

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

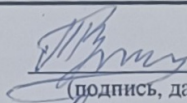
Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

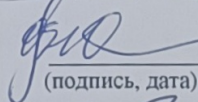
КАРТОГРАФИЧЕСКИЙ СЕРВИС ПОИСКА АЗС

Студент


(подпись, дата) 2.06.2025

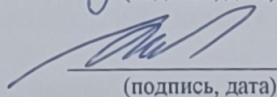
Тюкалов Василий Евгеньевич
(ФИО полностью)

Руководитель


(подпись, дата) 3.06.2025

Юрова О.В.
(фамилия, инициалы)

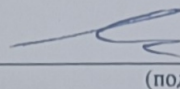
Нормоконтролёр


(подпись, дата) 3.06.2025

Бычков А.С.
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 28.05.2025 г. № 16)

Заведующий кафедрой


(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

02.06.2025 г. № 23)

Секретарь ГЭК



Е.И. Гурин
(подпись)

(фамилия, инициалы)

Пенза 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«26» 05 2025 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Тюкалов Василий Евгеньевич гр. 21ВВП2 факультета вычислительной техники

2. Тема работы «Картографический сервис поиска АЗС» для ОС Android

Тема утверждена приказом ПГУ №_601/о_____ от "6" мая_____ 2025 г.

3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ ПГУ Юрова О.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать мобильное приложение «Картографический сервис поиска АЗС», позволяющее отображать АЗС и строить маршруты. Язык программирования Kotlin, среда разработки IntelliJ IDEA. Программа должна обладать интуитивно понятным интерфейсом и использовать следующие технологии:

1. Yandex Mapkit

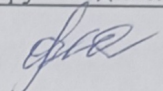
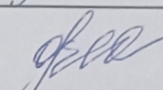
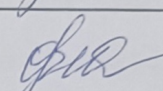
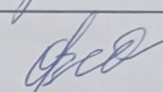
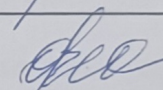
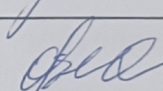
2. Асинхронное программирование

3. Room Database

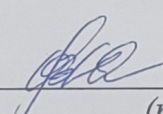
5. Перечень подлежащих разработке вопросов

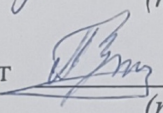
Требуется произвести анализ предметной области: изучить существующие аналоги, изучить способы проектирования и разработки мобильного приложения. Выбрать язык программирования, среду разработки и базу данных.

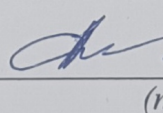
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	45	6.05.2025 – 10.05.2025	
2	Анализ требований	16	11.05.2025 – 12.05.2025	
3	Выбор технологий, применяемых при проектировании и разработке	24	13.05.2025 – 15.05.2025	
4	Описание алгоритмов	24	16.05.2025 – 18.05.2025	
5	Разработка мобильного приложения	80	19.05.2025 – 24.05.2025	
6	Оформление пояснительной записки	16	29.05.2025 – 30.05.2025	

Дата выдачи задания "6" мая 2025 г.

Руководитель бакалаврской работы  06.05.25 Юрова О.В.
(подпись, дата)

Задание к исполнению принял студент  06.05.2025 Тюкалов В.Е.
(подпись, дата)

Работу к защите допустить
Декан факультета  06.06.2025 Фионова Л.Р.
(подпись, дата)

Реферат

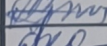
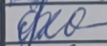
Бакалаврская работа содержит 114 листов, 33 рисунка, 1 таблица, 8 источника, 3 приложения.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ТЕСТИРОВАНИЕ, КАРТОГРАФИЧЕСКИЙ СЕРВИС, KOTLIN, ANDROID

Цель работы – разработать мобильное приложение, позволяющее искать АЗС.

Мобильное приложение должно позволять пользователям определить местоположение, строить маршруты до станций относительно пользователя или по адресам. Для взаимодействия с приложением требуется авторизация.

В конечном итоге было разработано мобильное приложение на языке программирования Kotlin, которое удовлетворяет всем поставленным задачам. Был реализован картографический сервис на основе данных, полученных от сервисов «Яндекс». Само приложение работает на устройствах с ОС Android и разрабатывалось в специальной среде разработки IntelliJIDEA.

					ПГУ1.090301.47.001 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата	Картографический сервис поиска АЗС Пояснительная записка	Лит.	Лист	Листов	
Разраб.		Тюкалов В.Е.		2.06.25					
Провер.		Юрова О.В.		3.06.25			2	118	
Н. Контр		Бычков А.С.		3.06.25		ФВТ, каф. ВТ гр. 21ВВП2			
Утверд.		Митрохин М.А.		3.06.25					

Перечень сокращений и обозначений

В данной работе применяются следующие сокращения и обозначения

AЗС – автомобильная заправочная станция

ПК – персональный компьютер

GUI – (от англ. Graphical User Interface) – графический интерфейс пользователя

SQL – (от англ. Structured Query Language) – язык структурированных запросов

UML – (от англ. Unified Modeling Language) – унифицированный язык моделирования

					<i>ПГУ1.090301.47.001 ПЗ</i>	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Содержание

Перечень сокращений и обозначений	5
Введение.....	8
1 Приложение «Картографический сервис поиска АЗС»	9
1.1 Анализ темы	9
1.2 Анализ аналогов	9
1.2.1 «Яндекс.Заправки»	9
1.2.2 «Топливо»	11
1.3 Постановка задачи	16
2 Выбор инструментов разработки	17
2.1 Выбор языка программирования.....	17
2.1.1 Язык программирования «Java».....	17
2.1.2 Язык программирования «Kotlin».....	17
2.1.3 Язык программирования «C++».....	17
2.1.4 Вывод	18
2.2 Выбор базы данных	18
3 Технологии разработки.....	19
4 Реализация приложения	21
4.1 Представление	25
4.2 Модель представления	26
4.2.1 Модель представления основного фрагмента	27
4.2.2 Модель представления авторизации.....	29
4.3 Модель.....	30
5 Тестирование	32
5.1 Тестирование авторизации.....	32
5.2 Построение маршрута до станции	34
5.3 Построение и отмена маршрута по адресам	36
5.4 Маршрутизация через другие сервисы	40
5.5 Выбор отображения различных типов маркеров.....	42
6 Безопасность жизнедеятельности.....	45

6.1 Техника безопасности при работе с ПК	45
6.2 Техника безопасности при работе с мобильными устройствами	46
Заключение	47
Список использованных источников	48
Приложение А – UML – диаграммы	49
Приложение Б – Листинг программы	54
Приложение В – Презентация.....	110

Введение

На сегодняшний день существует существенная разница в стоимости топлива на различных АЗС России, также качество топлива отличается. На АЗС станциях одной компании показатель цены тоже может быть различным, также существуют акции и скидки на конкретных АЗС.

Этот продукт разрабатывается для транспортных компаний, которые в целях экономии, заключают договора с различными сетями АЗС. В связи с этим существует перспектива создание собственной базы АЗС. Информирование сотрудников компании, в частности водителей грузовых транспортных средств, осуществляется с помощью мобильного приложения, которое, используя средства спутниковой связи, относительно местоположения водителя, показывает ближайшие АЗС для дозаправки транспортного средства.

Приложение разрабатывалось под операционную систему Android. Она занимает лидирующие позиции в сфере мобильных ОС, более 70% рынка по данным Statcounter Global Stats [1].

					ПГУ1.090301.47.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

1 Приложение «Картографический сервис поиска АЗС»

1.1 Анализ темы

Водителю транспортной логистической компании оперирует следующими данными:

- собственное местоположение: позволяет ориентироваться на карте и прокладывать маршруты;
- места загрузки/выгрузки товара: конечный точки, куда строиться маршрут
- местоположение АЗС: ориентирование, относительно собственного местоположения, для дозаправки грузового транспортного средства;
- места отдыха, где водитель может восстановить силы и продолжить рейс.

Все эти данные не могут быть отражены в популярных картографических сервисах «Яндекс.Карты» или «Google Maps». В связи, с этим компании вынуждены обращаться к разработчикам мобильного ПО для решения поставленных задач.

1.2 Анализ аналогов

В настоящее время существует ряд картографических сервисов:

- «Яндекс.Заправки»;
- Сервис Т-Банк «Топливо»;
- «АЗС Роснефть».

Был проведен анализ и выявлены слабые и сильные стороны каждого из приложений.

1.2.1 «Яндекс.Заправки»

Приложение отображает АЗС на карте, и предоставляет возможность бесконтактной оплаты для заправки автомобилей на АЗС. Интерфейс представлен на рисунках 1 и 2.

					ПГУ1.090301.47.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

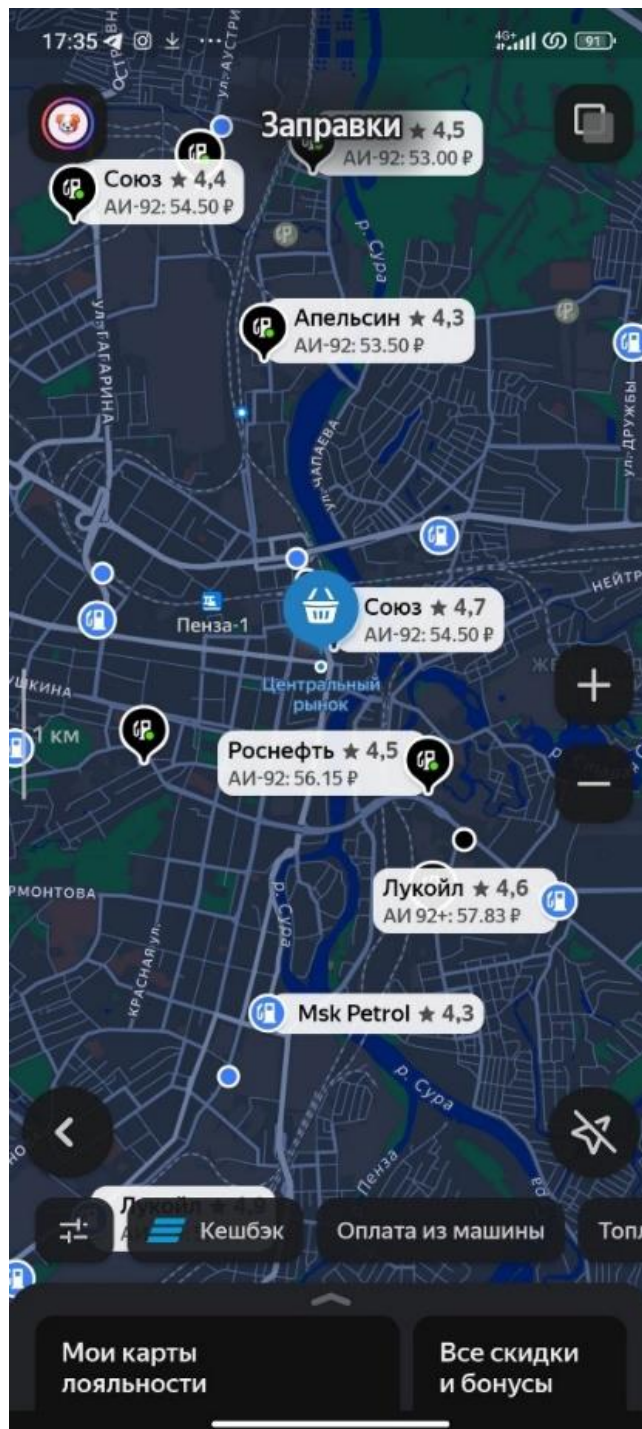


Рисунок 1 – Приложение «Яндекс.Заправки»

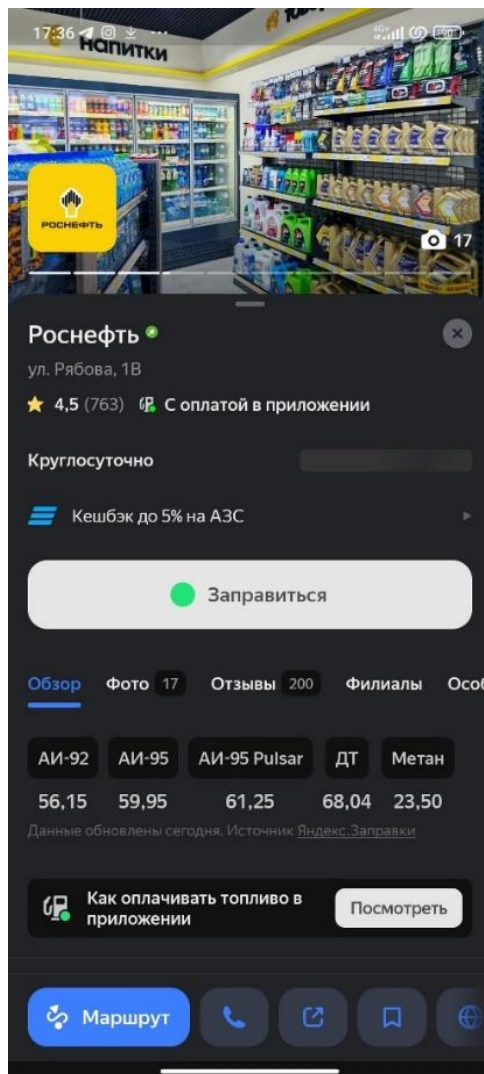


Рисунок 2 – Информация о АЗС

К достоинствам можно отнести:

- есть возможность получения местоположения пользователя;
- можно проложить маршрут до интересующей АЗС;
- понятный графический интерфейс;
- наличие информации о конкретной АЗС.

