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

РАБОТА ПРОВЕРЕНА

Рецензент
Руководитель ИП «Бокарева С.А.
(фирма «Альфа-софт»)»

_____ С.А. Бокарева

“___”_____ 2020 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой,
д.ф.-м.н., профессор

_____ Л.Б. Соколинский

“___”_____ 2020 г.

Разработка мобильного геоинформационного Android-приложения для поиска автозаправочных станций

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2020.308-033.ВКР

Научный руководитель,
к. ф.-м. н., доцент кафедры СП
_____А.Т. Латипова

Автор работы,
студент группы КЭ-401
_____И.А. Шарков

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
“___”_____ 2020 г.

Челябинск-2020

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

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
09.02.2020

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-401
Шаркову Илье Александровичу,
обучающемуся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. Тема работы (утверждена приказом ректора от 24.04.2020 № 627)
Разработка мобильного геоинформационного Android-приложения для поиска автозаправочных станций.
2. Срок сдачи студентом законченной работы: 05.06.2020 г.
3. Исходные данные к работе
3.1. JDK (Java SE Development Kit) [Электронный ресурс] URL: http://www.oracle.com/technetwork/java/javase/downloads/index.html.
3.2. Android Studio the Official IDE for Android [Электронный ресурс] URL: https://developer.android.com/studio/.
4. Перечень подлежащих разработке вопросов
4.1. Провести обзор существующих решений по тематике работы.
4.2. Выполнить проектирование системы.
4.3. Выполнить реализацию приложения на платформе Android.
4.4. Провести функциональное тестирование полученного приложения.
5. Дата выдачи задания: 08.02.2020.

Научный руководитель А.Т. Латипова
к ф.-м. н., доцент кафедры СП

Задание принял к исполнению И.А. Шарков

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы работы

Актуальность разработки под Android представлена данными, которые отражает диаграмма на рисунке 1, отображающая количество устройств, работающих на данной платформе в мире. У Android есть несколько преимуществ по сравнению с другими платформами. Во-первых, порог вхождения достаточно низок. Чтобы начать разрабатывать необходимо иметь минимальные знания Java, а также загрузить IDE и SDK. Во-вторых, у выбранной платформы огромное сообщество, с помощью которого появляется большое количество постоянно обновляемых материалов – документаций, книг, онлайн-уроков.

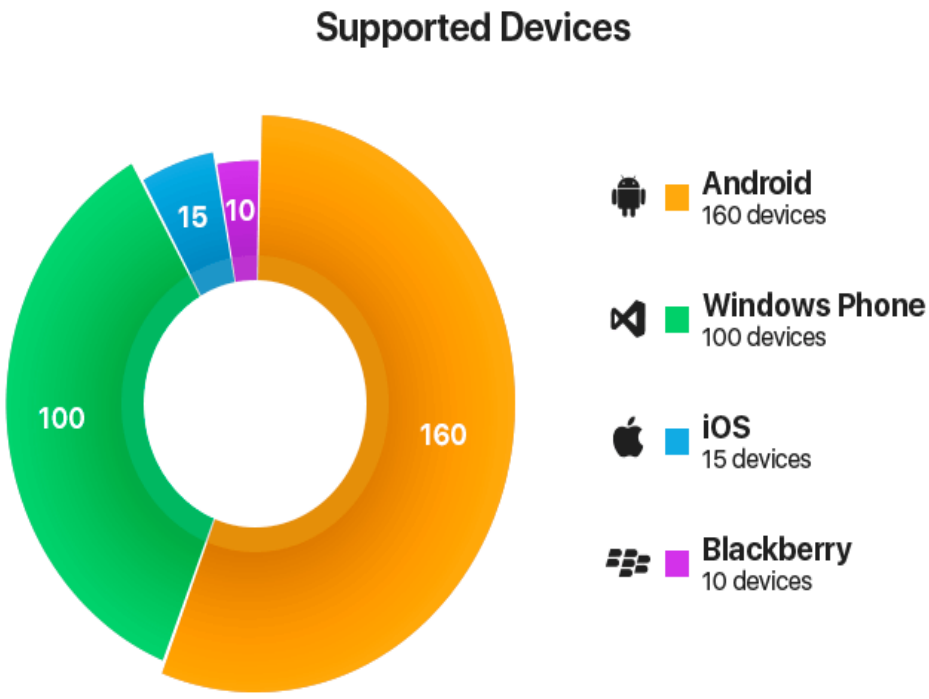


Рис. 1. Доли популярных мобильных платформ в начале 2020 года

На сегодняшний день существует множество различных приложений, а разработка приложений переживает период своего подъема. Навигация по печатным картам уже давно устарела, сейчас популярностью пользуются

GPS-навигаторы. Благодаря развитию технологий мобильной разработки удалось интегрировать навигацию в компактные мобильные устройства. Самый известный мобильный навигатор во всём мире на текущий момент Google maps.

Карта Google Maps – набор приложений, построенных на основе бесплатного картографического сервиса и технологии, предоставляемых компанией Google. Созданы в 2005 году.

Сервис представляет собой карту и спутниковые снимки планеты Земля. Для многих регионов доступны высокодетализированные аэрофотоснимки (снятые с высоты 250-500 м), для некоторых – с возможностью просмотра под углом 45° с четырёх сторон света.

Google maps делает возможным просмотр карты местности, прокладывание маршрута между двумя пунктами назначения, просмотр отзывов и другой дополнительной информации об искомом объекте.

Цель и задачи

Целью данной работы является разработка мобильного геоинформационного Android-приложения для поиска автозаправочных станций. Мобильное приложение позволит пользователям просматривать масштабируемую карту, смотреть информацию об АЗС, прокладывать маршрут до ближайших АЗС, просматривать актуальные цены на бензин и другую информацию.

Для достижения поставленной цели мною были сформулированы следующие задачи:

1) провести обзор аналогичных проектов по тематике работы, выполнить их сравнительный анализ;
2) выявить главные требования, предъявляемые к приложению для инициализации карт;
3) выполнить проектирование приложения средствами языка UML;
4) выполнить проектирование;
5) выполнить реализацию приложения;

6) провести функциональное тестирование полученного приложения.

Структура и объем работы

Работа состоит из введения, четырех глав, заключения, библиографии и приложения. Объем работы составляет 40 страниц, объем библиографии – 15 источников.

Краткое содержание работы

В первой главе, «Анализ предметной области» дается обзор аналогичных проектов и обзор существующих технологий для реализации проекта. В этом разделе выявляются наиболее перспективные технологии для создания приложения из рассмотренных.

Во второй главе, «Требования к системе» выявляются функциональные и нефункциональные требования к системе, а также выполняется проектирование системы с использованием спецификации языка UML - создаются диаграммы классов и вариантов использования.

В третьей главе, «Архитектура системы» описываются детали реализации мобильного приложения на платформе Android.

В четвертой главе, приведены результаты тестирования мобильного приложения вместе со скриншотами основных экранов разработанного приложения.

В заключении сделаны выводы о проделанной работе.

Приложение содержит спецификацию основных вариантов использования, сформулированных для мобильного приложения.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Анализ аналогичных проектов

В магазине Google Play на данный момент существует несколько аналогичных приложений, но все они имеют некоторые отличия друг от друга. На рисунке 2 представлены скриншоты экранов из приложения GPS – Route Finder, которое также основано на Google Maps. Функционал не слишком широкий, так как большинство функций этого приложения вызывают работу других приложений, то есть функции в приложении GPS – Route Finder присутствуют не в виде отдельных экранов данного приложения. При проведении анализа отзывов о данном приложении было выявлено два недостатка – всплывающая реклама после каждого перехода между экранами приложения и долгая загрузка в любом экране приложения, то есть приложение долго ждёт отклик от сервера.

