1.5 Python

Python можно использовать для разработки приложений для Android, даже если Android не поддерживает разработку на Python. Это можно сделать с помощью различных инструментов, преобразующих приложения Python в Android Packages, которые могут работать на устройствах Android.

Примером этого является Kivy, библиотека Python с открытым исходным кодом, используемая для разработки мобильных приложений. Он поддерживает Android, а также способствует быстрой разработке приложений.

1.3.1.6 Web языки

Приложения Android могут быть созданы с использованием HTML, CSS и JavaScript с использованием платформы Adobe PhoneGap, работающей на Apache Cordova. Платформа PhoneGap в основном позволяет использовать

навыки веб-разработки для создания гибридных приложений, которые отображаются через «WebView», но упакованы как приложения.

1.3.1.8 Итог

Писать приложения под ОС Android можно практически с помощью любого популярного языка, утилиты и фреймворки можно найти под все. Однако, чтобы воспользоваться всеми возможностями операционной системы, иметь доступ к последним функциям Android – верными спутниками станут Kotlin или Java. Данные языки подходят для написания красивых, разнообразных и функциональных приложений под ОС Android.

У Java и Kotlin есть свои преимущества и недостатки. У Kotlin маленькое сообщество разработчиков, в связи с чем количество ресурсов для изучения языка ограничено. Java, в свою очередь, имеет обширную документацию и большое сообщество разработчиков, у которых можно получить помощь практически по любой проблеме.

Рассмотрев все преимущества и недостатки – был выбран язык Java.

1.3.2 Выбор базы данных

1.3.2.1 SQLite

SQLite – является библиотекой которая реализует автономный транзакционный SQL движок базы данных. Код для SQLite находится в открытом доступе и, таким образом, является бесплатным для использования в любых целях, коммерческих или частных. SQLite является самой широко развернутой базой данных в мире [9].

SQLite – это встроенный движок базы данных SQL. В отличие от большинства других баз данных SQL, SQLite не имеет отдельного серверного процесса. SQLite читает и записывает напрямую в файлы. Полная база данных

SQL с несколькими таблицами, индексами, триггерами и представлениями содержится в одном файле. Формат файла базы данных является кроссплатформенным. При всех включенных функциях размер библиотеки может быть менее 600 КБ.

SQLite очень тщательно тестируется перед каждым выпуском и имеет очень надежную репутацию. Большая часть исходного кода SQLite посвящена исключительно тестированию и проверке.

Данная СУБД имеет 5 типов данных (NULL, INTEGER, REAL, TEXT, BLOB).

1.3.2.2 MySQL

MySQL – это быстрая и простая в использовании СУБД, используемая для многих малых и крупных приложений [10].

MySQL обычно используется как сервер, обращение к которому происходит локально, либо с помощью удалённых клиентов. Однако в дистрибутив входит библиотека, которая позволяет встраивать MySQL в автономные программы.

Данная СУБД имеет 21 тип данных (INTEGER, FLOAT, DOUBLE, DATE, TIME, TEXT, ENUM и т.д.).

Из достоинств данной СУБД хотелось бы отметить: безопасность, простоту в работе и богатый функционал, масштабируемость и скорость.

1.3.2.3 PostgreSQL

PostgreSQL – это одна из самых профессиональных СУБД, часто используется для веб-баз данных. Она соответствует стандартам SQL и свободно распространяется [11].

Отличие PostgreSQL от других СУБД в поддержке востребованного реляционного и объектно-ориентированного подхода к БД. PostgreSQL

поддерживает надежные транзакции (атомарность, изоляционность, последовательность, прочность). Мощные технологии PostgreSQL делают данную СУБД очень производительной. PostgreSQL имеет функции, которые упрощают использование повторяемых операций.

PostgreSQL имеет 28 типов данных, что на порядок больше, чем у других.

1.3.2.4 Cloud Firestore

Недавно Google выпустил релизную версию Cloud Firestore. Cloud Firestore — это быстрая, полностью управляемая, облачная NoSQL база данных, которая упрощает хранение, синхронизацию и запросы данных мобильных, веб-приложений и приложений в глобальном масштабе [12]. Его клиентские библиотеки обеспечивают оперативную синхронизацию и автономную поддержку данных, а его функции безопасности и интеграции с Firebase и Google Cloud Platform (GCP) ускоряют создание действительно серверных приложений.

Cloud Firestore кэширует данные, которые приложения часто использует, поэтому приложение может писать, читать, слушать обновления и делать запросы даже, когда устройство находится в офлайне. Когда устройство вернется в онлайн - Firestore внесет локальные изменения в облако.

1.3.2.5 Итог

Из рассмотренных решений, можно сделать вывод, что не все СУБД справятся с поставленным заданием на выпускную квалификационную работу.

Так как в поставленном задании на выпускную квалификационную работу было указано, что приложение должно иметь локальную и облачную базу данных, и периодически синхронизоваться с ней, то SQLite не подходит.

Поскольку SQLite не имеет сервера, концепция запуска сервера в облаке не имеет смысла. SQLite – это библиотека, которая включается в приложение и позволяет читать и записывать в файл SQLite используя SQL запросы. Также присутствуют ограничения многопользовательского доступа.

Для СУБД MySQL и PostrgreSQL необходимо арендовать сервер и оплачивать его, что ставит их ниже Cloud Firestore, где сервер выделяется бесплатно, но с ограничениями в 50 тысяч запросов на чтение и 50 тысяч запрос на удаление и редактирование в день. Этого будет вполне достаточно. Так же для работы с Cloud Firestore не нужно ничего устанавливать, вся работа с БД происходит посредством браузера.

Хоть и функционал у MySQL и PostgreSQL больше, нежели у Cloud Firestore он будет излишним. Было принято решение начать работу с Cloud Firestore из-за его простоты, бесплатности в плане аренды сервера и удобства работы с ним.