К недостаткам относится отсутствие возможности добавить свою базу АЗС.

1.2.2 «Топливо»

Картографический сервис Т-Банк для поиска ближайших АЗС. Интерфейс представлен на рисунках 3 и 4.

– К достоинствам можно отнести:

					ПГУ1.090301.47.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		11

- есть возможность узнать информацию о конкретной станции;
- есть возможность получения местоположения пользователя;
- можно проложить маршрут до интересующей АЗС;
- понятный графический интерфейс.

К недостаткам относится отсутствие возможности добавить свою базу АЗС.

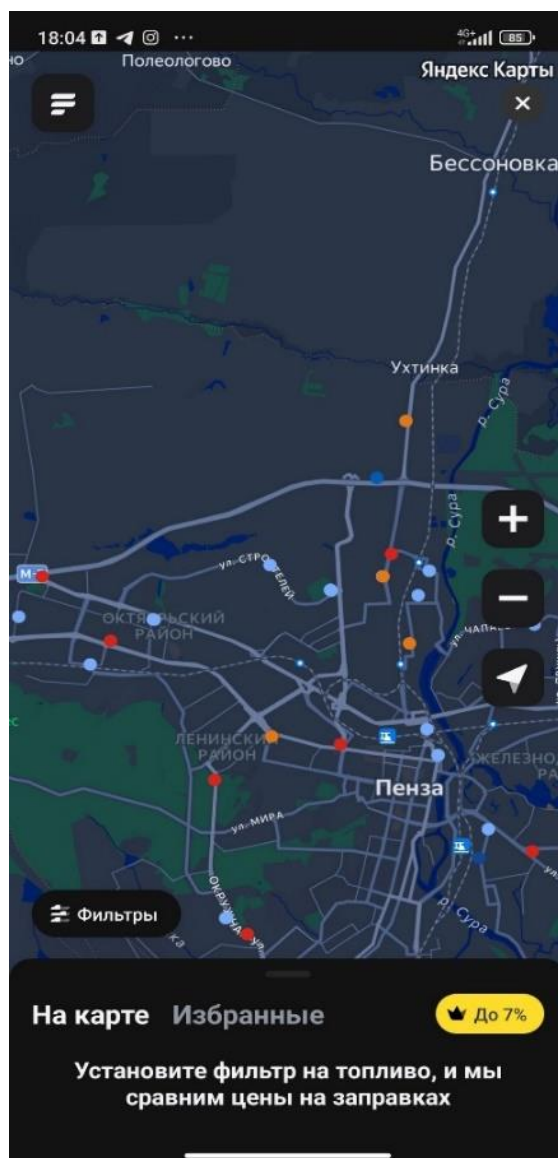


Рисунок 3 – Сервис Т-Банка «Топливо»

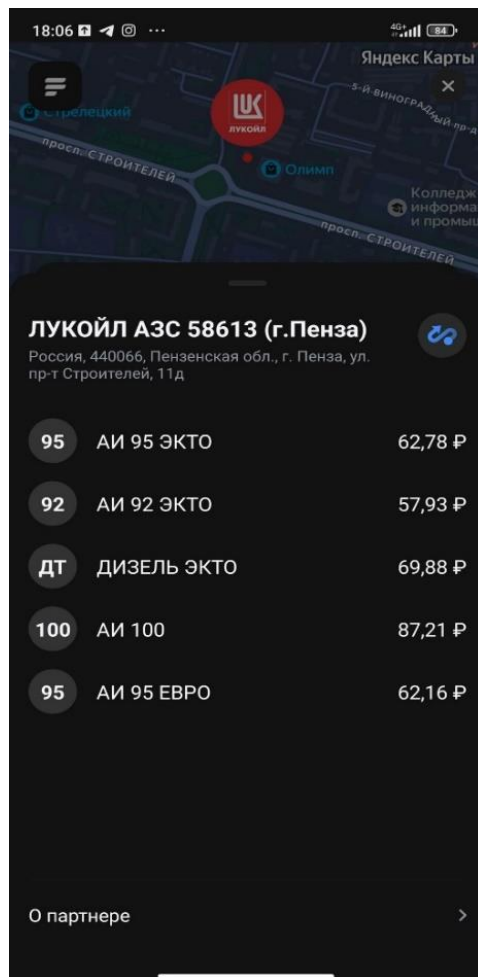


Рисунок 4 – Информация о АЗС

1.2.3 АЗС «Роснефть»

Картографический сервис создан для отображения АЗС компании Роснефть. Интерфейс представлен на рисунках 5 и 6.

К достоинствам можно отнести:

- есть возможность узнать информацию о конкретной станции;
- есть возможность получения местоположения пользователя;
- можно проложить маршрут до интересующей АЗС;
- понятный графический интерфейс.

К недостаткам относится отсутствие возможности добавить свою базу АЗС.

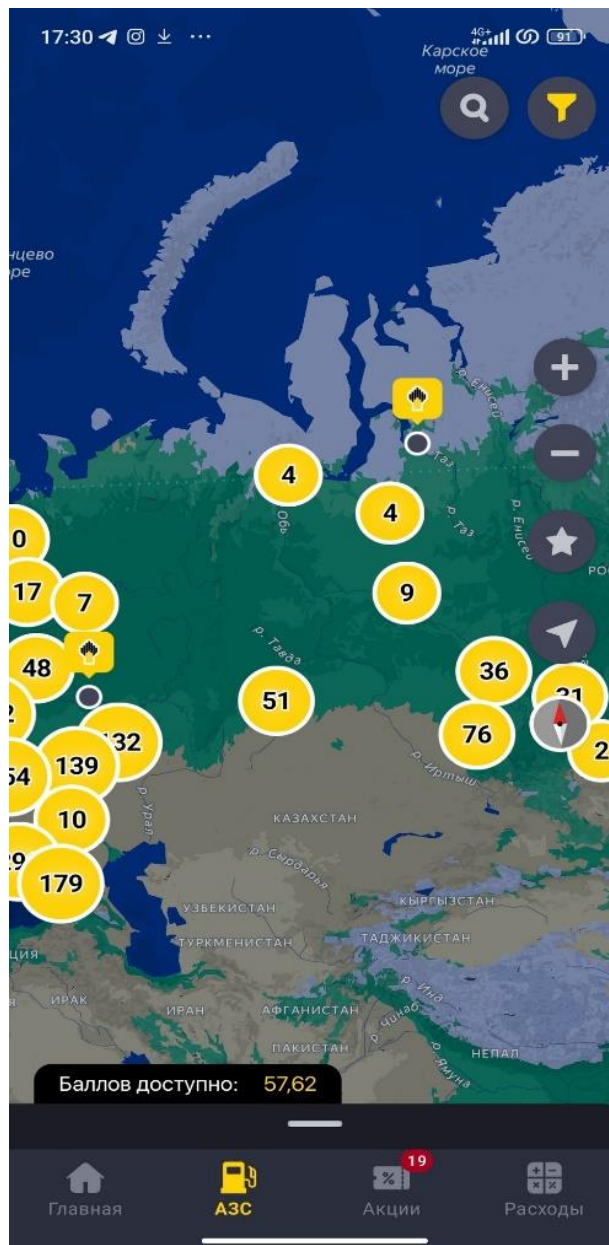


Рисунок 5 – Приложение «АЗС Роснефть»

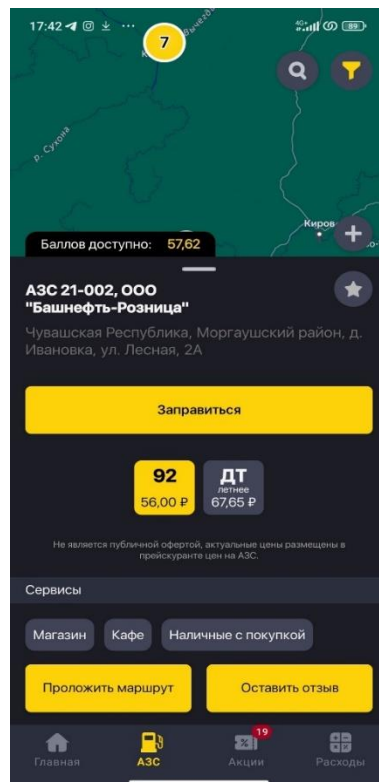


Рисунок 6 – Информация о АЗС

1.2.4 Вывод

Проанализировав ряд приложений, представленных в таблице 1, можно отметить, что ни одно из них не имеет возможность для взаимодействия со своей базой точек на карте. Поэтому для решения поставленных задач требуется собственный картографический сервис.

Таблица 1 – Достоинства и недостатки картографических сервисов

	Яндекс.Заправки	АЗС Роснефть	Топливо
Ин-фа о АЗС	+	+	+
Местоположение	+	+	+
Маршрутизация	+	+	+
Понятный GUI	+	+	+
Своя база данных	-	-	-

1.3 Постановка задачи

Необходимо разработать приложение для телефонов на базе операционной системы Android, показывающее ближайшие АЗС, относительно местоположения водителя. В приложении должно быть реализовано:

- регистрация пользователя;
- построение маршрута;
- обновление данных из базы;
- информационная панель интересующей АЗС;
- интуитивно понятный интерфейс

Минимальные системные требования:

- операционная система Android 10;
- наличие у устройства спутниковой системы навигации;
- наличие интернета.

					<i>ПГУ1.090301.47.001 ПЗ</i>	Лист
						16
Изм.	Лист	№ докум.	Подпись	Дата		

2 Выбор инструментов разработки

Для создания программы требуется выбрать правильный инструмент разработки, который будет соответствовать интересам бизнеса: качество и скорость создания программного обеспечения.

2.1 Выбор языка программирования

В зависимости от типа задачи, есть ряд языков, которые применяются для разработки программного обеспечения под операционную систему Android.

2.1.1 Язык программирования «Java»

Java зародился в 1991 году в компании Sun Microsystems. Согласно индексу ТЮВЕ, Java занимает четвертую строчку среди самых популярных языков программирования [2], имеет большое сообщество разработчиков и много различных инструментов для разработки, что делает этот язык хорошим выбором для разработки мобильных приложений.

2.1.2 Язык программирования «Kotlin»

Назван в честь российского острова Котлин в Финском заливе. Был основан под руководством Андрея Бреслава и продемонстрирован в июле 2011 года [3]. 17 мая на конференции Google I/O команда Android анонсировала, что Kotlin будет основным языком для разработки Android-приложений [4]. В связи с этим, выбор языка Kotlin является отличным решением для разработки программного обеспечения под операционную систему Android.

2.1.3 Язык программирования «C++»

Занимает второе по популярности место среди языков программирования [2]. Предоставляет инструменты для создания наиболее оптимизированных решений таких как игры или работа с программно-аппаратной частью. Но не подходит для решения более простых задач, так как является не безопасным языком, накладывая на разработчика много ответственности за управление памятью, что в свою очередь сказывается на

скорости разработки, не удовлетворяя основным требованиям бизнеса: быстрота и эффективность разработки программного обеспечения.

2.1.4 Вывод

В ходе анализа был выбран язык Kotlin, так как он обладает возможностями языка Java и его синтаксис позволяет писать более компактный код.

2.2 Выбор базы данных

Для работы с приложением в режиме офлайн требуется кэшировать информацию о АЗС, решением выступало использование локальной базы данных SQLite, так как она является библиотекой и становится составной частью программы, что в свою очередь ускоряет процесс взаимодействия с базой данных. Помимо этого, компания Google предоставляет возможность взаимодействия с SQLite с помощью персистентной библиотеки Room, которая обладает рядом преимуществ, в сравнении с прямым использованием программного интерфейса SQLite [5]:

- проверка SQL-запросов во время компиляции;
- удобные аннотации, которые минимизируют повторяющийся и подверженный ошибкам шаблонный код;
- оптимизированные пути миграции базы данных.

3 Технологии разработки

Для разработки приложения был выбран архитектурный паттерн Model-ModelView-View, позволяющий разделить логику графического пользовательского интерфейса от бизнес-логики. Был изобретен архитекторами компании Microsoft Кеном Купером и Тедом Питерсом в 2005 году [6].

Для увеличения быстродействия и комфортного пользования приложением требуется выполнять задачи в фоновых потоках. Одним из решений этой проблемы выступает реактивное программирование RxJava.