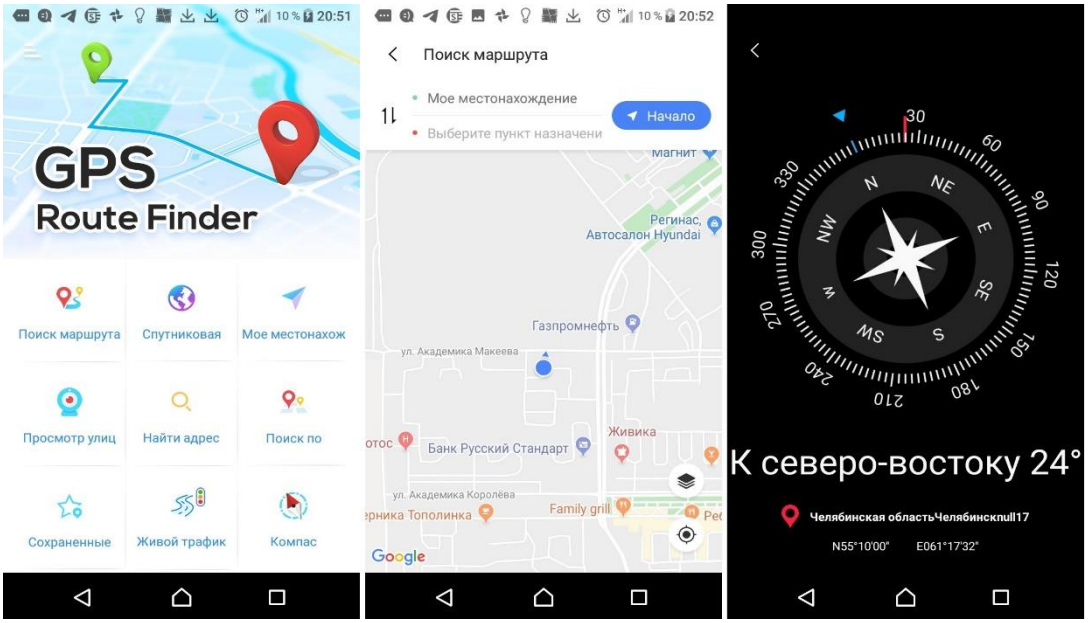


Рис 2. Скриншоты приложения GPS – Route Finder для Android

Еще одним аналогом является приложение Voice GPS Navigator, скриншоты которого приведены на рисунке 3. Данное приложение позволяет просматривать карты на основе Mapbox. Из недостатков данного приложения можно выделить большое количество рекламы, выдвижные вкладки на экране карты, закрывающие значительную часть карты и

отсутствие навигации в активити «Nearby places» с обширным выбором вкладок. Данное приложение – хорошая альтернатива Google maps для Android, так как в нем присутствуют не только основные функции для просмотра карты, но и дополнительные виджеты, позволяющие разнообразить пользовательский интерфейс.

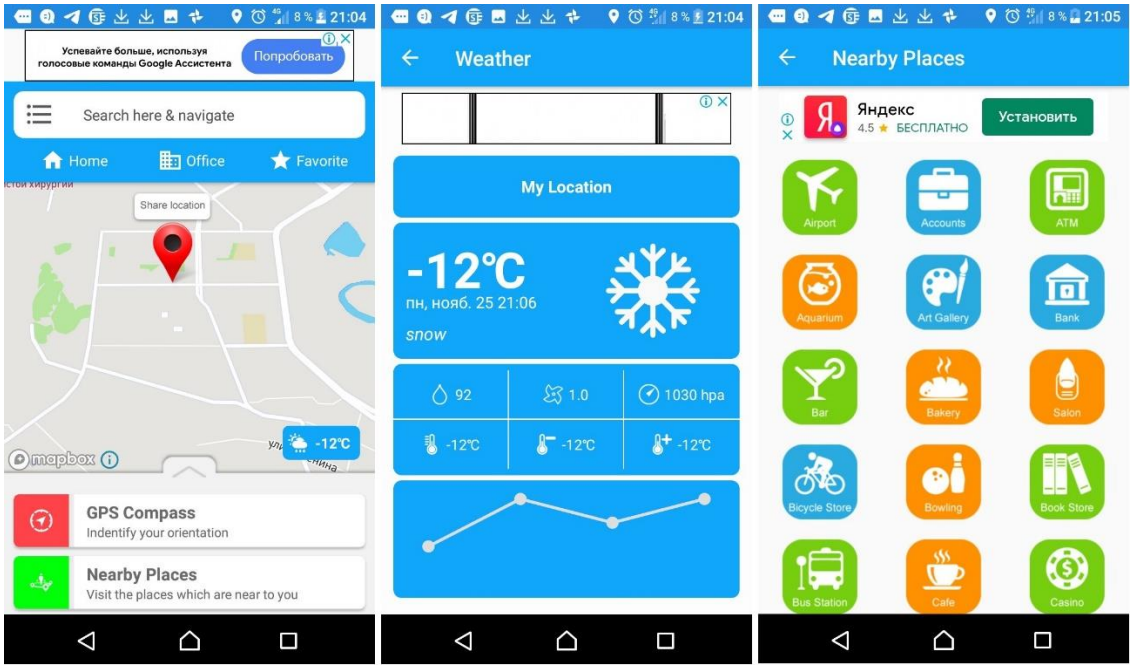


Рис 3. Скриншоты неофициального приложения Voice GPS Navigator для Android

Вывод

В данном подразделе были рассмотрены два приложения для навигации под Android: GPS – Route Finder и Voice GPS Navigator. В работе данных приложениях существенных недостатков не было обнаружено. Однако мною было принято решение разработать еще одно, собственное, мобильное приложение для навигации в учебных целях, а также в целях глубокой личной заинтересованности данной тематикой.

1.2. Анализ существующих технологий для реализации проекта

Для реализации приложения можно использовать следующие технологии разработки мобильных приложений:

1) графические конструкторы для разработки под платформу Android (Nwicode, Freel App);

2) мобильные ОС и языки программирования мобильных приложений (Java, Kotlin, Swift);

3) среды разработки ПО под Android: IntelliJ IDEA, Android Studio.

1.2.1. Графические конструкторы

Графические конструкторы представляют собой сервисы, позволяющие создавать мобильные приложения для определенных нужд без использования языков программирования. Приложения создаются на базе заготовленных в конструкторе шаблонов с добавлением нужных виджетов и вариантов оформления. Как правило, подобные конструкторы позволяют создавать кроссплатформенные приложения.

Одним из примеров графических конструкторов для создания мобильных приложений является *Nwicode* – это лидер на рынке графических конструкторов приложений. На этой платформе можно создавать не только приложения для Android, iOS, но и Web приложения. Данный конструктор имеет открытый исходный код и стили, возможность интеграции с социальными сетями, а также предоставляет бесплатный хостинг для приложений на первый месяц обслуживания. Однако абонентская плата за использование платформы составит $9.99 ежемесячно.

Достоинством любых графических конструкторов в сравнении со средами разработки является возможность написания программ без каких-либо знаний языков программирования, используя готовые шаблоны. Недостатком является отсутствие возможности создания гибкого приложения и ограниченная функциональность приложения, а также некоторые проблемы с безопасностью.

1.2.2. Языки программирования

Важным шагом в создании приложения является выбор языка программирования. В данной работе будет произведено сравнение двух современных популярных языков программирования, предназначенных для разработки мобильных приложений.

Kotlin – это статически типизированный язык программирования от компании JetBrains, работающий поверх JVM и созданный специально для разработки Android приложений. Компилируется также в JavaScript и на другие платформы через инфраструктуру LLVM. Основными достоинствами данного языка программирования являются:

1) лаконичность языка;
2) язык поддерживается компанией Google;
3) полная совместимость с Java.

На данный момент у данного языка есть единственный недостаток – маленькое сообщество. В связи с этим маленькое количество литературы.

Java – строго типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems. Разработка ведётся сообществом, организованным через Java Community Process, язык и основные реализующие его технологии распространяются по лицензии GPL [9]. Достоинствами языка Java являются:

1) автоматическое управление памятью;
2) большое сообщество;
3) повышенная безопасность.

Недостатками данного язык являются:

1) многострочный и сложный код;
2) низкая производительность.

1.2.3. Среды разработки мобильных приложений

Важнейшим элементом в процессе создания приложения в среде разработки IDE является уровень собственной подготовки. От него будет зависеть качество и быстродействие приложения.

IntelliJ IDEA - интегрированная среда разработки программного обеспечения для многих языков программирования, разработанная компанией JetBrains [6]. Достоинствами данной среды разработки являются:

1) множество языков программирования: Java, Kotlin, JavaScript, Python, Ruby, Groovy, Scala, PHP, C, C++;

2) интеграция с Git.

К недостаткам можно отнести относительно низкую производительность, долгое ожидание выполнения компиляции.

Android Studio – интегрированная среда разработки производства Google, с помощью которой разработчикам становятся доступны инструменты для создания приложений на платформе Android ОС [5]. Достоинства:

1) встроенный SDK;

2) удобный конструктор интерфейсов.

Основным недостатком данной среды можно выделить постоянную синхронизацию с Gradle, которая значительно замедляет компиляцию проектов.

Вывод

Было принято решение разработать мобильное приложение на платформе Android с использованием языка программирования Java, среды разработки Android Studio. Поскольку данная среда разработки предоставляет необходимые инструменты для реализации приложения для навигации с Google maps на языке программирования Java.

2. ПРОЕКТИРОВАНИЕ

Проектирование приложения включает в себя работу по планированию навигации и функционала, позволяя структурировать идеи, предотвратить ошибки и избежать лишней работы на ранних стадиях разработки. Проектирование включает в себя определение функциональных и нефункциональных требований. На данном этапе создаются диаграмма классов, а также диаграмма вариантов использования.