1.3.3 Итог выбора инструментов

Приложение должно работать на операционной системе Android и обеспечивать графический интерфейс пользователя.

При выборе фреймворка главными условиями были: бесплатность, функциональность и удобство, поэтому выбор пал на Android Studio. Так же данный фреймворк с 2014 года является официальной средой разработки под ОС Android.

Что же касается языка программирования, то выбор пал на Java по нескольким причинам:

- java является первым официальным языком программирования под Android;
- обширная документация от Google;
- множество разнообразных библиотек;
- многопоточность;

- автоматическое управление памятью;

- ООП.

В качестве облачной базы данных была выбрана Cloud Firestore.

1.4 Итоги анализа задания

На основе анализа аналогов был сделан вывод, что

- ни одно из рассмотренных приложений не позволяет обеспечить эффективное выполнение задач автовладельца:

- напоминания о событиях: плановое ТО, ремонт, страховка;

- расчёт расхода топлива и стоимости 1 км пути;

- распределение расходов на заправку, мойку, ремонт, обслуживание и многое другое;

- возможность выгрузки данных, синхронизация с удалённой БД.

В качестве инструментов для работы были выбраны: IDE Android Studio, язык Java и облачная база данных Cloud Firestore.

Также были сформулированы функциональные требования к разрабатываемому приложению в виде диаграммы прецедентов и их текстового описания.

2 Проектирование и реализация приложения

2.1 Архитектура приложения

На рисунке 8 представлена разработанная схема архитектуры приложения. Пользователь, взаимодействуя с клиентом, проходит авторизацию, получая ответ от сервера. Пройдя авторизацию, пользователь попадает в основное окно приложения, где и происходит взаимодействие с данными. При добавлении/редактировании/удалении данных, происходит синхронизация с локальной базой данных, а затем с облачной базой данных (Cloud Firestore).

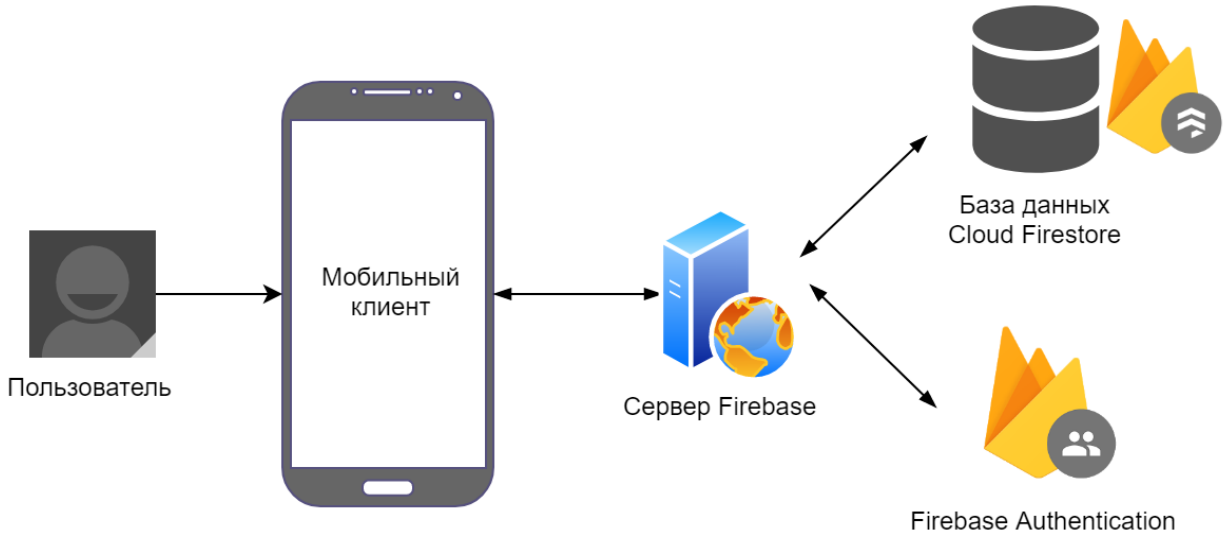

Рисунок 8 – Архитектура приложения

2.2 Структура приложения

В результате объектно-ориентированной декомпозиции задачи были выделены следующие основные сущности: интерфейсы Vehicle, Expense, Serializable, классы User, Car, Refueling, Product, Service. Интерфейс Vehicle

предоставляет методы getIdDoc (выполняет действие получения идентификатора документа в базе данных), getPhotoUri (выполняет действие получения пути до изображения автомобиля), getBrand (выполняет действие получения бренда авто), getModel (выполняет действие получения модели авто), getName (выполняет действие получения имени авто), getColor (выполняет действие получения цвета авто), getYearIssue (выполняет действие получения года выпуска авто), getMileage (выполняет действие получения пробега авто), getPurchseDate (выполняет действие получения даты приобретения авто), getDateAdd (выполняет действие получения даты добавления авто в базу данных).

Класс Car реализует интерфейс Vehicle, представляет в модели предметной области автомобиль. Класс Car имеет атрибуты number, fuelType, vin, а также методы getNumber (выполняет действие получения регистрационного номера авто), getFuelType (выполняет действие получения типа топлива авто), getVin (выполняет действие получения VIN номера авто).

Интерфейс Expense предоставляет методы getIdDoc (выполняет действие получения идентификатора документа в базе данных), getType (выполняет действие получения типа расхода), getDate (выполняет действие получения даты совершения расхода), getAmount (выполняет действие получения суммы расхода), getMileage (выполняет действие получения пробега на момент совершения расхода), getNote (выполняет действие получения примечания), getDateAdd (выполняет действие получения даты добавления расхода в базу данных).