Для взаимодействия с картами требуется выбрать библиотеку, позволяющую использовать картографические данные, и предоставлять следующие возможности:

Необходимый и достаточный функционал для осуществления полноценной городской навигации:

- интуитивный интерфейс, следование трендам в дизайне;
- доступ к актуальной справочной информации с постоянными обновлениями;
- возможность работать офлайн, желательно использовать для этого маленький объем памяти.

Такие технологии предоставляют ряд компаний: Google, Яндекс, 2GIS. Ниже будут рассмотрены сервисы каждой из компаний.

Google MapKit – это крупнейший сервис, содержащий картографические данные со всего мира, на данный момент времени не подходит российским компаниями в связи с ограничениями использования Google-сервисов.

Yandex Mapkit – это популярный сервис в СНГ регионе, ориентирован на отечественных пользователей, предоставляет большую базу картографических данных, на данный момент времени имеет преимущество перед Google Mapkit, так как не ограничивает свои сервисы для российских граждан.

2GIS Mapkit - это популярный сервис в СНГ-регионе, но, в отличие от Yandex карт, является менее информативным. Предоставляет возможность построения маршрутов без доступа в интернет.

					ПГУ1.090301.47.001 ПЗ	Лист
						20
Изм.	Лист	№ докум.	Подпись	Дата		

4 Реализация приложения

Создание приложения начинается с идеи, выбора технологий разработки и описания функционала. Программа должна включать в себя следующее:

- возможность получения данных о местоположении;
- создания маршрутов из относительно собственного местоположения и из точек;
- полученных с помощью поиска их на карте;
- получение местоположения и краткой информации о конкретной АЗС.

Приложение разделено на три основных модуля:

- View – отвечает за отображение объектов на экране;
- ModelView – получает информацию о взаимодействии пользователя с представлением;
- Model – хранит данные, полученные с базы данных.

Данный функционал представлен на диаграмме, изображенной на рисунке 7.

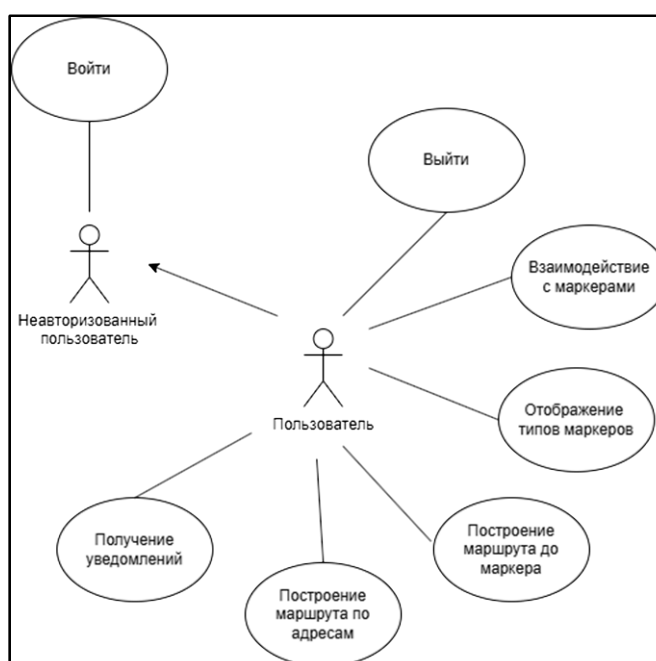


Рисунок 7 – UML диаграмма последовательностей

Более точное взаимодействие пользователя отражено на диаграммах деятельности.

Для авторизации пользователь вводит данные, которые программа отправляет на сервер. Если авторизация успешна, то пользователь переходит к основному фрагменту, иначе программа уведомляет о некорректных данных. Изображено на рисунке 8.

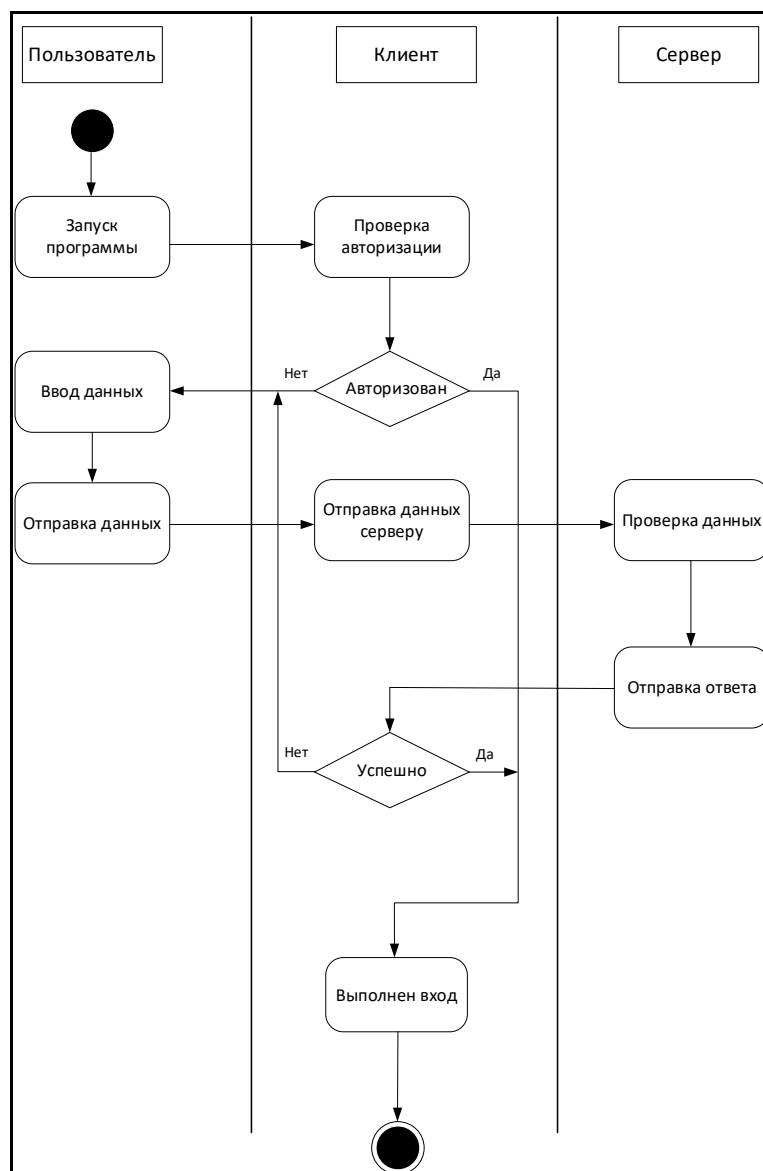


Рисунок 8 – Диаграмма деятельности авторизации

Для построения маршрута пользователь может выбрать один из вариантов: создание маршрута до конкретного маркера на карте, построение по адресам. Чтобы построить маршрут приложение обращается к сервисам Яндекс, посылая конечный координаты, тип транспорта, количество точек. В качестве ответа, программа получает маршрут в виде объекта Polyline, который передается карте для последующей отрисовки. Изображено на рисунке 9.

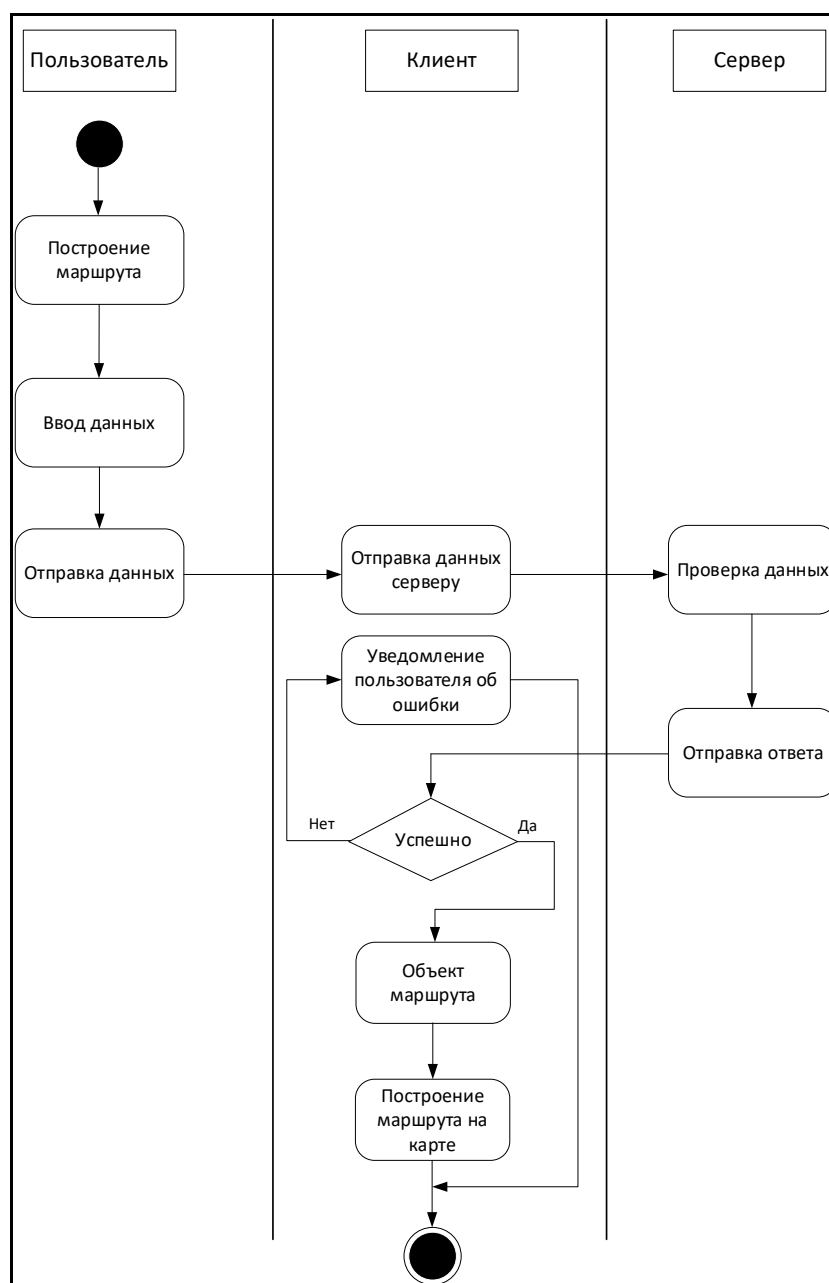


Рисунок 9 – Диаграмма деятельности маршрутизации

Все выполняемые действия как пользователя, так и клиента с сервером изображены на диаграмме последовательности, изображенной на рисунке 10.



Рисунок 10 - Диагармма последовательности

Физическое представление компонентов показана на диаграмме развертывания, изображенной на рисунке 11.

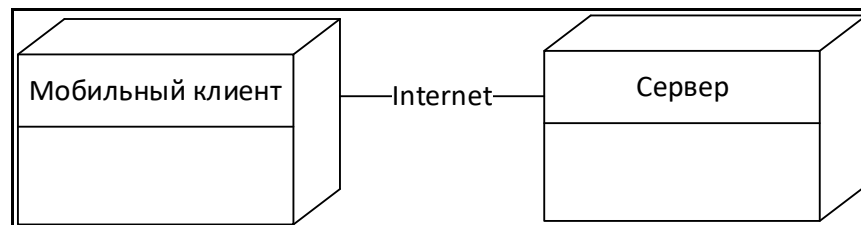


Рисунок 11 – Диаграмма развертывания

Далее будет рассмотрено назначение каждого из модулей.

4.1 Представление

Представление реализовано с помощью xml-разметки. Состоит из главного слоя `FrameLayout`.

```

<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/contentFrame"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:keepScreenOn="true"
tools:context="ru.agattrans.ui.MainActivity"/>
  
```

Далее, идет разбиение на фрагменты: регистрация и отображение карты. Основной фрагмент включает в себя:

– Представление карты:

```

<com.yandex.mapkit.mapview.MapView
android:id="@+id/mapView"
android:layout_width="match_parent"
android:layout_height="match_parent" />
  
```

– Панель с кнопками для поиска адресов и прокладывания маршрута, а также выбора отображаемых объектов: АЗС и адреса загрузки/выгрузки клиентов компании:

```

<androidx.appcompat.widget.Toolbar
android:id="@+id/toolbar"
style="@style/Widget.Toolbar"
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:title="@string/app_name"/>
  
```

– Кнопка определения местоположения:

```

        <com.google.android.material.floatingactionbutton.FloatingAction
nButton
        android:id="@+id/myLocationButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_marginBottom="16dp"
        android:layout_marginEnd="16dp"
        android:src="@drawable/ic_fab_my_location"
        app:backgroundTint="#FFFFFF"
        app:fabSize="normal"/>

```

Фрагмент регистрации состоит из:

– Полей ввода логина и пароля:

```

<com.google.android.material.textfield.TextInputLayout
        android:id="@+id/carNumberLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp">

<com.google.android.material.textfield.TextInputEditText
        android:id="@+id/carNumberEdit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/car_number"
        android:inputType="textVisiblePassword"/>

</com.google.android.material.textfield.TextInputLayout>

```

– Кнопки авторизации пользователя в системе:

```

<Button
        android:id="@+id/signInButton"
        style="@style/Widget.AppCompat.Button.Colored"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="@string/sign_in"/>

```

4.2 Модель представления

Модель представления получает данные о действиях, совершенных пользователем. В приложении есть несколько моделей представления: для фрагмента авторизации и для основного фрагмента.