2.1. Функциональные требования

Функциональные требования определяют функциональность программного обеспечения, то есть описывают, что необходимо реализовать в продукте или системе, какие возможности должна предоставлять разрабатываемая система. Функциональные требования включают в себя бизнес-требования и пользовательские требования. Были выявлены следующие функциональные требования для мобильного приложения навигации:

1) система должна отображать главное меню;
2) система должна отображать интерактивную карту;
3) система должна отображать ближайшие АЗС на карте;
4) система должна строить маршрут до ближайших АЗС;
5) система должна предоставлять экран для отображения списка цен на бензин;
6) система должна предоставлять экран для отображения новостного портала.

2.2. Нефункциональные требования

Нефункциональные требования описывают свойства и ограничения, накладываемые на систему. Нефункциональные требования определяют, как должна работать система или программный продукт, и какими свойствами или характеристиками она должна обладать. Для реализации

приложения для навигации были сформулированы следующие нефункциональные требования:

1) приложение должно быть написано на языке программирования Java;

2) приложение должно работать на Android с версии 4.2;

3) система должна забирать данные с сервера Google maps с использованием API ключа.

2.3. Варианты использования приложения

Диаграммы вариантов использования применяется для моделирования функциональных требований к программному продукту при его проектировании и разработке.

На рисунке 4 представлена диаграмма вариантов использования мобильного Android-приложения для навигации. Диаграмма вариантов использования отражает отношение между актерами и прецедентами.

В данной системе существует единственный актер – *пользователь*. *Пользователь* может взаимодействовать с приложением: открыть карту, просматривать информацию об известных АЗС, просматривать актуальные цены на бензин, просматривать новости, а также просматривать истории и открывать настройки.

Описание вариантов использования системы

На рисунке 4 представлены следующие базовые варианты использования мобильного приложения:

1) *просмотреть карту* – просмотр карты, и меток на АЗС;

2) *просмотреть информации об АЗС* – просмотр списка популярных АЗС, а также информацию о них;

3) *просмотреть цены на бензин* – просмотр цен на топливо;

4) *просмотреть новости* – просмотр новостей;

5) *просмотр историй* – анекдотов;

6) *просмотр настроек* – просмотр настроек.

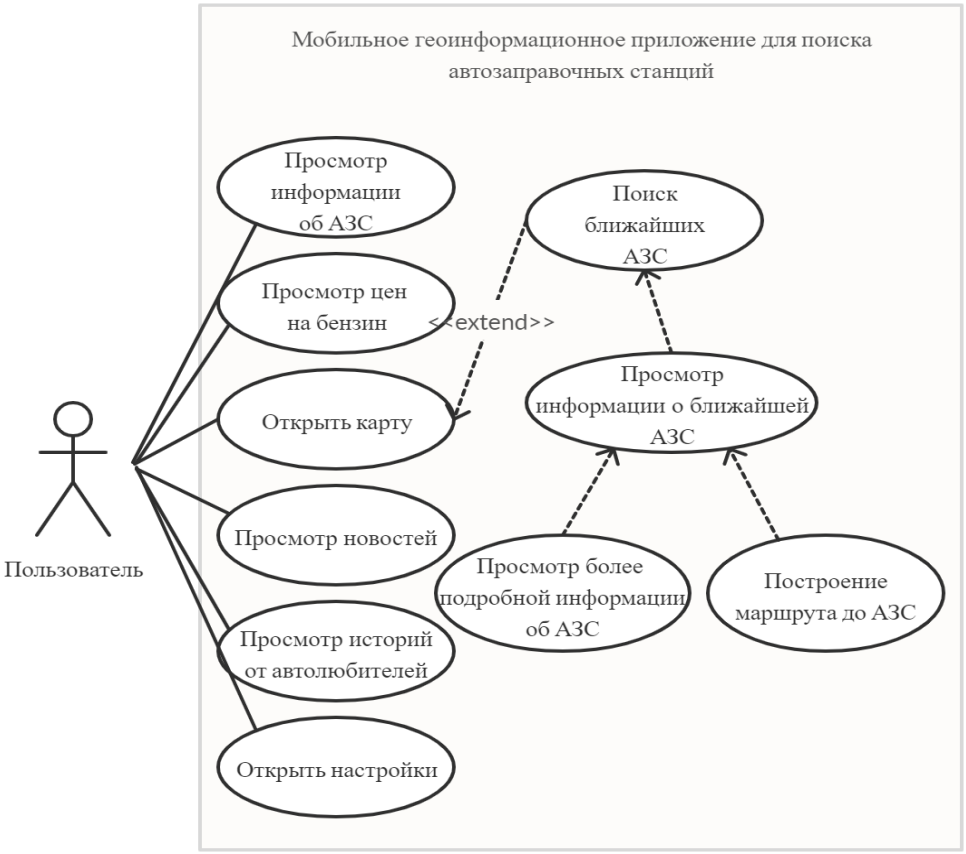


Рис. 4. Диаграмма вариантов использования

2.4. Диаграмма классов

Диаграммы вариантов Диаграмма классов – диаграмма, которая отображает классы системы, их названия, атрибуты, методы и взаимосвязи между ними. Диаграмма классов служит для представления статической структуры модели системы в терминологии классов ООП.

На рисунке 5 представлена диаграмма классов разрабатываемого приложения.

Класс *MainActivity* связан с остальными классами двунаправленной ассоциативной связью, так как остальные классы между собой не связаны, то для перехода из каждого нужно вернуться в *MainActivity*.

Класс Text_Content_Activity предназначен для хранения записанной в него информации и перезаписи этих данных в зависимости от того, какой раздел был выбран пользователем.

Класс Web отображает web-страницу.

Класс MapsActivity отображает карту.

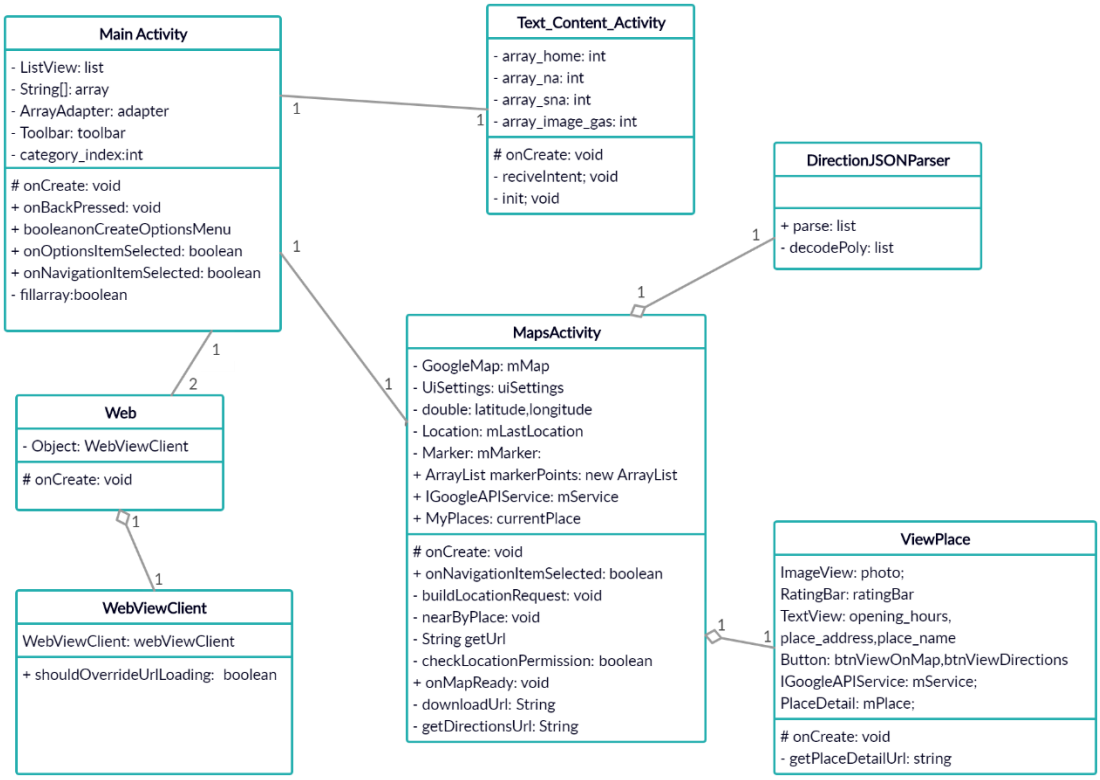


Рис. 5. Диаграмма классов

Вывод

Таким образом, на основе выявленных требований была спроектирована архитектура системы. Были выявлены компоненты системы, а также ее основные элементы. Была спроектирована структура приложения и определен принцип взаимодействия компонентов системы. Дальнейшая реализация системы будет опираться на разработанную структуру.

3. РЕАЛИЗАЦИЯ

Реализация мобильного приложения была разбита на следующие этапы:

1) конфигурация приложения. Создание и настройка android-приложения в Android Studio, добавление запрашиваемых разрешений в файл манифеста;

2) работа с картами и API запросами;

3) создание и настройка xml файлов для вывода полученных данных.