Класс Refueling реализует интерфейс Expense, представляет в модели предметной области заправку. Класс Refueling имеет атрибуты volume, price, fuelType, fuelGrade, а также методы getVolume (выполняет действие получения объёма залитого топлива), getPrice (выполняет действие получения стоимости литра/галлона топлива), getFuelType (выполняет действие получения типа залитого топлива), getFuelGrade (выполняет действие получения марки залитого топлива).

Классы Product и Service реализуют интерфейс Expense, представляя в модели предметной области типы расходов «продукт» и «сервис». Классы Product и Service на данный момент не имеют уникальных атрибутов.

На диаграмме классов продемонстрирована композиция между классом User и классом Car. И композиция между классом Car и классами Service, Product и Refueling. Это говорит о жесткой зависимости времени существования экземпляров класса контейнера и экземпляров содержащихся классов. Если контейнер уничтожен, то все его содержимое будет уничтожено.

Класс Car реализует поведение интерфейса Serializable.

Таким образом, система может быть расширена без изменения существующего кода путём добавления новых классов и реализуя необходимый интерфейс.

Диаграмма классов представлена на рисунке 9.

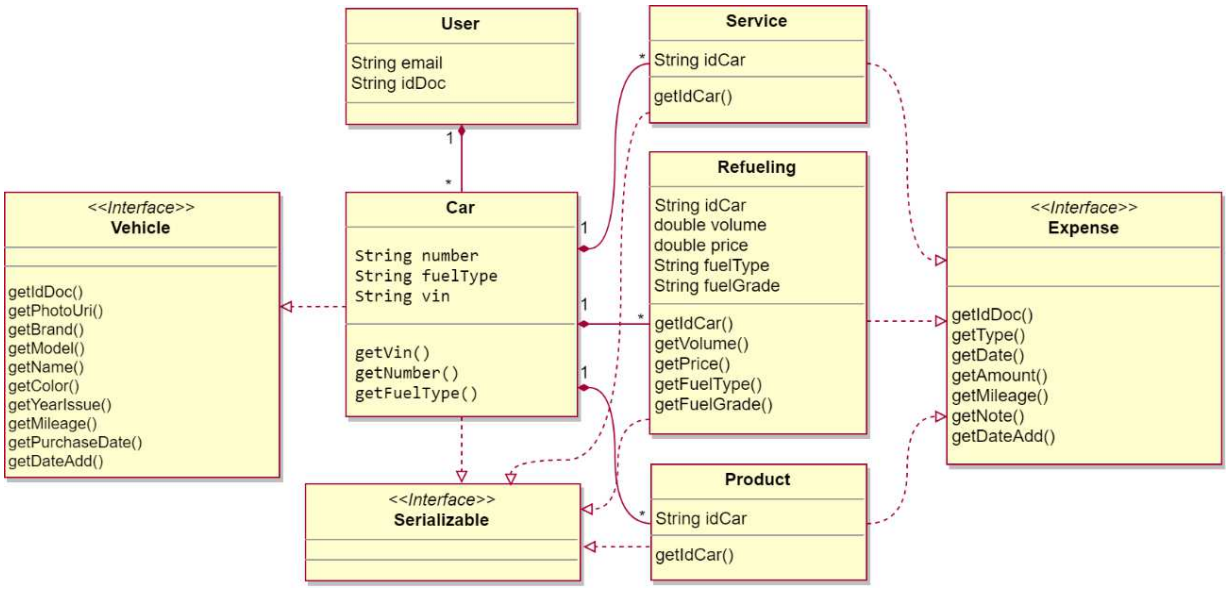

Рисунок 9 – Диаграмма классов

2.3 Структура базы данных

В приложении используется облачная NoSQL база данных Cloud Firestore.

Она использует документы и коллекции для хранения данных. Документ – это запись, содержащая некие поля. Документы группируются в коллекции. Документ может содержать коллекции. Аналогия с SQL-базой данных: коллекция – это таблица, документ – это запись в данной таблице. Коллекция может содержать документы с разными полями.

Приложение можно использовать даже при отсутствии интернета, после его появления, Cloud Firestore автоматически синхронизирует данные с облаком.

Диаграмма сущностей и связей базы данных представлена на рисунке 10.

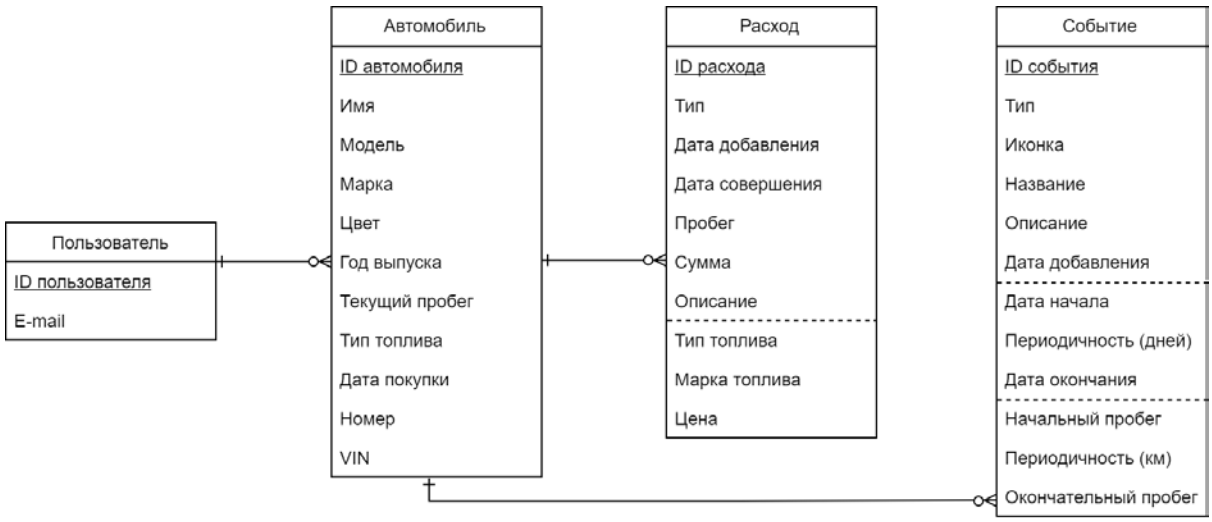

Рисунок 10 – Диаграмма сущностей и связей

Она поддерживает 11 типов данных. В приложении используются следующие типы данных:

- Date
- Integer

- Floating-point number

- Text string

В каком виде хранятся данные в базе данных показано в листинге 1.