4.2.1 Модель представления основного фрагмента

Информирование о нажатии на отображенный объект карты происходит с помощью вызова события на стороне представления, отображая данные о объекте в сплывающем окне:

```
override fun onMapObjectTap(marker: MapObject, point: Point): Boolean {
    showStationInfo((marker as PlacemarkMapObject).userData as Station)
    return true
}
```

Приложение поддерживает несколько вариантов построения маршрута: построение маршрута относительно пользователя и построение маршрутов через адреса.

Для построения маршрута относительно местоположения пользователя происходит поэтапно.

Сначала, приложение получает данные о местоположении пользователя путем запроса к серверу через менеджера геолокации.

```
locationManager.requestSingleUpdate(this)
```

Затем, приходит ответ в виде объекта с данными о геолокации, после чего создается экземпляр класса, содержащий информацию о маршруте:

– массив точек

```
val requestPoints = listOf(
    RequestPoint(pointFrom, RequestPointType.WAYPOINT, null, null, null),
    RequestPoint(pointTo, RequestPointType.WAYPOINT, null, null, null)
)
```

– количество точек

```
val drivingOptions = DrivingOptions().setRoutesCount(3)
```

– тип транспорта

```
VehicleOptions().setVehicleType(VehicleType.TRUCK)
```

Далее, совершается запрос к серверу.

```
drivingSession = drivingRouter.requestRoutes(
    requestPoints,
    drivingOptions,
```

```

        VehicleOptions().setVehicleType(VehicleType.TRUCK),
        this
    )

```

Полученные данные о маршруте отображаются на карте.

```

val map = fragmentMapBinding?.mapView?.map
polylineMap?.let { map?.mapObjects?.remove(it) }
polylineMap =
map?.mapObjects?.addPolyline(routes.last().geometry)

```

Получение адресов осуществляется в окне с полями ввода начала и конца маршрута, где с помощью технологии «Геосаджест» пользователя получает список подсказок адресов, построенный на основе введенных данных, откуда выбирает искомый адрес.

```

override fun onResponse(suggest: SuggestResponse) {
    suggestResult.clear()
    suggestItems.clear()
    run {
        suggest.items.forEachIndexed { index, item ->
            if (index == RESULT_NUMBER_LIMIT) return@run
            suggestResult.add(item.displayText!!)
            suggestItems.add(item)
        }
    }
    suggestResultAdapter.notifyDataSetChanged()
    suggestResultAdapter.setNotifyOnChange(true)
}

```

Затем, после нажатия на кнопку «Маршрут», по вышеописанному алгоритму, происходит построение маршрута.

Для отображения объектов на карте создается массив маркеров, которые содержат данные, полученные из базы.

```

map?.mapObjects?.addPlacemark()?.apply {
    this.geometry = Point(it.latitude, it.longitude)
    this.userData = it

    val objectType = objectTypes.find { objectType ->
        it.objectTypeId == objectType.id }
    val decodedString = Base64.decode(objectType?.image,
        Base64.DEFAULT)
    if (objectType != null && objectType.image.isNotEmpty()) {
        val bitmap =
        BitmapFactory.decodeByteArray(decodedString, 0,
            decodedString.size)
        this.setIcon(ImageProvider.fromBitmap(bitmap))
    } else {

```

```

        when (it.priority) {
            1 ->
            setIcon(ImageProvider.fromResource(mainActivity,
            R.drawable.mark_green))
            2 ->
            setIcon(ImageProvider.fromResource(mainActivity,
            R.drawable.mark_yellow))
            else ->
            setIcon(ImageProvider.fromResource(mainActivity,
            R.drawable.mark_red))
        }
    }

    this.addTapListener(this@MapScreenFragment)
}?.let { it1 -> placemarks?.add(it1) }

```

Если на карте, ранее, были построены объекты, то, перед добавлением новых, массив объектов карты очищается.

```
map?.mapObjects?.clear()
```

Определение местоположения осуществляется с помощью запросу к серверу, на основе полученных координат, создается маркер и меняется позиционирование камеры.

```

private fun animateToUserLocation() {
    locationManager.requestSingleUpdate(this)
    userLocationMarker?.let {
        fragmentMapBinding?.mapView?.map?.move(
            CameraPosition(it, 14.0f, 0.0f, 0.0f),
            Animation(SMOOTH, 1f),
            null
        )
        followUserLocation = true
    }
}

```

4.2.2 Модель представления авторизации

Для авторизации пользователь вводит государственный номер транспортного средства и пароль.

Затем, после нажатия на кнопку в «Войти», вызывается метод авторизации, где из представлений для ввода берутся данные профиля.

```

val carNumber =
fragmentLoginBinding?.carNumberEdit?.text.toString()
val password =
fragmentLoginBinding?.passwordEdit?.text.toString()

```

Далее, делается запрос на сервер, при успешной авторизации, открывается основной фрагмент, иначе пользователя информируют об ошибке авторизации.

```
progressDialog.setMessage("Авторизация...")
subscriptions.add(
    api.login(carNumber, password)
        .observeOn(AndroidSchedulers.mainThread())
        .doAfterTerminate { progressDialog.dismiss() }
        .subscribe(
            { (token) ->
                Timber.i("User was successfully sign in")
                tokenPref.set(token)
                openMapScreen()
            },
            { error ->
                Timber.w(error, "Failed to sign in")
                showAuthorizationError()
            }
        )
    )
)
```

4.3 Модель

Модели представляют несколько классов:

- Stations – хранит в себе информацию о местоположении объекта: адрес и координаты, приоритет и комментарий к объекту;
- ObjectTypes – хранит данные о типе представления объекта пользователю;
- Notifications – хранит сообщение пользователю от диспетчера и время отправки сообщения.

Получение данных происходит с помощью библиотеки Room следующим образом, есть три основных компонента:

Класс базы данных, который содержит базу данных и служит основной точкой доступа для базового подключения к постоянным данным вашего приложения.

Объекты данных, представляющие таблицы в базе данных вашего приложения.

Объекты доступа к данным (DAO), которые предоставляют методы, которые ваше приложение может использовать для запроса, обновления, вставки и удаления данных в базе данных.

Извлечение данных из базы происходит с помощью объектов DAO, которые предоставляются классом базы данных. Полученные данные являются экземплярами связанных объектов данных [4]. Взаимосвязь между различными компонентами Room отражена на рисунке 12.

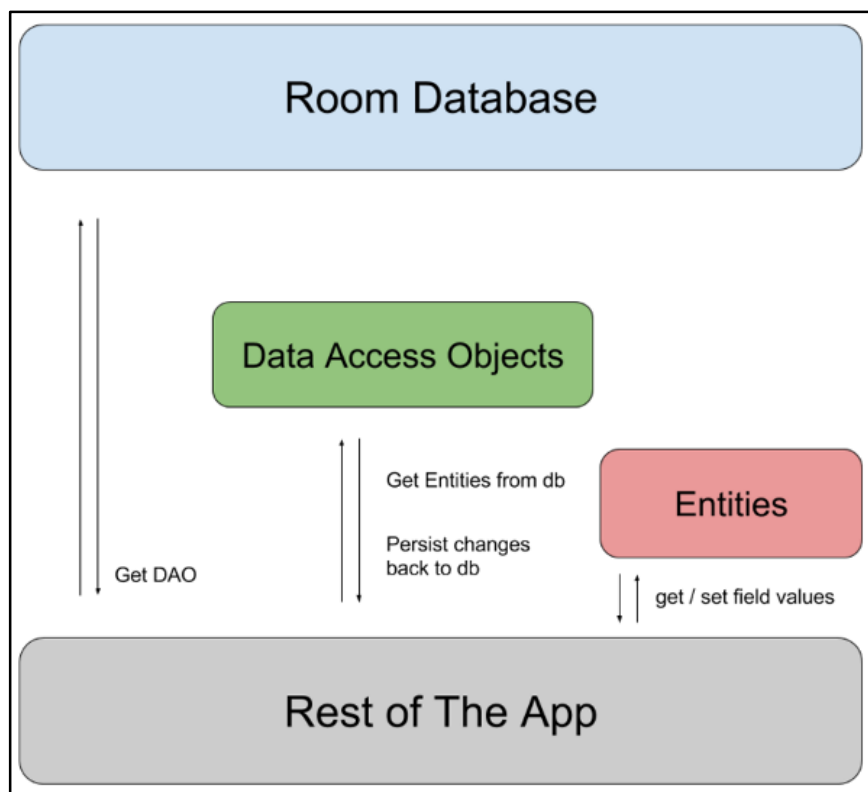


Рисунок 12 – Взаимосвязь между различными компонентами Room

5 Тестирование

В этом разделе будет проведено функциональное тестирование приложения.

5.1 Тестирование авторизации

Для авторизации пользователь должен заполнить следующие поля: ввести государственный номер и пароль, как показано на рисунке 13.

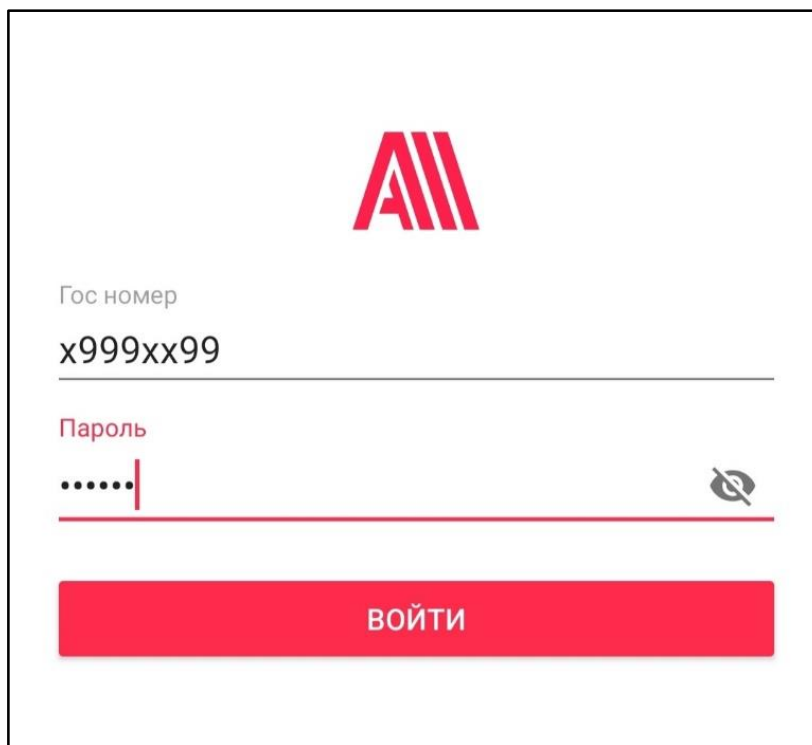


Рисунок 13 – Поля ввода данных авторизации

Затем, после нажатия на кнопку «Войти», если данные корректны откроется основное окно приложения, изображенно на рисунке 14.

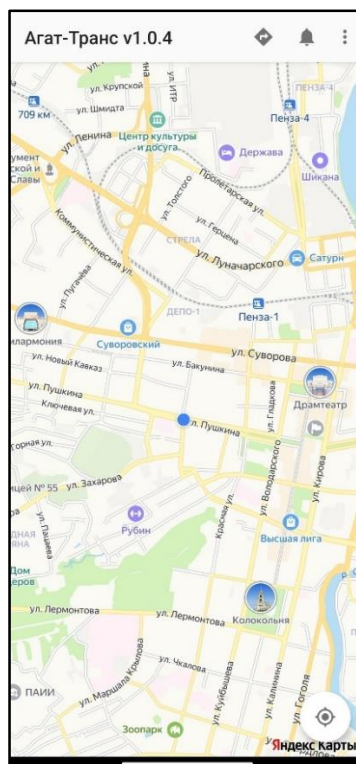


Рисунок 14 – Ошибка авторизации

Иначе, придет уведомление о неправильных данных, изображенное на рисунке 15.

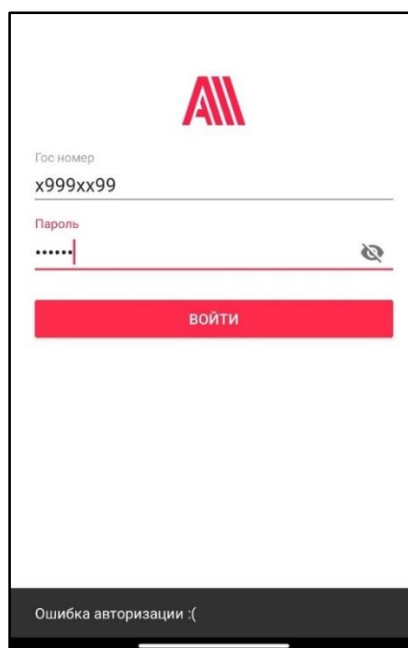


Рисунок 15 – Ошибка авторизации

					<i>ПГУ1.090301.47.001 ПЗ</i>	Лист
						33
Изм.	Лист	№ докум.	Подпись	Дата		

5.2 Построение маршрута до станции

Для построения маршрута до станции надо нажать на маркер, изображенный на карте, на рисунке 16. Появится всплывающее окно, в котором надо нажать кнопку «Маршрут».