3.1. Конфигурация приложения и структура проекта

Для создания приложения использовался пустой шаблон, содержащий основной класс приложения *MainActivity.java*, а также основной файл разметки *activity_main.xml* [1].

Для использования методов работы с интернет запросами в android необходимо получить разрешение android.permission.INTERNET, для работы с картой необходимы разрешения android.permission.ACCESS_FINE_LOCATION и android.permission. ACCESS_COARSE_LOCATION. Эти разрешения должны быть записаны в файл *AndroidManifest.xml* и запрошены у пользователя в момент установки приложения [2].

Приложение было решено разделить на пять экранов:

1) экран главного меню;
2) экран карты;
3) экран подробной информации об АЗС;
4) экран с актуальными ценами на бензин;
5) экран с новостями.

На рисунке 7 отображена структура проекта.

В папке *manifests* расположен файл *AndroidManidest.xml*, содержащий важную информацию о приложении, которая потребуется системе Android.

В папке *res/layout* расположены xml файлы, используемые для вывода данных на экран.

Файлы *MapsActivity.xml* и *MainActivity.xml* содержат в себе виджет *MapView* [8], то есть карту.

В Файле *nav_header_main.xml* [7] располагаются виджеты *ImageView* и *LinearLayout*.

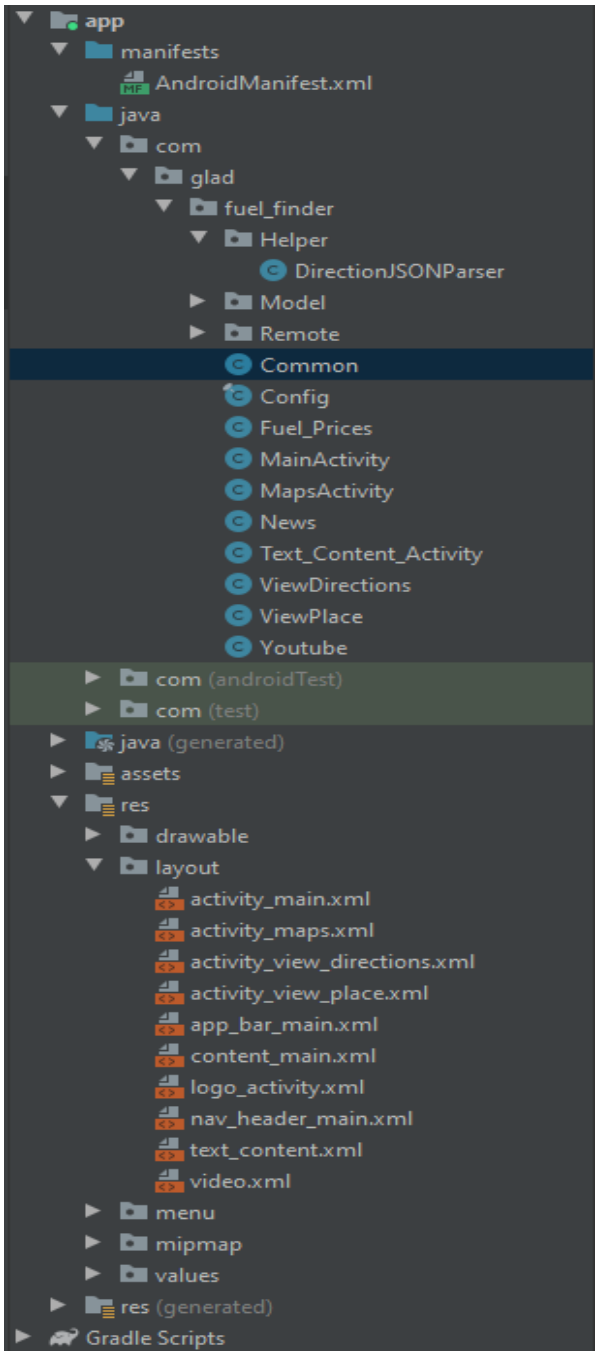


Рис. 7. Структура проекта

В папке *java/com.glad.fuel_finder* расположены классы, используемые в приложении. Основной класс *MainActivity* вызывает *nav_header_main.xml* который является главным меню приложения.

Класс *MapsActivity* отвечает за инициализацию и настройку Google Maps, отправку запроса на получение координат и построение маршрута.

В классе *Text_Content_Activity* производится вывод текста в зависимости от нажатой вкладки.

В классе *ViewPlaces* производится поиск ближайших АЗС и отображение дополнительной информации о них.

В классе *DirectionJSONParser* производится обработка JSON ответа от сервера с координатами.

В классах Fuel_Prices и News происходит отображение страницы с ценами на бензин и новостями.

Для удобства написания кода были созданы ресурсы *arrays.xml* и *strings.xml*. *Arrays.xml* (рисунок. 8) содержит в себе объекты которые поочередно заменяют друг друга в виджете *ListView*.

```
<?xml version="1.0" encoding="utf-8"?><resources>
<string-array name="home_array">
<item>27.11.2019</item>
<item>Добро пожаловать!</item>
<item>в</item>
<item>FuelFinder</item>
<item>v0.0.1</item>
</string-array>
<string-array name="na_array">
<item>Shell</item>
<item>Газпром</item>
<item>Лукойл</item>
<item>Роснефть</item>
```

Рис. 8. Код *arrays.xml*

Ресурс *strings.xml* содержит в себе заголовки и тексты, которые могут быть переданы в любой класс или лэйаут. На рисунке 9 представлена часть кода.

```
<resources>
<string name="seek_to">Jump To</string>
<string name="seek_to_hint">Seconds</string>
<string name="app_name">Fuel_Finder</string>
<string name="navigation_drawer_open">Open navigation drawer</string>
<string name="navigation_drawer_close">Close navigation drawer</string>
<string name="nav_header_title">FuelFinder</string>
<string name="nav_header_subtitle">android.studio@android.com</string>
<string name="nav_header_desc">Navigation header</string>
<string name="action_settings">Settings</string>

<string name="home">Главная страница</string>
<string name="gas">Об АЗС</string>
<string name="pri">Цены на бензин</string>
<string name="news">Новости</string>
<string name="history">Истории автолюбителей</string>
<string name="set">Настройки </string>
```

Рис. 9. Часть кода *strings.xml*

Для вывода разных данных была создана переменная *adapter*, которая очищает и заполняет *ListView* разными данными, таким образом сокращая написание кода, так как не приходится создавать несколько лэйаутов. Пример работы *adapter* представлен на рисунке 10.

```
else if (id == R.id.id_history)
        {
            toolbar.setTitle(R.string.history);
            array = getResources().getStringArray(R.array.history_ar-
ray);
            adapter.clear();
            adapter.addAll(array);
            adapter.notifyDataSetChanged();
```

Рис. 10. пример применения переменной *adapter*

Для вывода данных был создан класс *Text_Content_Activity*, который рассчитан на вывод информации.

3.2. Работа с API

Для работы Google Maps необходимо получить уникальный ключ API, который можно получить бесплатно для некоммерческих целей, имея платёжеспособный аккаунт Google. Для работы приложения потребуется отправить несколько запросов на сервера Google. Первый запрос отображает карту и текущее местоположение устройства. Второй запрос отображает на карте АЗС. Третий запрос загружает в приложение дополнительную информацию об этих точках. Четвёртый запрос получает в качестве ответа координаты между текущим местоположением и выбранной на карте АЗС, которые в приложении разбиваются на несколько точек, по которым потом проводится линия. После генерации ключа для инициализации карты он обозначается в AndroidManifest.xml для работы с Google maps [10, 11, 12].