Листинг 1 – Вид хранения данных в базе данных

```
brand: "Lada"
color: "Серый"
dateAdd: 3 мая 2019 г., 19:48:16 UTC+7
fuelType: "Бензин"
idDoc: "stITJ64I9LAJHr03052019194816"
mileage: 152888
model: "2110"
name: "Lada 2110"
number: "О826КК"
photoUri: "file:///data/carnotebook/car_photo/cropLada.jpg"
purchaseDate: 9 августа 2012 г., 00:00:00 UTC+8
vin: "EWG54FFY54H534"
yearIssue: 1 января 2009 г., 00:00:00 UTC+7
```

2.4 Графический интерфейс

Реализация графической части была осуществлена с помощью макетов. Определение визуальной структуры пользовательского интерфейса происходит с помощью макета. Макет состоит из элементов построенных с использованием иерархии объектов View и ViewGroup [13]. ViewGroup – это невидимый контейнер, он определяет структуру макета для View и других объектов ViewGroup, как показано на рисунке 11.

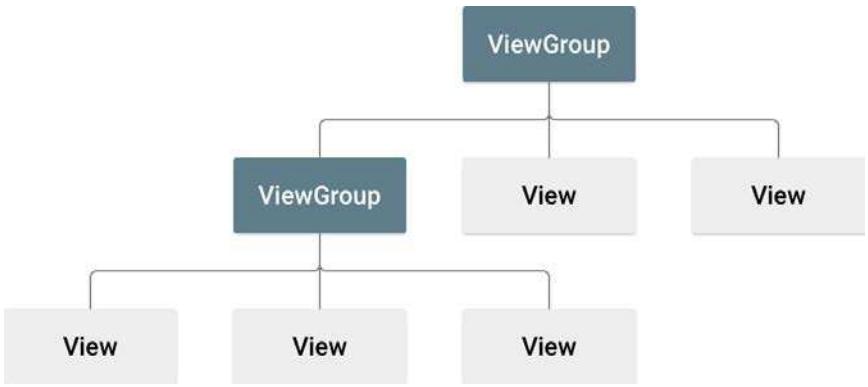

Рисунок 11 – Иерархии представления, которая определяет макет пользовательского интерфейса

Элементы пользовательского интерфейса объявляются в XML файлах.

В объявлении пользовательского интерфейса в файле XML есть ряд преимуществ, таких как:

- эффективное отделение представления приложения от кода, который управляет его поведением [13];

- описание пользовательского интерфейса не в коде приложения [13];

- возможность изменять или адаптировать интерфейс без правок исходного кода и его последующей компиляции [13];

- разделение XML файлов для каждого экрана;

- удобная отладка проблем.

Пример макета XML, в котором используется вертикальный объект LinearLayout, в котором размещены элементы ImageView и TextView расположен в листинге 2.

Листинг 2 – Пример макета XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingStart="24dp"
    android:paddingEnd="24dp">
    <ImageView
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_margin="10dp"
        android:contentDescription="@string/warning"
        android:src="@drawable/ic_warning" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="@string/delete_car_confirmation"
        android:textSize="16sp" />
</LinearLayout>
```

В проект были реализованы макеты для каждого класса, а также макеты для пользовательских виджетов, меню, диалоговых окон и отдельных

элементов списка. Отношение нескольких XML макетов к классам предоставлено в таблице 2.

Таблица 2 – Отношение XML макетов к классам.

Класс	XML макет
CarAddActivity	Activity_car_add.xml
CarEditActivity	Activity_car_edit.xml
CarViewActivity	Activity_car_view.xml
CarListActivity	Activity_car_list.xml Item_car.xml
MainActivity	Activity_main.xml Content_main.xml Nav_header_main.xml Activity_main_drawer.xml

Пример реализации макета пользовательского интерфейса на примере expense_view.xml представлен на рисунке 12.

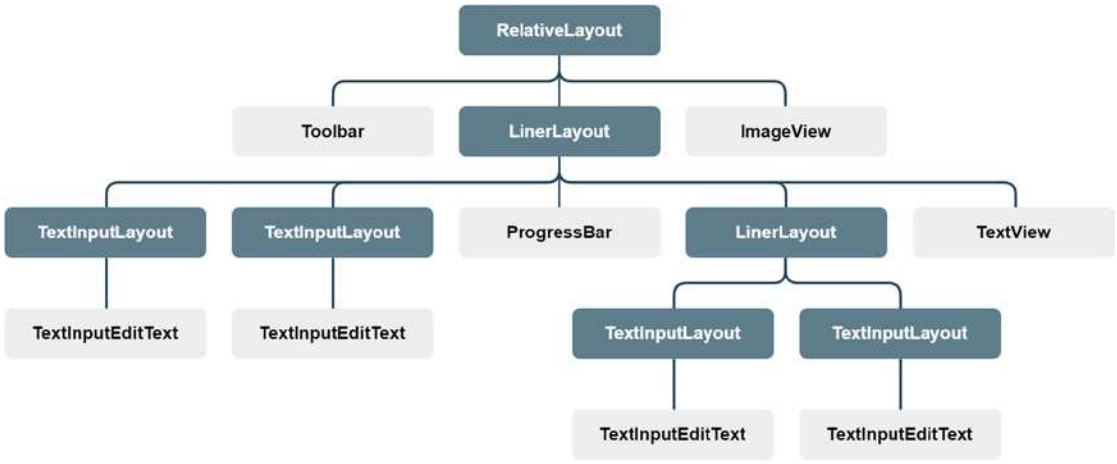

Рисунок 12 – Иерархия представления пользовательского интерфейса

Графическое представление исходя из реализации макета представлено на рисунке 13.

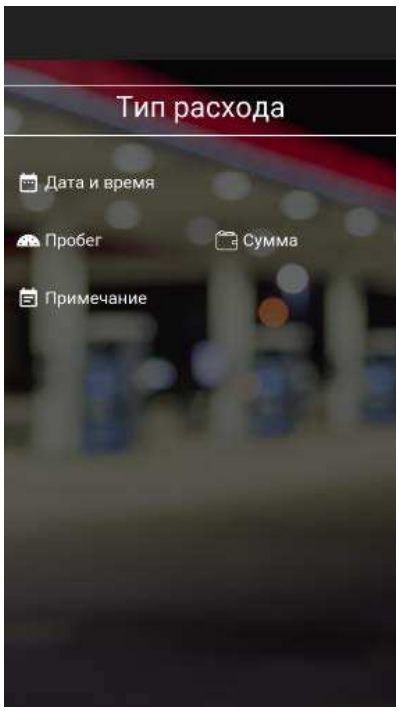

Рисунок 13 – Графическое представление пользовательского интерфейса

Пример кода реализации текстового заголовка (TextView) представлен в листинге 3.

Листинг 3 – Реализация текстового заголовка (TextView).