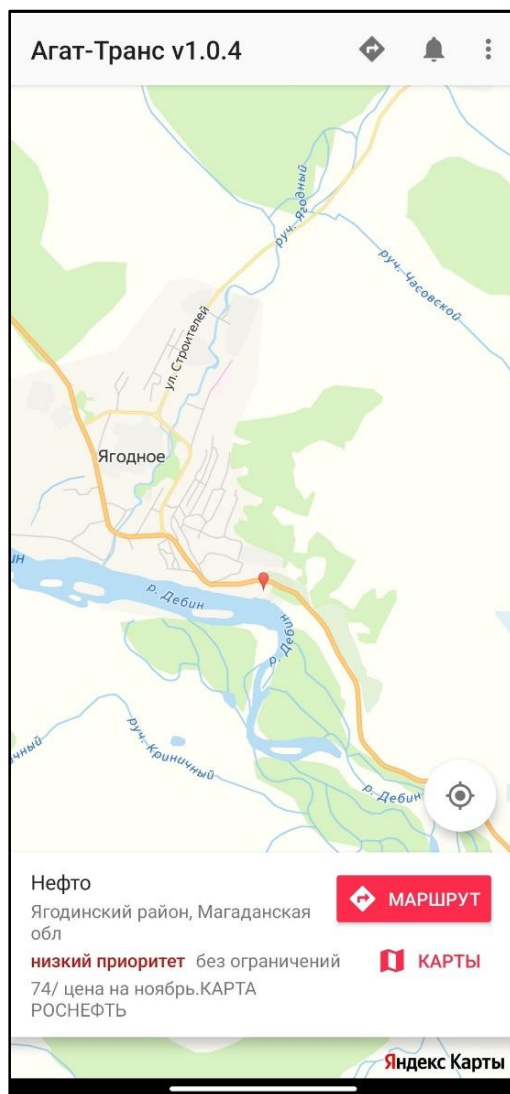


Рисунок 16 – Взаимодействие с маркером на карте

После этого появится окно с текстом: «Построение маршрута...», изображенное на рисунке 17.

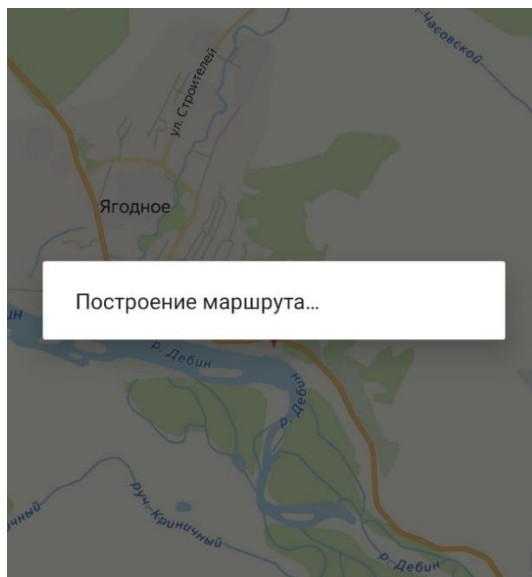


Рисунок 17 – Уведомление о построении маршрута

Затем, камера поменяет свое местоположение относительно построенного маршрута, изображенно на рисунке 18.



Рисунок 18 – Отображение маршрута на карте

5.3 Построение и отмена маршрута по адресам

Для построения маршрута по адресам требуется открыть представление, в котором будут, поля ввода конечных точек и кнопка для построения маршрута, изображенные на рисунке 19.

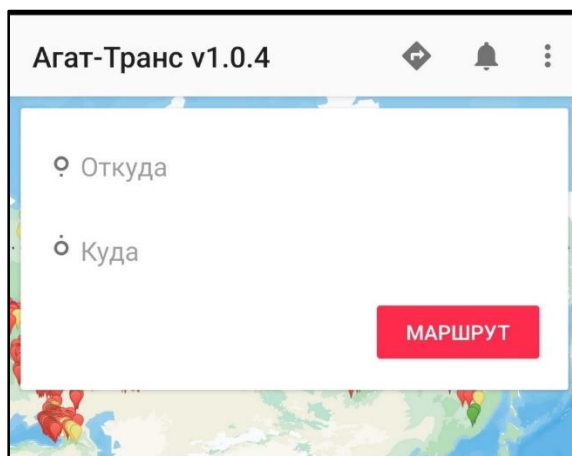


Рисунок 19 – Окно маршрутизации по адресам

Во время заполнения полей, пользователю будет предоставлен список адресов, который создается на основе введенных им данных, изображенно на рисунке 20.

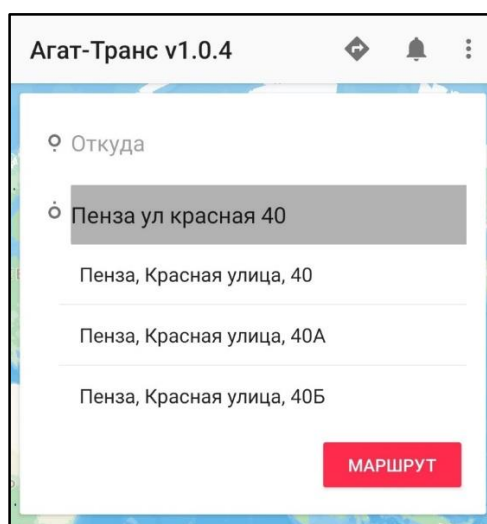


Рисунок 20 – Список адресов

Далее, пользователь нажимает на один из предложенных вариантов, который заносится в строку ввода, и приложение получает данные от сервисов Яндекс, о местоположении искомого адреса, изображенно на рисунке 21.

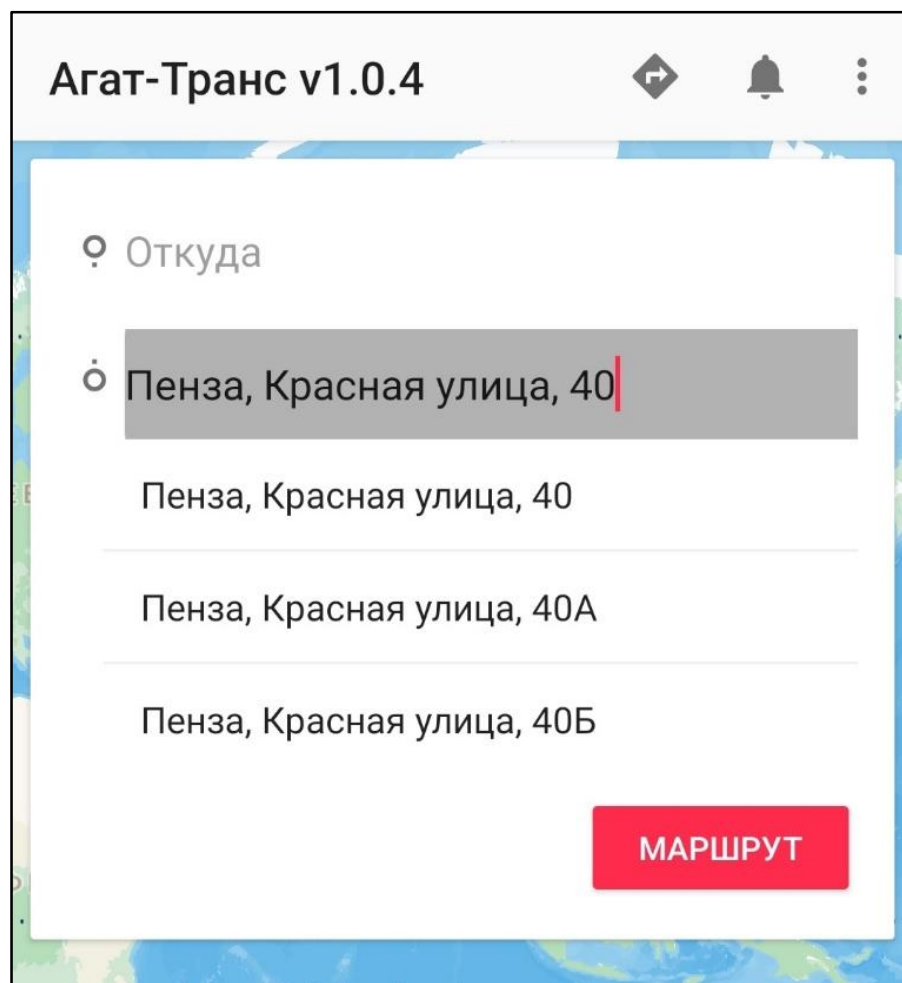


Рисунок 21 – Получение данных искомого адреса

Есть несколько вариантов маршрутизации по адресам. Первый, основывается на наличие данных о конечном адресе, началом маршрута будет считаться местоположение пользователя, изображенно на рисунке 21. После нажатия на кнопку «Маршрут» будет построен маршрут, изображенно на рисунке 22.

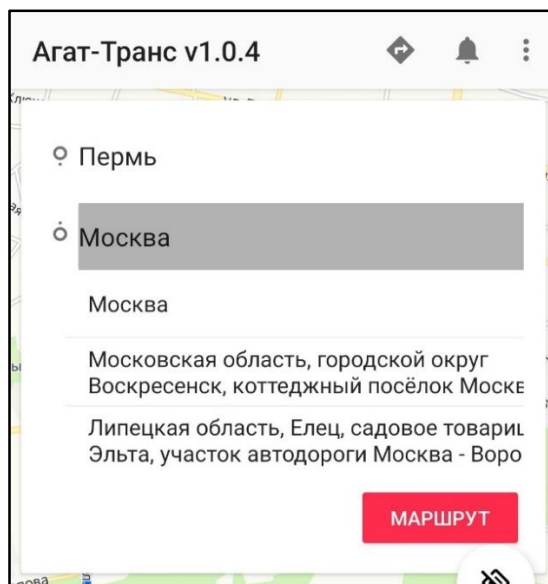


Рисунок 23 – Ввод двух адресов

После нажатия на кнопку «Маршрут» будет построен маршрут от одного адреса к другому, изображенно на рисунке 24.

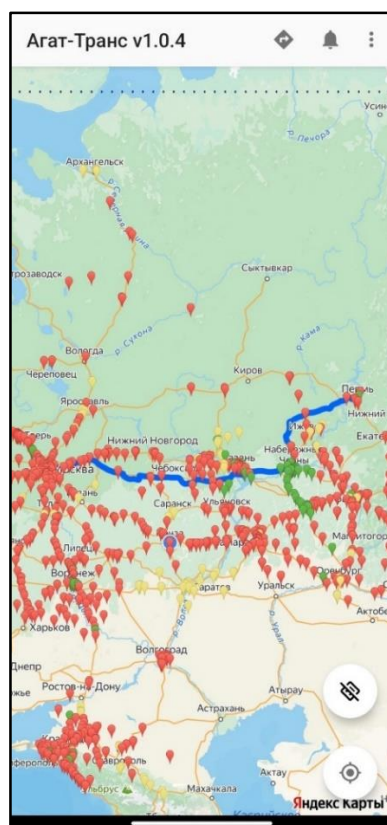


Рисунок 24 – Отображение маршрутизации по двум адресам

Отмена маршрута осуществляется с помощью кнопки в правом нижнем углу экрана, которая появляется после построения маршрута, изображенно на рисунке 25.

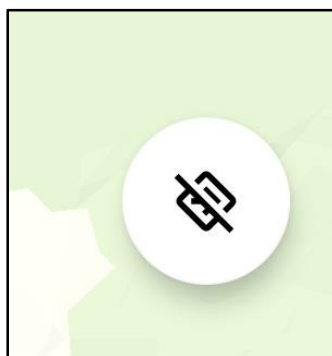


Рисунок 25 – Кнопка отмены построения маршрута

5.4 Маршрутизация через другие сервисы

Взаимодействие со станциями через другие картографические сервисы происходит через нажатия на кнопку «Карта» в сплывающем окне, после нажатия на маркер, изображенный на карте, изображенно на рисунке 26.