При инициализации карты срабатывает вызов кнопки, по нажатию на которую приложение отправляет второй запрос, который найдёт ближайшие АЗС. Пример представлен на рисунке 11.

```
private String getUrl(double latitude, double longitude, String pla-
ceType) {

        StringBuilder googlePlacesUrl = new String-
Builder("https://maps.googleapis.com/maps/api/place/near-
bysearch/json?");
        googlePlacesUrl.append("location="+latitude+","+longitude);
        googlePlacesUrl.append("&radius="+10000);
        googlePlacesUrl.append("&type="+placeType);
        googlePlacesUrl.append("&sensor=true");
        googlePlacesUrl.ap-
pend("&key="+getResources().getString(R.string.browser_key));
        Log.d("getUrl",googlePlacesUrl.toString());
        return googlePlacesUrl.toString();
```

Рис. 11. Отправка запроса на получении координат о ближайших АЗС

Далее идёт обработка полученных данных и расстановка маркеров для перехода на экран с дополнительной информацией по полученным координатам [13], пример на рисунке 12.

```
private void nearByPlace(final String placeType) {
        String url = getUrl(latitude,longitude,placeType);
        mService.getNearByPlaces(url)
                .enqueue(new Callback<MyPlaces>() {
                    @Override
                    public void onResponse(Call<MyPlaces> call, Re-
sponse<MyPlaces> response) {
                        currentPlace = response.body();
                        if(response.isSuccessful())
                            for (int i=0;i<re-
sponse.body().getResults().length;i++){
                                MarkerOptions markerOptions = new
MarkerOptions();
                            Results googlePlace = re-
sponse.body().getResults()[i];
                            double lat =Double.parseDouble(google-
Place.getGeometry().getLocation().getLat())
                            double lng =Double.parseDouble(google-
Place.getGeometry().getLocation().getLng())
                            String placeName = googlePlace.getName();
                            String vicinity  = googlePlace.getVicin-
ity();
                            LatLng latLng = new LatLng(lat,lng);
                            markerOptions.position(latLng);
                            markerOptions.title(placeName);
                            if(placeType.equals("gas_station"))
                                   markerOptions.icon(Bit-
mapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_YEL-
LOW));
                            else
                                markerOptions.icon(BitmapDescriptor-
Factory.defaultMarker(BitmapDescriptorFactory.HUE_RED));
                            markerOptions.snippet(String.valueOf(i));
                            mMap.addMarker(markerOptions);
mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
mMap.animateCamera(CameraUpdateFactory.zoomTo(11));
```

Рис. 12. Обозначение ближайших АЗС на карте

При нажатии на маркер отправляется третий запрос, открывается отдельное окно, которое показывает название АЗС и её адрес, в этом же окне можно нажать на 2 кнопки. Первая откроет приложение Google Maps, в котором покажется вся доступная там информация о предприятии. Нажатие на вторую кнопку инициализирует построение маршрута. В этот момент происходит отправка четвёртого запроса[14,15]. Пример представлен в разделе «Приложение А» на рисунке А.1. При получении ответа на 4 запрос, он передается в класс DirectionJSONParser, в котором происходит обработка ответа, которая затем передается обратно в класс, реализующий маршрут. Пример работы DirectionJSONParser представлен на рисунке 14.

```
public class DirectionJSONParser {
    public List<List<HashMap<String,String>>> parse(JSONObject jOb-
ject){
        List<List<HashMap<String, String>>> routes = new
ArrayList<List<HashMap<String,String>>>() ;
        JSONArray jRoutes = null;
        JSONArray jLegs = null;
        JSONArray jSteps = null;
        Try {
            jRoutes = jObject.getJSONArray("routes");
            for(int i=0;i<jRoutes.length();i++){
                jLegs = (
(JSONObject)jRoutes.get(i)).getJSONArray("legs");
                List path = new ArrayList<HashMap<String, String>>();
                For(int j=0;j<jLegs.length();j++){
                    jSteps = ( (JSONObject)jLegs.get(j)).get-
JSONArray("steps");
                    for(int k=0;k<jSteps.length();k++){
                        String polyline = "";
                        Polyline =
(String)((JSONObject)((JSONObject)jSteps.get(k)).get("pol-
yline")).get("points");
                        List<LatLng> list =
```

Рис. 14. Работа класса *DirectionJSONParser*

3.3. Работа с XML шаблонами

Для каждого класса были созданы отдельные xml файлы. Для классов *MainActivity* был создан xml файл *activity_main*.

Для классов *Text_Content_Activity* и *MapsActivity* были созданы xml файлы *text_content* и *activity_maps* соответственно. Код *activity_maps* представлен на рисунке 15.

```
<fragment
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom ="true"
        android:id="@+id/bottom_navigation"
        app:itemBackground="@color/colorPrimary"
        app:itemIconTint="@drawable/nav_selector"
        app:itemTextColor="@drawable/nav_selector"
        app:menu="@menu/bottom_menu"
        />
```

Рис. 15. Часть кода оформления *activity_maps*

Был создан xml файл *text_content*. Код представлен на рисунке 16.

```
<LinearLayout
android:layout_width="match_parent"
android:orientation="vertical">
<LinearLayout
android:layout_width="match_parent"
android:layout_height="100dp"
android:background="@color/design_default_color_primary"
</LinearLayout>
<TextView
android:id="@+id/text_main_content"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:text="@string/home_1" />
```

Рис. 16. Код *text_content*

Для навигации в приложении было решено использовать выдвижное меню. Для этого был создан *nav_header_main.xml*.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="@dimen/nav_header_height"
android:background="@color/design_default_color_primary"
android:gravity="bottom"
android:orientation="vertical"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingBottom="@dimen/activity_vertical_margin"
android:theme="@style/ThemeOverlay.AppCompat.Dark">

<ImageView
android:id="@+id/imageView"
android:layout_width="90dp"
android:layout_height="90dp"
android:contentDescription="@string/nav_header_desc"
android:paddingTop="@dimen/nav_header_vertical_spacing"
app:srcCompat="@drawable/can" />
```

Рис. 17. Код *nav_header_main*

В activity_main реализован вызов навигационного меню. Код представлен на рисунке 18.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout xmlns:android="http://sche-
mas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/drawer_layout"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:fitsSystemWindows="true"
tools:openDrawer="start">

<include
layout="@layout/app_bar_main"
android:layout_width="match_parent"
android:layout_height="match_parent" />

<com.google.android.material.navigation.NavigationView
android:id="@+id/nav_view"
android:layout_width="wrap_content"
android:layout_height="match_parent"
android:layout_gravity="start"
android:fitsSystemWindows="true"
app:headerLayout="@layout/nav_header_main"
app:menu="@menu/activity_main_drawer" />

</androidx.drawerlayout.widget.DrawerLayout>
```

Рис.18. Код лэйаута *activity_main*

Вывод

Таким образом, на основе архитектуры системы было реализовано приложение. Были созданы и отредактированы лэйауты необходимые для приложения, а также классы, отвечающие за отображение информации, страниц, инициализации карты и работы с ней.

4. ТЕСТИРОВАНИЕ

Тестирование системы состоит из функционального тестирования мобильного приложения. Было проведено функциональное тестирование приложения, которое направлено на проверку реализуемости функциональных требований.

Для тестирования приложения был использован эмулятор устройства Pixel с уровнем API 24, эмулятор устройства Pixel 2 с уровнем API 26, а также устройство DIGMA City 653 с уровнем API 28.

Табл. 1. Функциональное тестирование

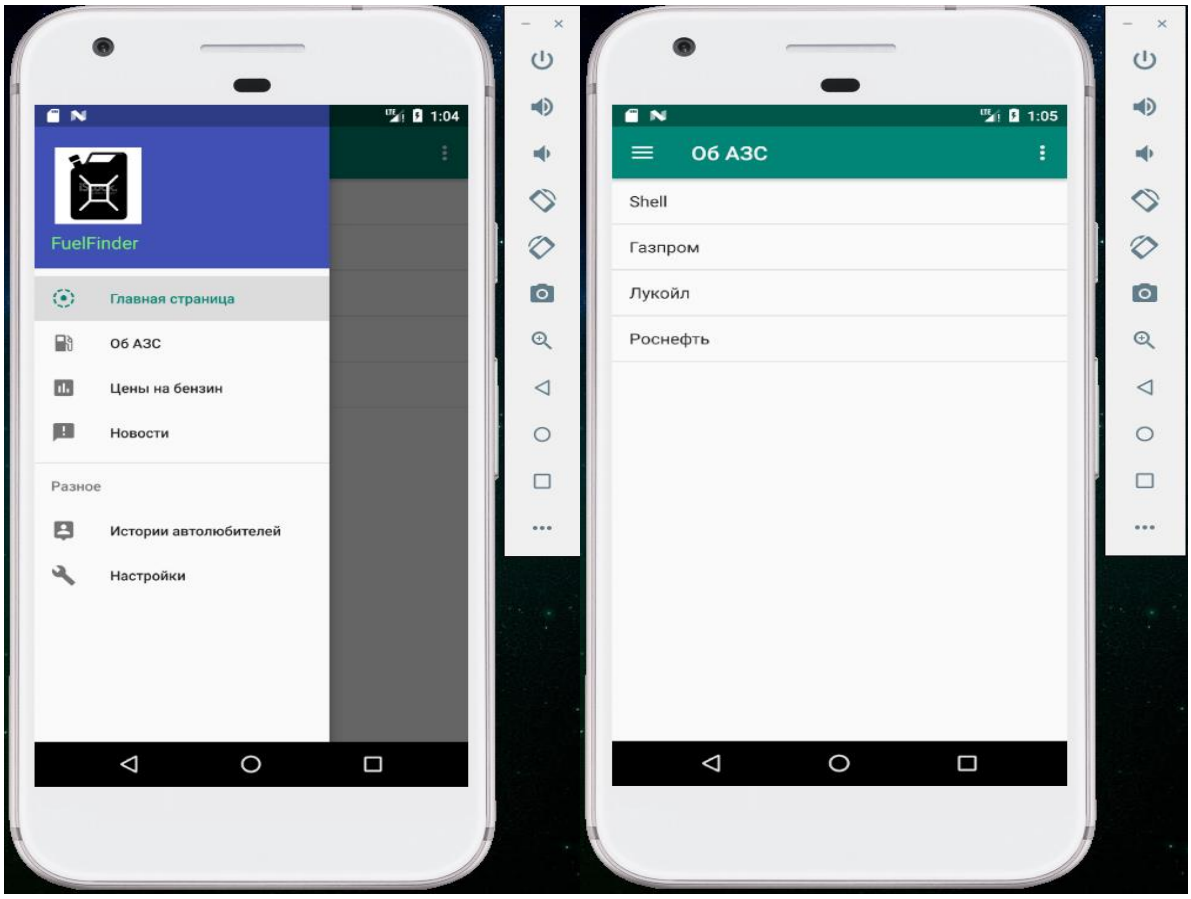
№	Название теста	Ожидаемый результат	Полученный результат	Тест пройден?
1	Отображение главного меню приложения	На экране отображается главное меню приложения с кнопками «Карта», «Об АЗС», «Цены на бензин», «Новости»	На экране отобразилось главное меню приложения с кнопками «Карта», «Об АЗС», «Цены на бензин», «Новости»	Да
2	Работа кнопки «Карта»	При нажатии кнопки «Карта» откроется карта с текущим местоположением устройства	После нажатия кнопки «Карта» открылась карта с текущим местоположением устройства	Да
3	Работа кнопки «Об АЗС»	При нажатии кнопки «Об АЗС» откроется окно с информацией об АЗС	После нажатия кнопки «Об АЗС» открывается окно с информацией об АЗС	Да
4	Работа кнопки «Цены на бензин»	При нажатии кнопки «Цены на бензин» отобразится страница с актуальными ценами на бензин по городу.	После нажатия кнопки «Цены на бензин» отображается страница с актуальными ценами на бензин по городу.	Да