```
<TextView
    android:id="@+id/tv_type_expense"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:text="@string/expense_type_label"
    android:textColor="@android:color/white"
    android:textSize="30sp" />
```

Описание свойств TextView:

- id – уникальный идентификатор;
- layout_width – ширина;
- layout_height – высота;
- layout_gravity – привязка к центру родителя;
- text – текст;
- textColor – цвет текста;
- textSize – размер текста.

2.5 Примеры интерфейса приложения

На рисунках 14, 15, 16 изображен интерфейс приложения.

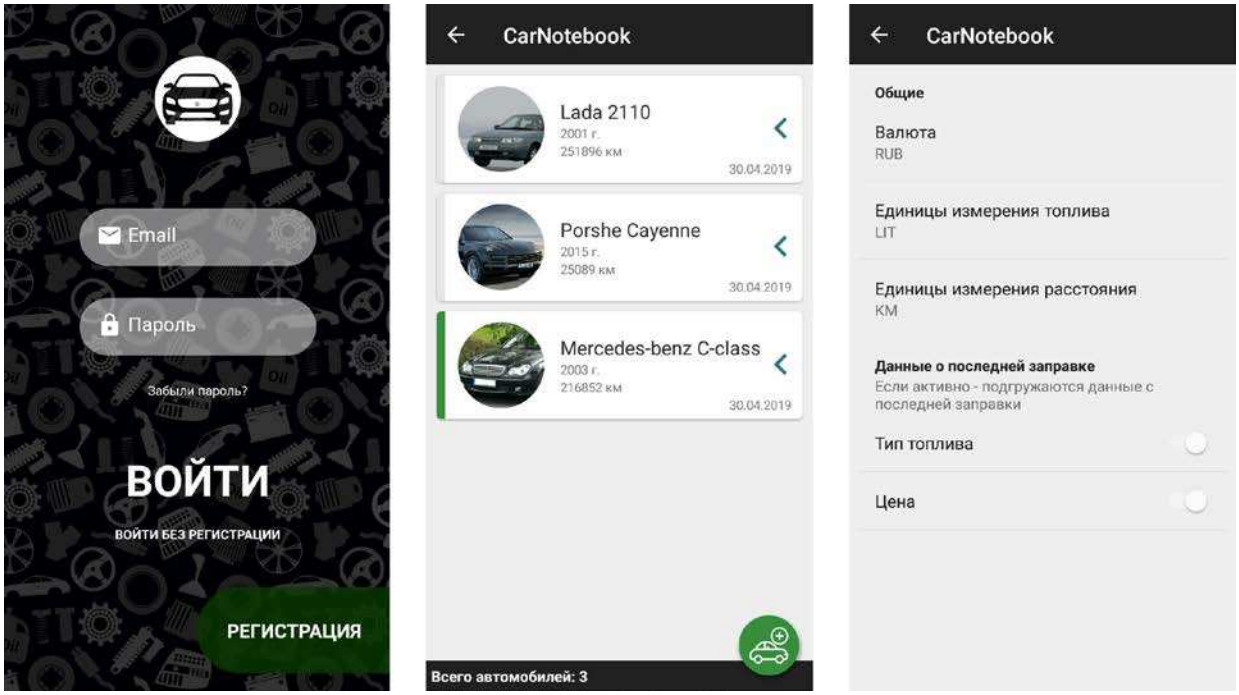

Рисунок 14 – Окна: «Авторизация», «Гараж», «Настройки»

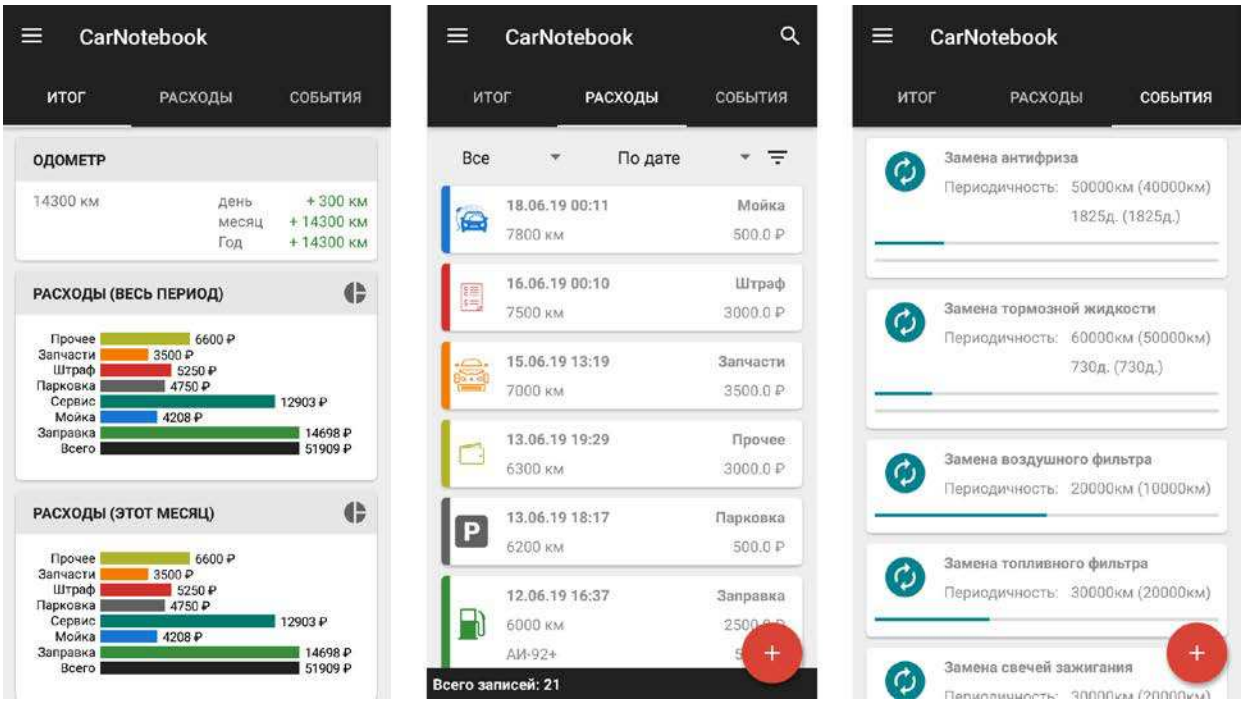

Рисунок 15 – «Итог», «Расходы», «События»

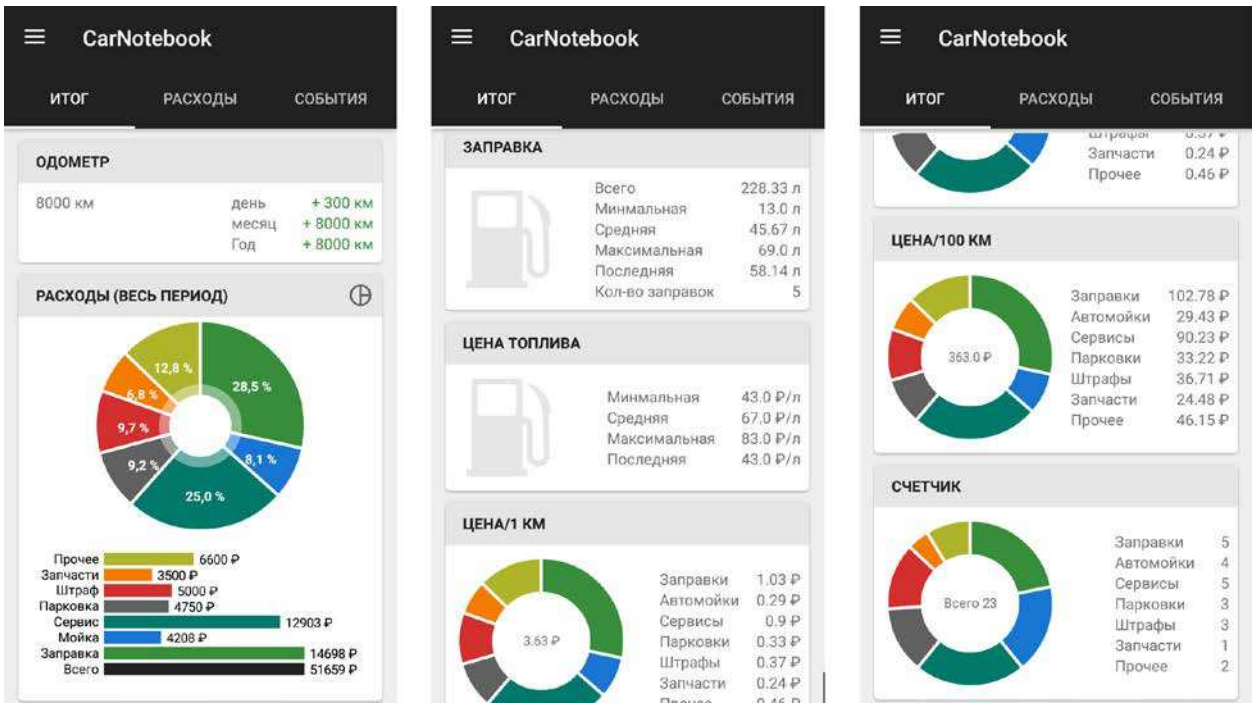

Рисунок 16 – Окно «Итог»

3 Инструкции

3.1 Инструкция пользователя

При открытии программы откроется окно авторизации. Для авторизации не обязательно проходить регистрацию – имеется возможность анонимной авторизации, после чего можно будет продолжить регистрацию из главного окна программы. Данный вид авторизации не рекомендуется по ряду причин: при переустановке приложения данные о авто и расходах удаляются, нет возможности синхронизации с другим устройством.

Для начала работы с приложением необходимо открыть меню слева и перейти в гараж, где нужно добавить ваш автомобиль. После добавления авто, и вернувшись на главный экран – появится возможность добавлять расходы. Нажав на «+» нужно выбрать тип расхода и заполнить поля.

Для смены автомобиля (при количестве автомобилей в гараже больше 1), нужно перейти в «гараж» и «смахнуть» автомобиль влево (появится меню с пунктами: выбрать, редактировать, удалить) и выбрать пункт меню «выбрать». После чего выбранное авто подсветиться зеленым слева. Все последующие добавления расходов будут относиться к выбранному автомобилю.

Для редактирования/удаления расхода, необходимо удерживать его в течении 2 сек и выбрать соответствующий пункт меню.

Для смены валюты, либо единиц измерения топлива/расстояния, необходимо открыть меню слева и выбрать пункт настройки. В настройках нажать на соответствующий пункт и выбрать значение из выпадающего списка.