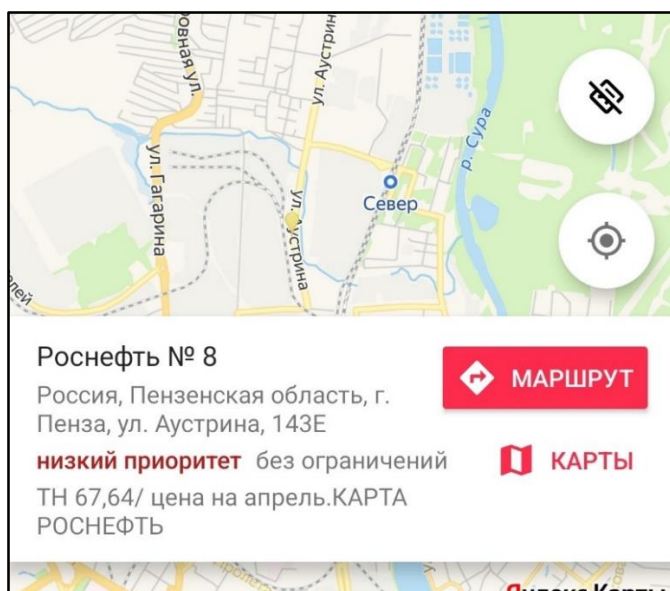


Рисунок 26 – Взаимодействие с маркером на карте

После, открывается окно с вариантами картографических сервисов, изображенно на рисунке 27.

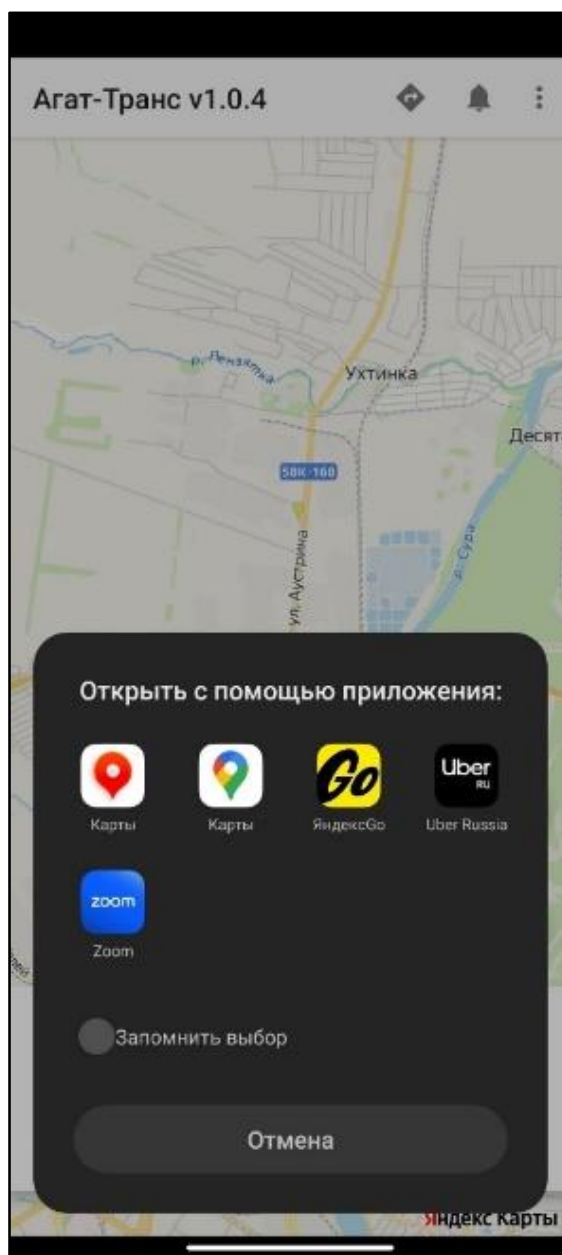


Рисунок 27 – Варианты картографических сервисов

После выбора одного из них приложение передает координаты метки выбранному сервису, изображенно на рисунке 28.

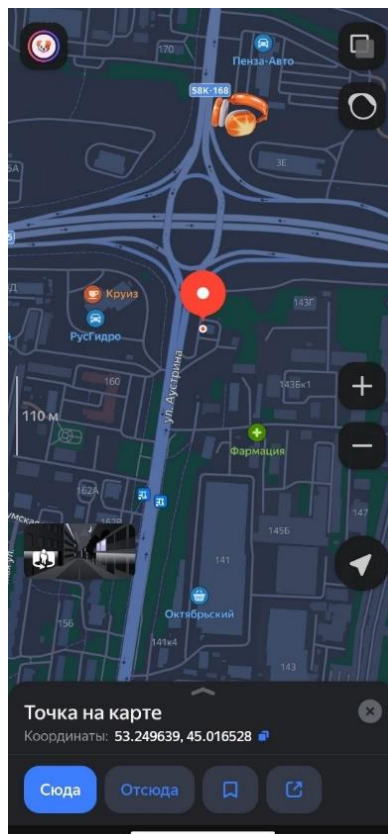


Рисунок 28 – Сторонний картографический сервис

5.5 Выбор отображения различных типов маркеров

Для выбора отображения различных типов маркеров требуется в верхней панели нажать крайнюю левую кнопку с изображением троеточия, изображенно на рисунке 29.

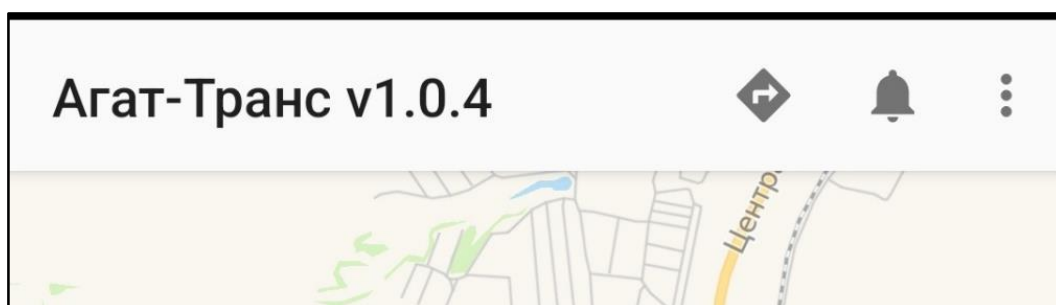


Рисунок 29 – Верхняя панель приложения

В открывшемся списке нажимаем на кнопку «Типы объектов», изображенно на рисунке 30.

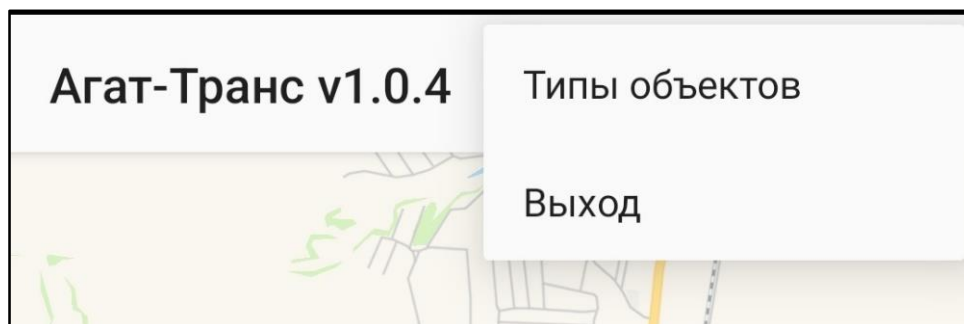


Рисунок 30 – Список кнопок

В открывшемся окне выбираем типы объектов, изображенно на рисунке 31.

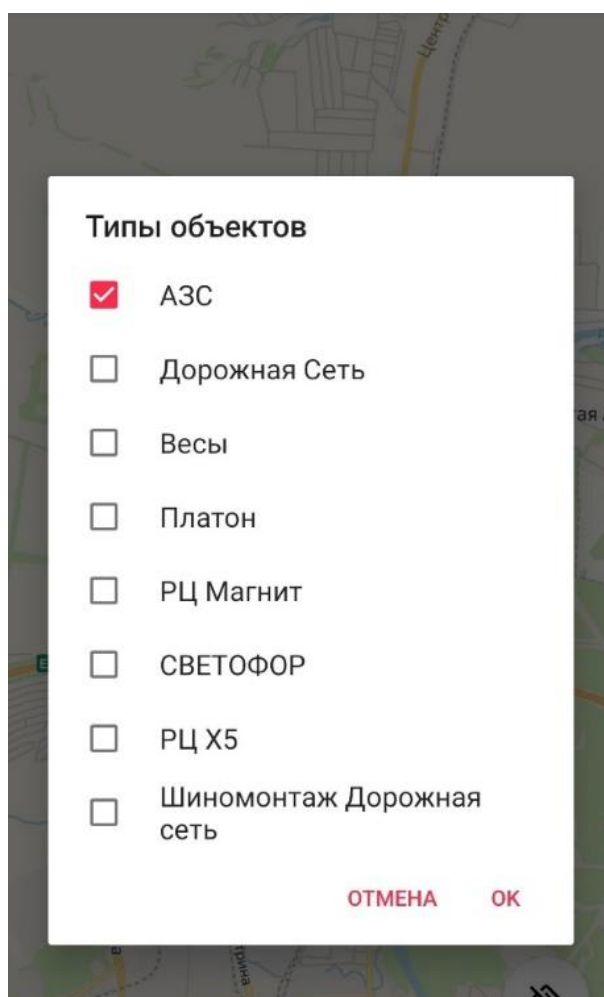


Рисунок 31 – Окно выбора отображения типов объектов

Далее, нажимаем на кнопку «Ок» для отображения объектов на карте, изображенно на рисунке 32.

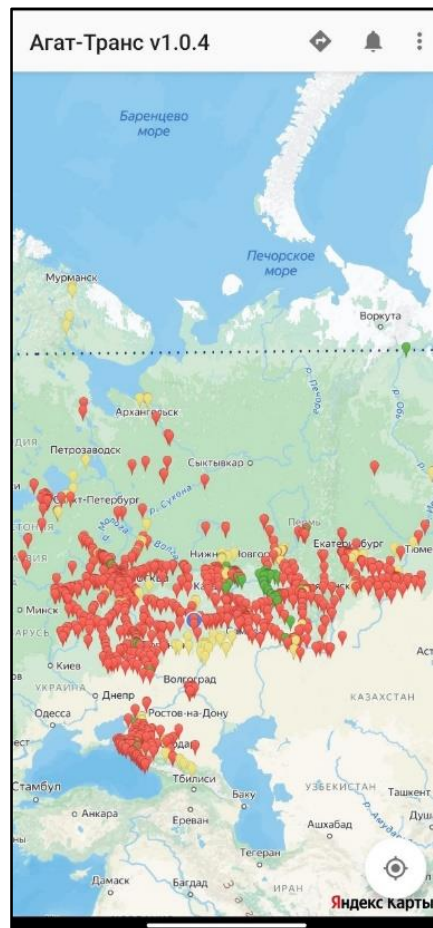


Рисунок 32 – Карта с выбранными типами объектов

6 Безопасность жизнедеятельности

6.1 Техника безопасности при работе с ПК

Согласно общим требованиям к организации рабочих мест пользователей ПЭВМ, изложенных в документе санитарно-эпидемиологических правил и нормативов [7], для организации рабочего места надо следовать следующим инструкциям, изображенно на рисунке 33:

- экран монитора должен находиться на расстоянии 600-700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов;
- конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования, поверхность стола должна иметь коэффициент отражения 0,5-0,7;
- конструкция рабочего стула должна обеспечивать поддержание рациональной рабочей позы при работе на ПЭВМ, позволять изменять позу с целью снижения статического напряжения мышц.

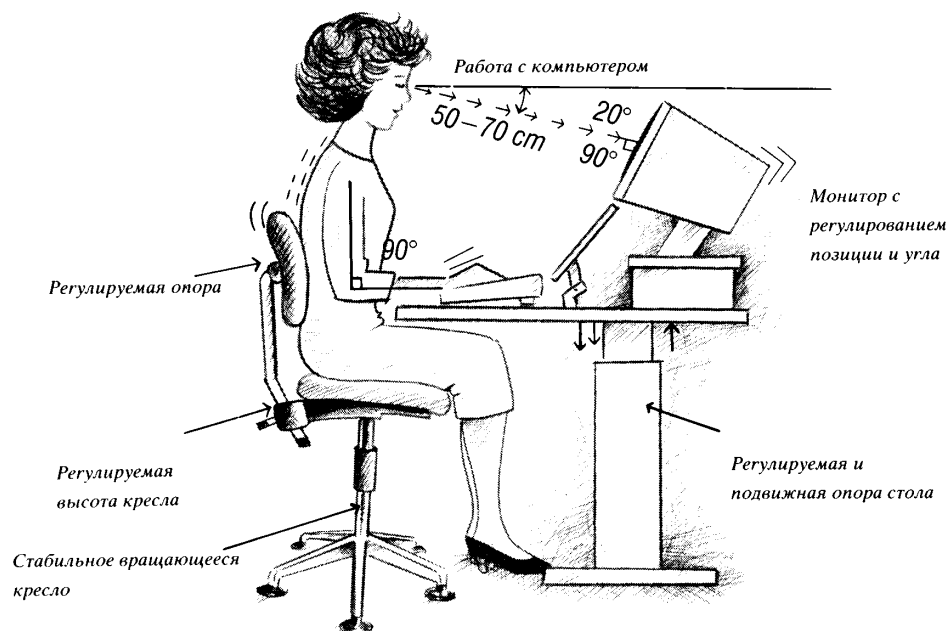


Рисунок 33 – Инструкция к рабочему месту

6.2 Техника безопасности при работе с мобильными устройствами

Согласно предписанию центра гигиены и эпидемиологии Пензенской области, был составлен список правил по использованию мобильных устройств [8]:

- держите телефон на уровне глаз;
- делайте перерывы по правилу 20-20-20: каждые 20 минут смотрите 20 секунд на объект в 20 метрах;
- используйте режим «темной темы» и фильтр синего света.