5	Работа кнопки «Новости»	При нажатии кнопки «Новости» отобразится страница с новостной лентой.	После нажатия кнопки «Новости» отображается страница с новостями.	Да

Окончание табл. 1

6	Работа кнопки «Gas_station»	При нажатии кнопки «Gas_station» на карте отображаются маркеры на ближайших АЗС в установленном радиусе от устройства.	После нажатия кнопки «Gas_station» на карте отобразились маркеры на ближайших АЗС в установленном радиусе от устройства.	Да
7	Отображение подробной информации о выбранной ближайшей АЗС	При нажатии на маркер ближайшей АЗС открывается экран с названием, адресом, текущим статусом работы, и двумя кнопками: «SHOW ON MAP» и «SHOW DIREC-TIONS»	После нажатия на маркер ближайшей АЗС открыдся экран с названием, адресом, текущим статусом работы, и двумя кнопками: «SHOW ON MAP» и «SHOW DIREC-TIONS»	Да
8	Работа кнопки «SHOW ON MAP»	При нажатии кнопки «SHOW ON MAP» открывается приложение Google Maps с максимально подробной информацией об АЗС	После нажатия кнопки «SHOW ON MAP» открылось приложение Google Maps с максимально подробной информацией об АЗС	Да
9	Работа кнопки «SHOW DIREC-TIONS»	При нажатии кнопки «SHOW DIREC-TIONS» строится маршрут до выбранной ближайшей АЗС	После нажатия кнопки «SHOW DI-RECTIONS» построился маршрут до выбранной ближайшей АЗС	Да

Были протестированы кнопки, расположенные на выдвижном меню приложения на эмуляторе устройства Pixel с уровнем API 24 (рисунок 19-а), отображается список опций, а также список на 2-ой вкладке. (рисунок 19-б).



а) б)

Рис. 19. Скриншот основного экрана приложения с боковым меню (а) отображением ListView (б)

Тест меню эмуляторе устройства Pixel 2 с уровнем API 26 представлен на рисунке 20.

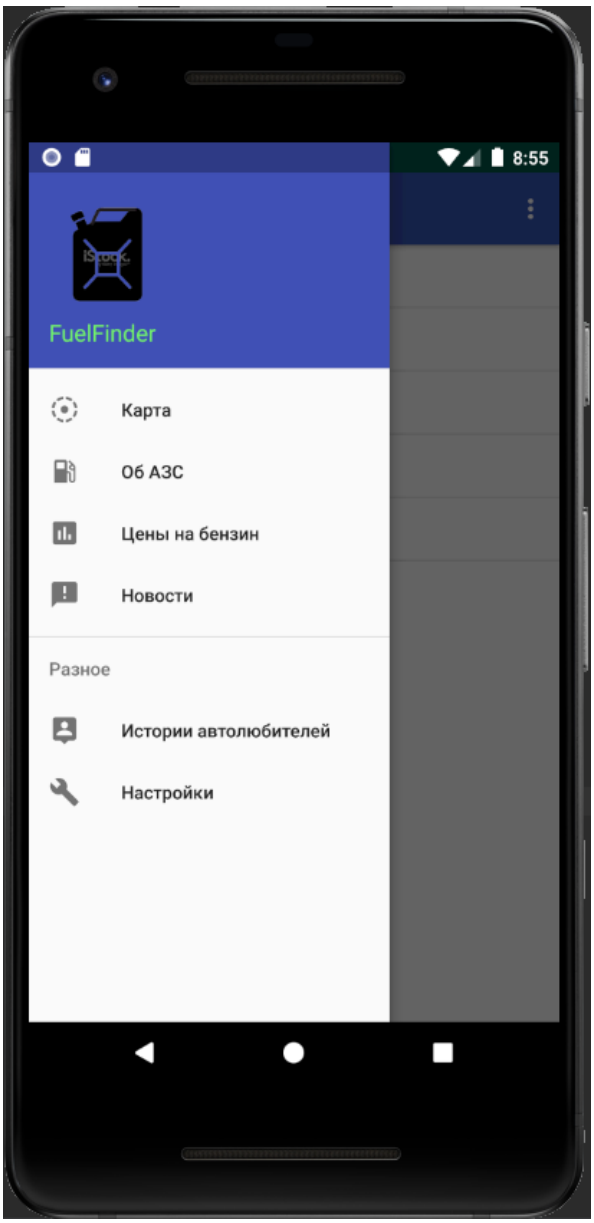


Рис.20. Запуск меню на устройстве другой версии Android OS

Была протестирована карта устройстве DIGMA City 653 с уровнем API 28. При нажатии на кнопку “Карта” происходит переход на виджет Google Maps. Также при нажатии на кнопку «Gas_station» находятся и отображаются ближайшие АЗС. Пример представлен на рисунке 21.

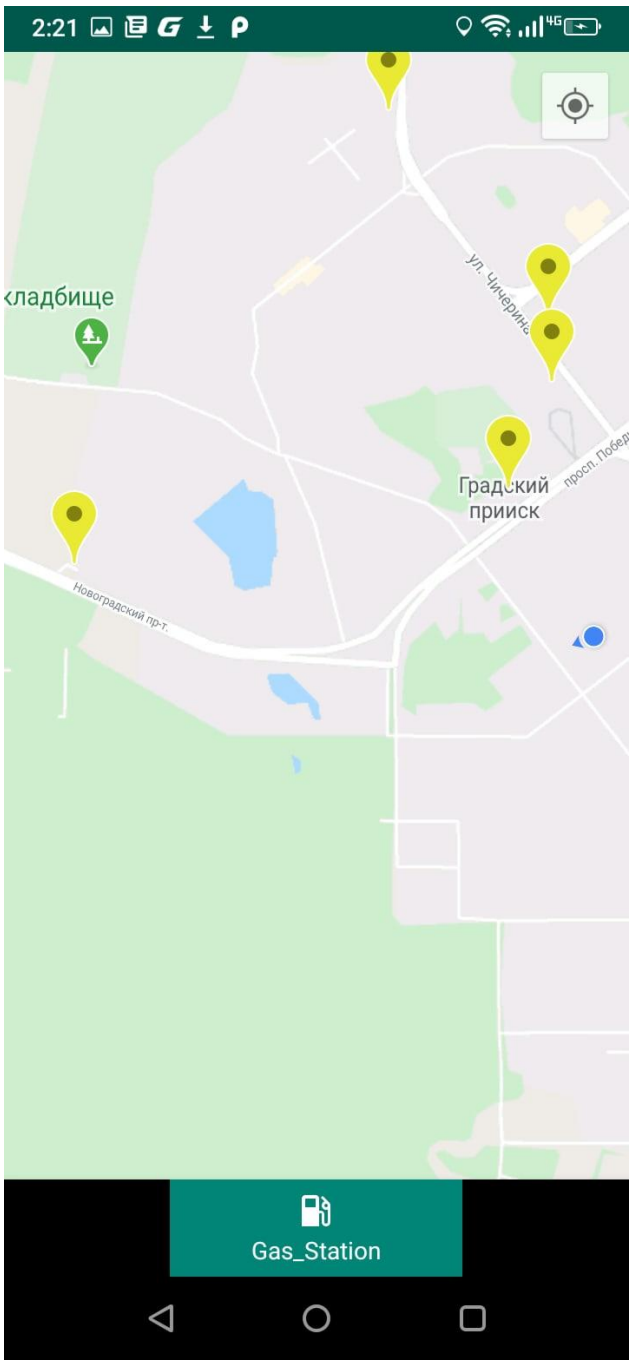


Рис. 21. Скриншот экрана приложения с картой

Был протестирован экран с информацией о ближайшей АЗС на рисунке 22.