Для поиска расхода, нужно нажать на лупу и ввести текст.

Для сортировки расходов, нужно нажать на «сортировка» и выбрать соответствующий критерий для сортировки, после чего список обновиться исходя из выбранных критериев.

Для просмотра статистики, находясь на главном экране, нужно выбрать «статистика», после чего окно сменится и появится статистика. Аналогично с графиками и итоговой информацией.

Для выхода из программы, нужно вызвать меню слева и нажать кнопку выйти, после чего появится диалоговое окно с предупреждением, нажать «выйти».

Внимание! При анонимной авторизации и последующем выходе из программы – данные не сохраняются. Необходимо продолжить регистрацию из главного меню приложения.

3.2 Инструкция разработчика

Для работы с исходным кодом и его компиляцией необходимо:

- IDE Android Studio 3.4;
- SDK 28.0.2;
- JRE: 1.8.0_152-release-1343-b01 amd64;
- JVM: OpenJDK 64-Bit Server VM by JetBrains;
- Google Play services version 49;
- Эмулятор или смартфон под управлением ОС Android (API > 20).

Все ПО распространяется на ОС GNU/Linux, macOS и Microsoft Windows. IDE, JRE, JVM можно загрузить с официального сайта [14]. SDK, Google Play services и эмулятор можно загрузить прямо из Android Studio (рисунок 17).

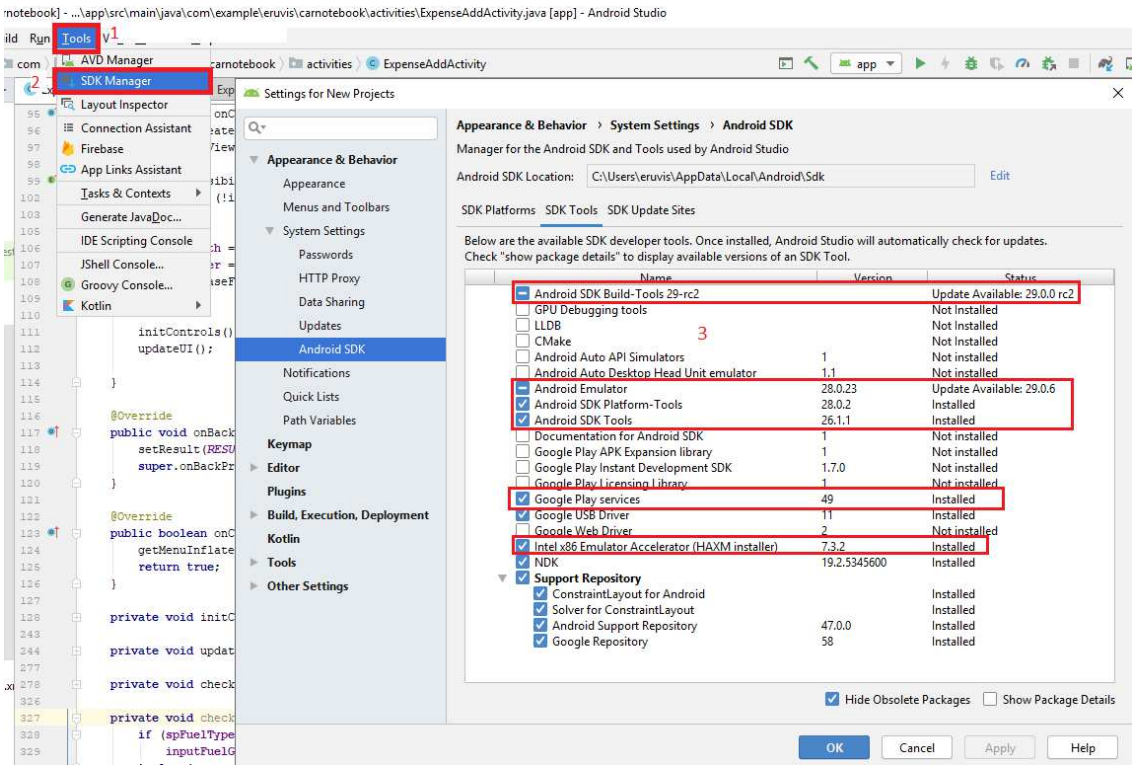

Рисунок 17 – Установка дополнительных инструментов

После открытия проекта, необходимо подождать, пока проект соберется. Затем, подключить телефон к компьютеру, либо же запустить эмулятор и нажать кнопку «Run», после чего должно скомпилироваться и запуститься приложение (рисунок 18).

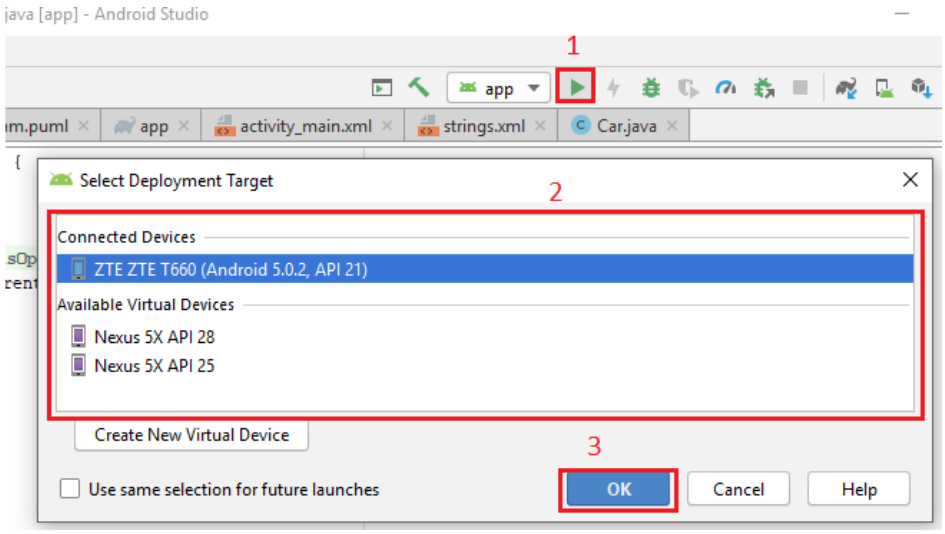

Рисунок 18 – Последовательность действий

Для добавления новых пунктов меню, необходимо открыть файл activity_main_drawer.xml и добавить новый элемент item:

<item android:id="Идентификатор пункта меню"

android:icon="Иконка"

android:title="Текст" />

После чего в меню появится новый пункт. Вся работа с пунктами меню производится в файле MainActivity.