					<i>ПГУ1.090301.47.001 ПЗ</i>	Лист
						46
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

Результатом выполнения данной выпускной квалификационной работы можно считать успешно разработанное мобильное приложение «Картографический сервис АЗС», полностью соответствующее заданию. Также были выполнены следующие задачи:

- полный анализ темы и аналогов;
- выбор инструментов разработки;
- анализ архитектуры разрабатываемого приложения;
- непосредственная разработка приложения;
- тестирование ПО.

В результате было создано полностью протестированное приложение для мобильных устройств написанное на языке программирования Kotlin. Также в ходе разработки были получены навыки работы с Yandex Mapkit. Таким образом результаты показали, что приложение полностью соответствует всем поставленным задачам, имеет удобный пользовательский интерфейс и выполняет свою работу корректно.

В перспективе приложение можно будет улучшить, добавив новый функционал и доработав пользовательский интерфейс для более комфортного использования со стороны пользователей. Данное мобильное приложение готово к использованию в бизнесе, связанном с транспортной логистикой, и поможет улучшить поиск АЗС.

Список использованных источников

1. gs.statcounter.com: [Электронный ресурс].
URL: <https://gs.statcounter.com/os-market-share/mobile/worldwide> (дата обращения: 02.05.2025)
2. tiobe.com: [Электронный ресурс]. URL: <https://www.tiobe.com/tiobe-index/>
(дата обращения: 02.05.2025)
3. ru.wikipedia.org: [Электронный ресурс].
URL: <https://ru.wikipedia.org/wiki/Kotlin> (дата обращения: 0.05.2025)
4. android-developers.googleblog.com: [Электронный ресурс].
URL: <https://android-developers.googleblog.com/2017/05/android-announces-support-for-kotlin.html> (дата обращения: 04.05.2025)
5. developer.android.com: [Электронный ресурс].
URL: <https://developer.android.com/training/data-storage/room?hl=ru> (дата обращения: 04.05.2025)
6. en.wikipedia.org: [Электронный ресурс].
URL: <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel>
(дата обращения: 05.05.2025)
7. Санитарно-эпидемиологические правила и нормативы СанПиН 2.2.2/2.4.1340-03.
8. cge58.ru [Электронный ресурс]. URL: <https://www.cge58.ru/news/07-05-2025/gigiena-ispolzovaniya-mobilnyh-telefonov-kak-snizit-riski-dlya-zdorovya> (дата обращения: 06.05.2025)

Приложение А

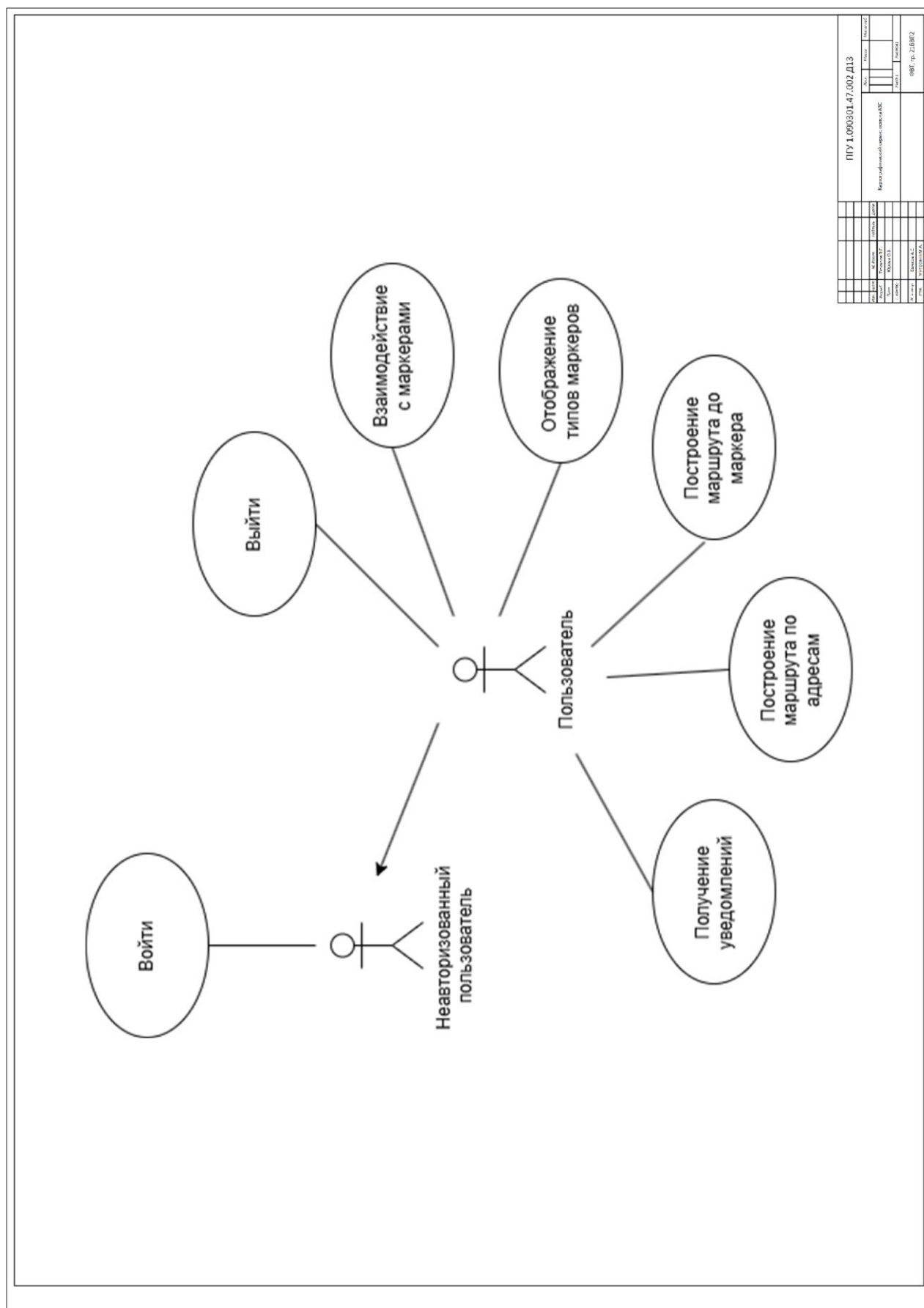
(Обязательное)