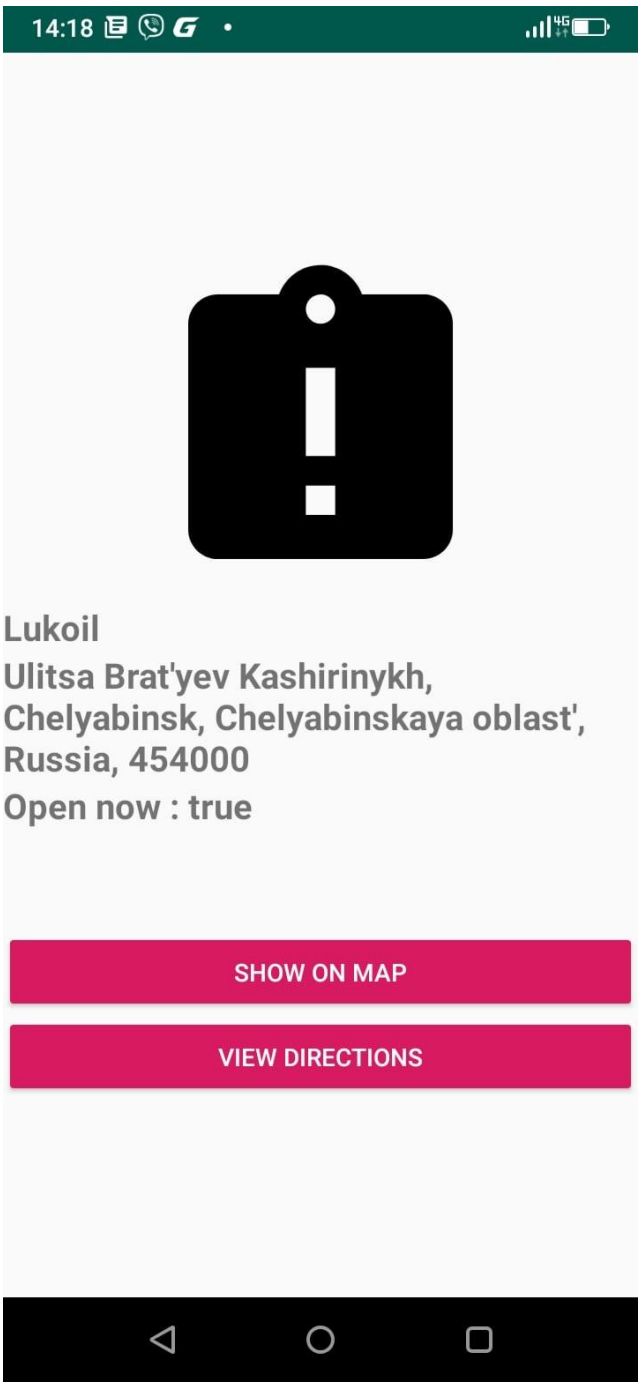


Рис. 22. Скриншот экрана информацией об АЗС

Также была протестирована кнопка «SHOW ON MAP» на рисунке 23.

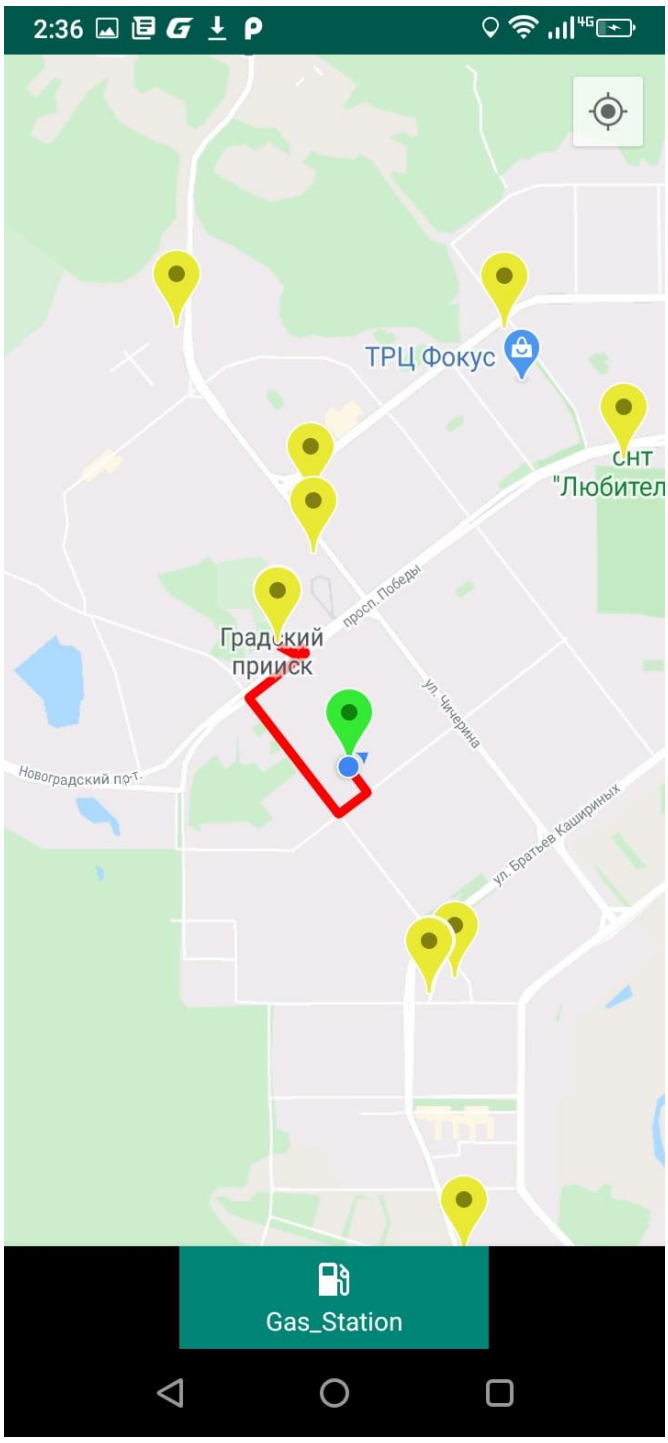


Рис.23. Скриншот с построенным маршрутом

На рисунке 24 представлена работа кнопки «Цены на бензин» с горизонтальной ориентацией работы приложения

Рис.24. Скриншот экрана с ценами на бензин

Был протестирован экран с новостным порталом на рисунке 25, также на рисунке 26 было проведено тестирование с поиском ближайших АЗС относительно местоположения в другом городе.