Для добавления нового типа расхода, необходимо создать новый класс наследующий интерфейс Expense. Для работы с графической составляющей используются файлы: activity_expense_add.xml, activity_expense_edit.xml, activity_expense_view.xml и ExpenseRecyclerView.

Для добавления нового типа транспортного средства, необходимо создать новый класс наследующий интерфейс Vehicle. Для работы с графической составляющей используются файлы: activity_car_add.xml, activity_car_edit.xml, activity_car_view.xml и CarRecyclerView.

Для добавления, либо редактирования пунктов настроек используется файл preference.xml.

Для добавления нового окна в главное окно, необходимо создать новый класс наследующий класс Fragment и xml файл для графической составляющей. Затем добавить данные в ViewPagerAdapter.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы было спроектировано и реализовано приложение «Мобильный помощник автовладельца», которое в будущем будет опубликовано в Play Market.

В разработанном приложении были реализованы следующие функции:

- регистрация, авторизация;
- добавление больше одного автомобиля и работа с каждым из них;
- добавление расходов разных типов для авто;
- просмотр итоговой информации (графики, диаграммы, текст);
- напоминания о событиях: плановое ТО, ремонт, страховка;
- расчёт расхода топлива и стоимости 1 км пути;
- сортировка расходов;
- настройки (валюта, выбор единиц измерения топлива и расстояния, сохранение данных о последней поездке);
- возможность выгрузки данных, синхронизация с удалённой БД.

Приложение было неоднократно протестировано и работает исправно.

Приложение можно улучшить, путем добавления возможностей или улучшения имеющихся:

- возможность сканировать данные с чека по QR-коду;
- возможность удалять не интересующую итоговую информация с главного экрана.

Последняя версия приложения размещена в git-репозитории [15].

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. My Car [Электронный ресурс]. – Режим доступа: https://play.google.com/store/apps/details?id=cz.dorazil.jan.MyCar (дата обращения: 18.03.2019).

2. My Cars [Электронный ресурс]. – Режим доступа: https://play.google.com/store/apps/details?id=com.aguirre.android.mycar.activity (дата обращения: 18.03.2019).

3. Авто Расходы [Электронный ресурс]. – Режим доступа: https://play.google.com/store/apps/details?id=com.mycapitalgp.autoexpenses (дата обращения: 18.03.2019).

4. Моя Машина [Электронный ресурс]. – Режим доступа: https://play.google.com/store/apps/details?id=ru.Tkachuk.MyCar (дата обращения: 18.03.2019).

5. Java [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Java (дата обращения: 22.05.2019).

6. Kotlin [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Kotlin (дата обращения: 22.05.2019).

7. C++ – Энциклопедия [Электронный ресурс]. – Режим доступа: http://progopedia.ru/language/c-plus-plus/ (дата обращения: 13.04.2019).

8. C Sharp [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/ (дата обращения: 12.06.2019).

9. SQLite [Электронный ресурс]. – Режим доступа: https://www.sqlite.org/index.html (дата обращения: 20.04.2019).

10. MySQL – Свободная реляционная СУБД [Электронный ресурс]. – Режим доступа: https://habr.com/ru/hub/mysql/ (дата обращения: 24.04.2019).

11. PostgreSQL [Электронный ресурс]. – Режим доступа: https://www.postgresql.org/ (дата обращения: 20.04.2019).

12. Cloud Firestore [Электронный ресурс]. – Режим доступа: https://firebase.google.com/docs/firestore (дата обращения: 29.05.2019).

13. Макеты [Электронный ресурс]. – Режим доступа: https://developer.android.com/guide/topics/ui/declaring-layout?hl=ru (дата обращения: 20.04.2019).

14. Download Android Studio and SDK Tools [Электронный ресурс]. – Режим доступа: https://developer.android.com/studio (дата обращения: 30.04.2019).

15. Eruvis builds [Электронный ресурс]. – Режим доступа: https://bitbucket.org/eruvis/build/src/master/ (дата обращения: 30.04.2019).

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись
«_27_» _06_____ 2019 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 – «Информатика и вычислительная техника»

Мобильный помощник автовладельца

Руководитель	24.06.19 подпись, дата	ст. преподаватель	К.В. Пушкарев
Выпускник	24.06.19 подпись, дата		Р.У. Исаев
Консультант	24.06.19 подпись, дата	канд. техн. наук, доцент	Л.И. Покидышева
Нормоконтролер	27.06.19 подпись, дата		В.И. Иванов

Красноярск 2019