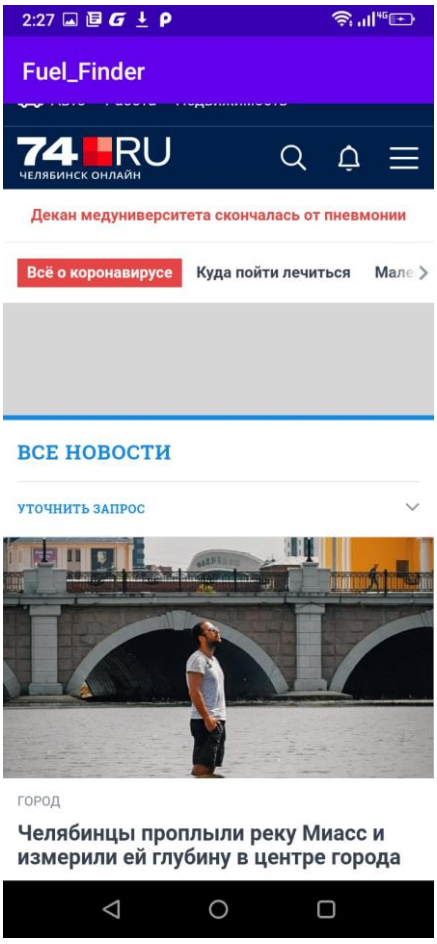


Рис.25. Экран новостей

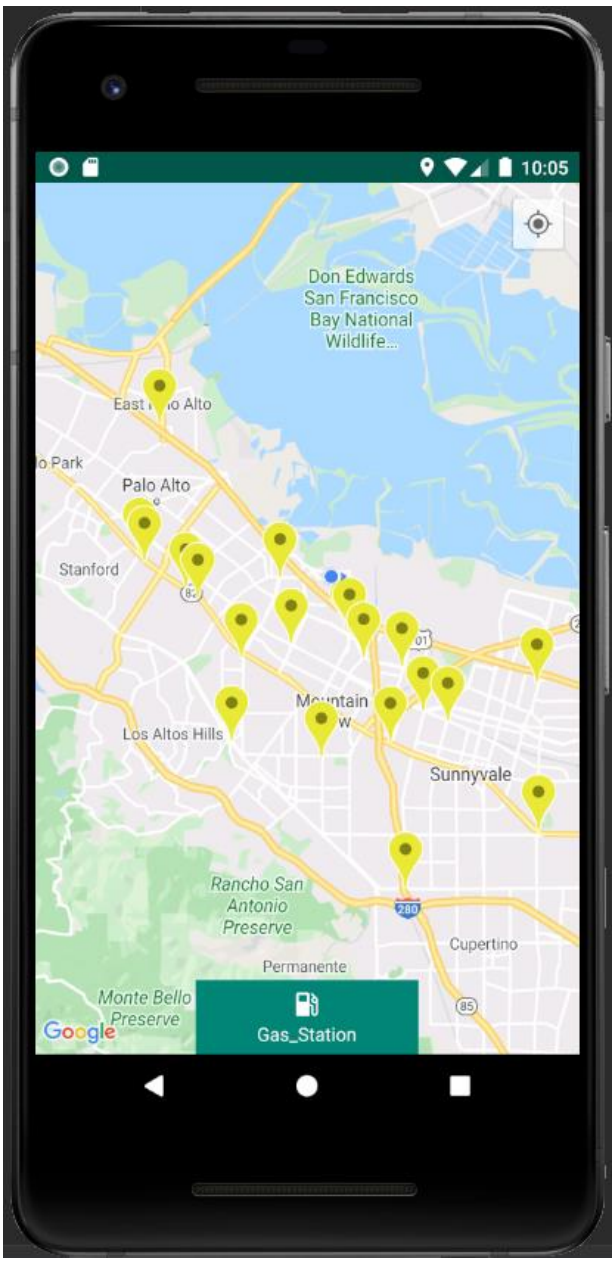


Рис.26 Экран теста поиска в другом городе

ЗАКЛЮЧЕНИЕ

В рамках данной работы было разработано геоинформационное мобильное Android-приложение для поиска автозаправочных станций. При этом были решены следующие задачи:

1) проведен анализ предметной области;
2) проведен обзор аналогов на рынке мобильных приложений;
3) определены требования к системе и разработаны варианты ее использования;
4) разработана структура мобильного приложения;
5) разработан метод построения маршрута между текущим местоположением и ближайшей АЗС;
6) разработано мобильное приложение на платформе Android;
7) проведено тестирование системы.

Планируется дальнейшее развитие проекта, включающее в себя следующие пункты:

1) разработка базы данных для хранения координат автозаправочных станций для пользования приложением в оффлайн режиме;
2) пополнение базы данных новыми координатами автозаправочных станций.

ЛИТЕРАТУРА

1. Медникс З. Программирование под Android. 2-е изд. /Дорнин Л., Мик Б., Накамура М. — СПб.: Питер, 2013. — 560 с.: ил.

2. П. Дейтел. Android для программистов: создаём приложения. /Х. Дейтел, Э. Дейтел — СПб.: Питер, 2013. —560 с.: ил.

3. Харди Б. Android. Программирование для профессионалов. 2-е изд. /Филлипс Б., Стюарт К., Марсикано К. — СПб.: Питер, 2016. — 640 с.: ил.

4. Васильев А.Н. Java. Объектно-ориентированное программирование, 2011. —397 с.

5. Статья «Android Studio IDE от Google» [Электронный ресурс] URL: http://wnfx.ru/android-studio-ide-ot-google/ (дата обращения 20.05.2020).

6. Статья «Внедрение платформы разработки и исполнения приложений «Radixware»» [Электронный ресурс] URL: http://www.scienceforum.ru/2016/1757/19149 (дата обращения 20.05.2020).

7. Статья «CardView (Карточка)» [Электронный ресурс] URL: http://developer.alexanderklimov.ru/android/views/cardview.php (дата обращения 20.05.2020).

8. Статья «Знакомство с элементом RecyclerView» [Электронный ресурс] URL: https://devcolibri.com/unit/урок-10-работа-с-recyclerview-на-примере-tweetsrecyclerview-2 (дата обращения 20.05.2020).

9. Статья «Using the YouTube API» [Электронный ресурс] URL: https://www.sitepoint.com/using-the-youtube-api-to-embed-video-in-an-android-app/ (дата обращения 20.05.2020).

10. Статья «Урок 139. Google maps. Создание и настройка проекта. Карта, камера, события» [Электронный ресурс] URL: https://star-tandroid.ru/ru/uroki/vse-uroki-spiskom/306-urok-139-google-maps-sozdanie-i-nastrojka-proekta-karta-kamera-sobytija.html (дата обращения 20.05.2020).

11. Статья «Google MAPs API в android или как работать с картами быстрее» [Электронный ресурс] URL: https://m.habr.com/ru/post/341548/ (дата обращения 25.11.2019).

12. Статья «30 Android-библиотек и инструментов, которые не должны пройти мимо вас в 2018 году» [Электронный ресурс] URL: https://m.habr.com/ru/post/431400/ (дата обращения 20.05.2020).

13. Статья «Объекты на карте» [Электронный ресурс] URL: https://developers.google.com/maps/documentation/android-sdk/map?hl=ru (дата обращения 20.05.2020).

14. Статья «Маршруты на картах Google в Android-приложении — некоторые уточнения» [Электронный ресурс] URL: https://habr.com/ru/post/275019/ (дата обращения 20.05.2020).

15. Статья «How to Draw Route in Google Maps API V2 from my location [duplicate]» [Электронный ресурс] URL: https://stackoverflow.com/questions/16262837/how-to-draw-route-in-google-maps-api-v2-from-my-location (дата обращения 20.05.2020).

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А. – Листинг реализации маршрута

Листинг А.1. Реализация маршрута

```
private class ParserTask extends AsyncTask<String, Integer,
List<List<HashMap<String, String>>>> {

        @Override
        Protected List<List<HashMap<String, String>>>
doInBackground(String... jsonData) {

            JSONObject jObject;
            List<List<HashMap<String, String>>> routes = null;

            Try {
                JObject = new JSONObject(jsonData[0]);
                DirectionsJSONParser parser = new
DirectionsJSONParser();

                Routes = parser.parse(jObject);
            } catch (Exception e) {
                e.printStackTrace();
            }
            Return routes;
        }

        @Override
        Protected void onPostExecute(List<List<HashMap<String,
String>>> result) {
            ArrayList<LatLng> points = new ArrayList<LatLng>();;
            PolylineOptions lineOptions = new PolylineOptions();;
            MarkerOptions markerOptions = new MarkerOptions();

            For (int i = 0; i < result.size(); i++) {
                Points = new ArrayList();
                LineOptions = new PolylineOptions();

                List<HashMap<String, String>> path = result.get(i);

                For (int j = 0; j < path.size(); j++) {
                    HashMap<String, String> point = path.get(j);
```

```
                    double lat =
Double.parseDouble(point.get("lat"));
                    double lng =
Double.parseDouble(point.get("lng"));
                    LatLng position = new LatLng(lat, lng);

                    Points.add(position);
                }

                lineOptions.addAll(points);
                lineOptions.width(12);
                lineOptions.color(Color.RED);
                lineOptions.geodesic(true);

            }

            MMap.addPolyline(lineOptions);
        }
    }

    private String getDirectionsUrl(LatLng origin, LatLng dest) {

        String str_origin = "origin=" + origin.latitude + "," +
origin.longitude;

        String str_dest = "destination=" + dest.latitude + "," +
dest.longitude;

        String sensor = "sensor=false";
        String mode = "mode=driving";
        String parameters = str_origin + "&" + str_dest + "&" +
sensor + "&" + mode;

        String output = "json";

        String url = "https://maps.googleapis.com/maps/api/direc-
tions/"+output+"?"+parameters + "&key=R.string.browser_key";


        return url;
    }
```

```
    private String downloadUrl(String strUrl) throws IOException {
        String data = "";
        InputStream iStream = null;
        HttpURLConnection urlConnection = null;
        try {
            URL url = new URL(strUrl);

            urlConnection = (HttpURLConnection) url.openConnection();

            urlConnection.connect();

            iStream = urlConnection.getInputStream();

            BufferedReader br = new BufferedReader(new In-
putStreamReader(iStream));

            StringBuffer sb = new StringBuffer();

            String line = "";
            while ((line = br.readLine()) != null) {
                sb.append(line);
            }

            data = sb.toString();

            br.close();

        } catch (Exception e) {
            Log.d("Exception", e.toString());
        } finally {
            iStream.close();
            urlConnection.disconnect();
        }
        return data;
    }
```