Картографический сервис поиска АЗС

UML– диаграммы

					ПГУ1.090301.47.001 ПЗ	Лист
						49
Изм.	Лист	№ докум.	Подпись	Дата		

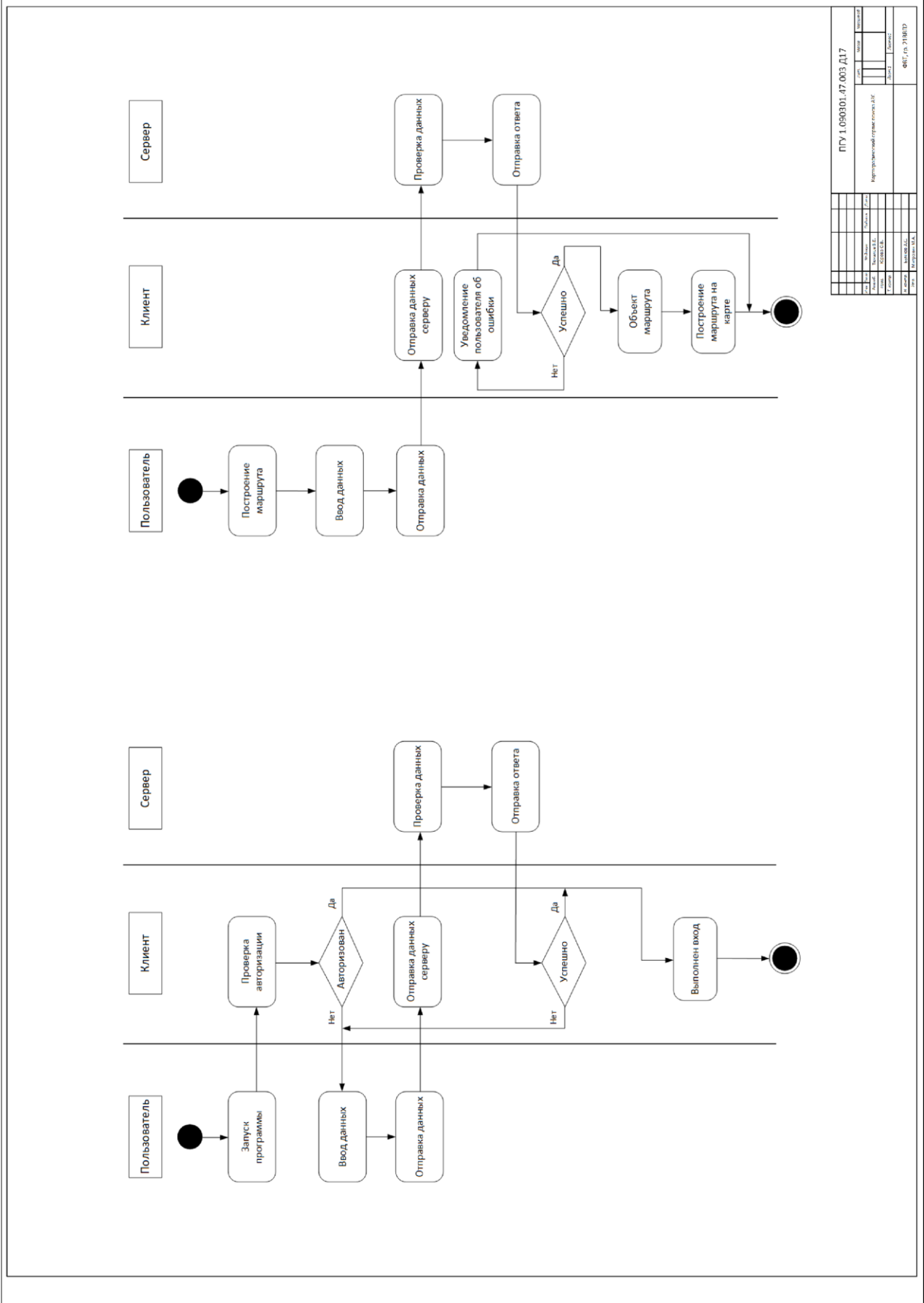
Приложение А.1-Диаграмма прецедентов



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.47.001 ПЗ

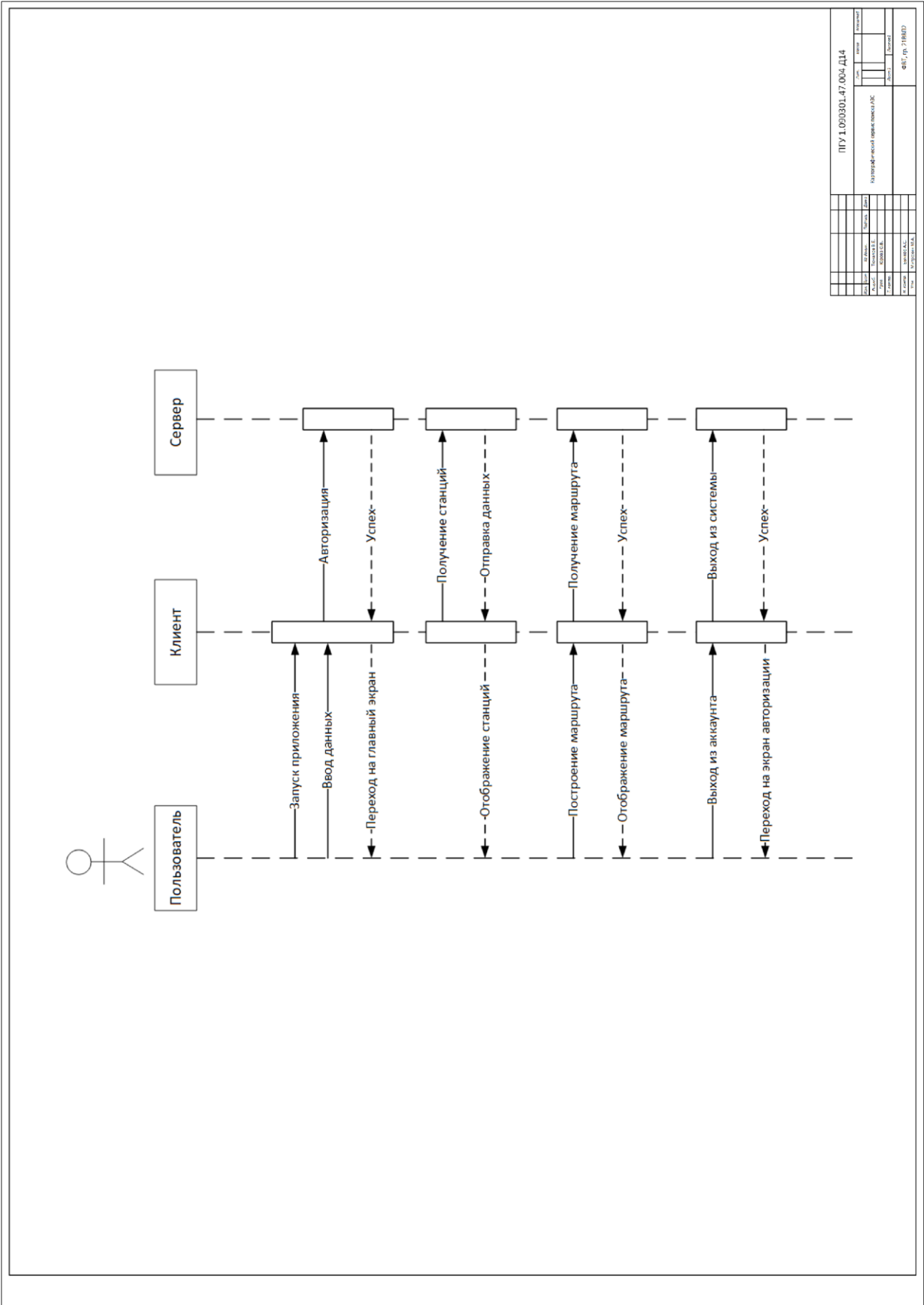
Приложение А.2-Диаграмма деятельности



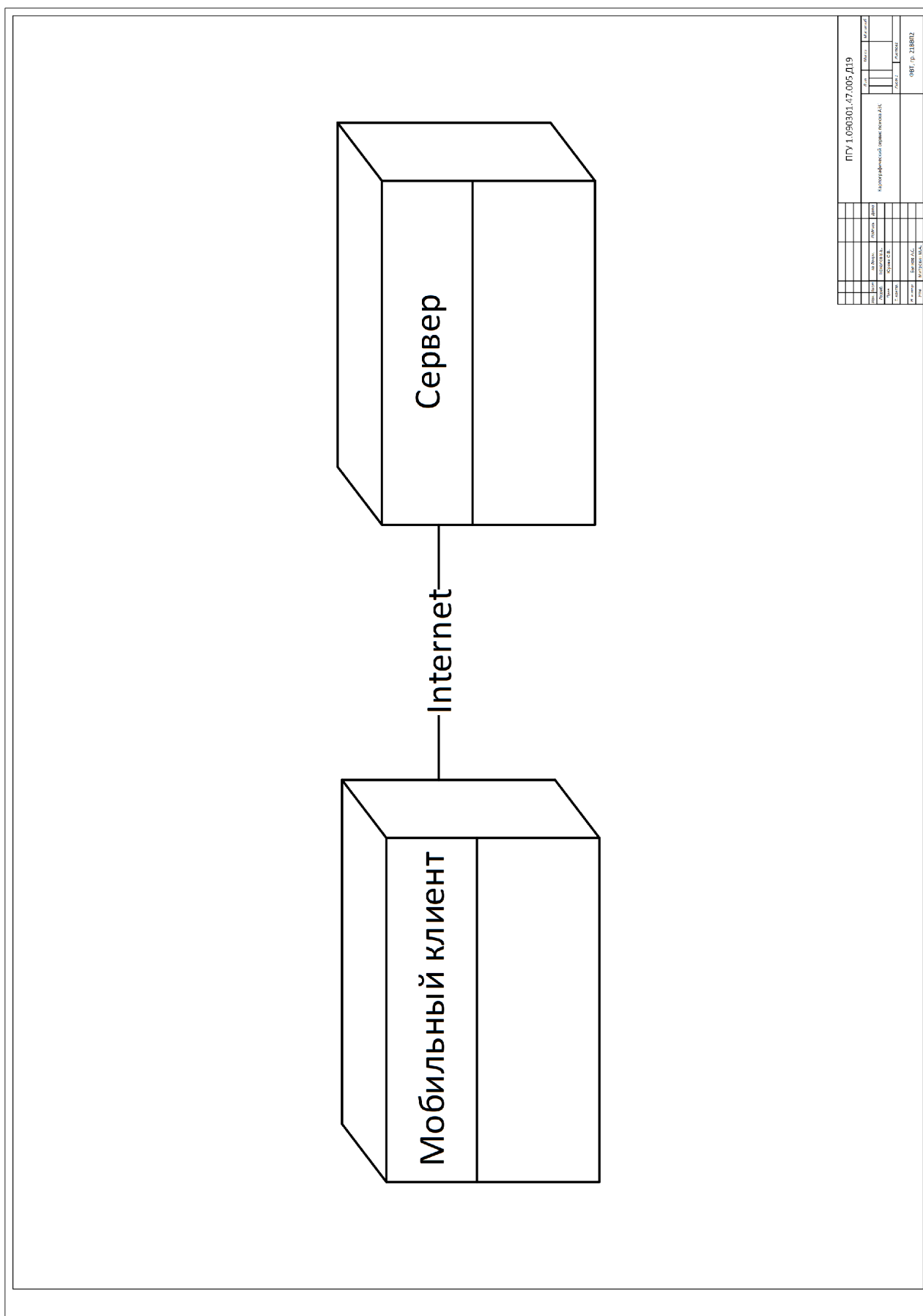
ПУ 1.090301.47.003 Д17									
Наименование государственного АИ									
И.О.	Ф.И.О.	Подпись	Дата	И.О.	Ф.И.О.	Подпись	Дата	И.О.	Ф.И.О.
ФГИС, п. 318037									

Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.3-Диаграмма последовательности



Приложение А.3-Диаграмма развертывания



Приложение Б

(Обязательно)

Картографический сервис поиска АЗС

Листинг программы

					ПГУ1.090301.47.001 ПЗ	Лист
						54
Изм.	Лист	№ докум.	Подпись	Дата		

Файл «App.kt»

```
package ru.agattrans

import android.content.Context
import android.os.StrictMode
import androidx.multidex.BuildConfig
import androidx.multidex.MultiDex
import com.crashlytics.android.Crashlytics
import com.facebook.stetho.Stetho
import com.jakewharton.threetenabp.AndroidThreeTen
import com.yandex.mapkit.MapKitFactory
import dagger.android.AndroidInjector
import dagger.android.support.DaggerApplication
import io.fabric.sdk.android.Fabric
import io.reactivex.plugins.RxJavaPlugins
import ru.agattrans.util.CrashlyticsTree
import timber.log.Timber

class App : DaggerApplication() {

    override fun applicationInjector(): AndroidInjector<App> =
        DaggerAppComponent.builder().create(this)

    override fun attachBaseContext(base: Context?) {
        super.attachBaseContext(base)
        MultiDex.install(this)
    }

    override fun onCreate() {
        super.onCreate()

        initRxJava()
        initThreeTen()
        initTimber()
        if (BuildConfig.DEBUG) {
            enableStrictMode()
            enableStetho()
        } else {
            enableCrashlytics()
        }

        MapKitFactory.setApiKey(ru.agattrans.BuildConfig.GEO_API_KEY)
    }

    private fun initRxJava() {
        RxJavaPlugins.setErrorHandler {}
    }

    private fun initThreeTen() {
        AndroidThreeTen.init(this)
    }
}
```

					ПГУ1.090301.47.001 ПЗ	Лист
						55
Изм.	Лист	№ докум.	Подпись	Дата		

```

private fun initTimber() {
    Timber.plant(
        if (BuildConfig.DEBUG) Timber.DebugTree()
        else CrashlyticsTree()
    )
}

private fun enableCrashlytics() {
    Fabric.with(this, Crashlytics())
}

private fun enableStrictMode() {
    StrictMode.setThreadPolicy(StrictMode.ThreadPolicy.Builder()
        .detectAll()
        .penaltyLog()
        .build())

    StrictMode.setVmPolicy(StrictMode.VmPolicy.Builder()
        .detectAll()
        .penaltyLog()
        .build())
}

private fun enableStetho() {
    Stetho.initializeWithDefaults(this)
}
}

```

Файл «AppModule.kt»

```

package ru.agattrans

import android.app.Application
import dagger.Binds
import dagger.Module

@Module
abstract class AppModule {

    @Binds
    abstract fun application(app: App): Application
}

```

Файл «AppComponent.kt»

```

package ru.agattrans

import dagger.Component
import dagger.android.AndroidInjector
import dagger.android.support.AndroidSupportInjectionModule
import ru.agattrans.data.api.ApiModule
import ru.agattrans.data.db.DatabaseModule
import ru.agattrans.data.preference.PreferencesModule

```

```

import ru.agattrans.location.LocationModule
import ru.agattrans.ui.UiModule
import javax.inject.Singleton

@Singleton
@Component(
    modules = [
        AndroidSupportInjectionModule::class,
        AppModule::class,
        ApiModule::class,
        DatabaseModule::class,
        LocationModule::class,
        PreferencesModule::class,
        UiModule::class
    ]
)
interface AppComponent : AndroidInjector<App> {

    @Component.Builder
    abstract class Builder : AndroidInjector.Builder<App>()
}

```

Файл «LoginFragment.kt»

```

package ru.agattrans.ui.login

import android.annotation.SuppressLint
import android.app.AlertDialog
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import com.f2prateek.rx.preferences2.Preference
import com.google.android.material.snackbar.Snackbar
import com.jakewharton.rxbinding2.widget.textChanges
import dagger.android.support.DaggerFragment
import io.reactivex.android.schedulers.AndroidSchedulers
import io.reactivex.disposables.CompositeDisposable
import io.reactivex.rxkotlin.Observables
import ru.agattrans.R
import ru.agattrans.data.api.AgatTransApi
import ru.agattrans.data.preference.TokenPref
import ru.agattrans.databinding.FragmentLoginBinding
import ru.agattrans.ui.map.MapScreenFragment
import ru.agattrans.util.applyInputMask
import ru.agattrans.util.hideKeyboard
import ru.agattrans.util.mainActivity
import ru.agattrans.util.singleClicks
import timber.log.Timber
import javax.inject.Inject

class LoginFragment : DaggerFragment() {
    @Inject lateinit var api: AgatTransApi
}

```

					ПГУ1.090301.47.001 ПЗ	Лист
						57
Изм.	Лист	№ докум.	Подпись	Дата		

```

@Inject @field:TokenPref lateinit var tokenPref:
Preference<String>

private val subscriptions = CompositeDisposable()

private var fragmentLoginBinding: FragmentLoginBinding? =
null

override fun onCreateView(inflater: LayoutInflater,
container: ViewGroup?, savedInstanceState: Bundle?): View {
    val binding = FragmentLoginBinding.inflate(inflater,
container, false)
    fragmentLoginBinding = binding
    return binding.root
}

override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
    super.onViewCreated(view, savedInstanceState)

fragmentLoginBinding?.carNumberEdit?.applyInputMask(CAR_NUMBER_I
NPUT_MASK)
    observeSignInButtonClicks()
    observeSignInButtonState()
}

@SuppressLint("RxSubscribeOnError")
private fun observeSignInButtonClicks() {
    subscriptions.add(

fragmentLoginBinding?.signInButton?.singleClicks()!!.subscribe {
    signIn()
}

)

}

@SuppressLint("RxSubscribeOnError")
private fun observeSignInButtonState() {
    subscriptions.add(

Observables.combineLatest(fragmentLoginBinding?.carNumberEdit?.t
extChanges()!!,

fragmentLoginBinding?.passwordEdit?.textChanges()!!) {
carNumber, password ->
    carNumber.length >= 8 && password.isNotBlank()
    }.subscribe { signInButtonEnabled ->
        fragmentLoginBinding?.signInButton?.isEnabled =
signInButtonEnabled
    }

)

}

```

```

        private fun signIn() {
            val carNumber =
fragmentLoginBinding?.carNumberEdit?.text.toString()
            val password =
fragmentLoginBinding?.passwordEdit?.text.toString()
            val progressDialog =
AlertDialog.Builder(this.activity).create()
            progressDialog.setMessage("Авторизация...")
            subscriptions.add(
                api.login(carNumber, password)
                    .observeOn(AndroidSchedulers.mainThread())
                    .doAfterTerminate { progressDialog.dismiss() }
                    .subscribe(
                        { (token) ->
                            Timber.i("User was successfully sign in")
                            tokenPref.set(token)
                            openMapScreen()
                        },
                        { error ->
                            Timber.w(error, "Failed to sign in")
                            showAuthorizationError()
                        }
                    )
            )
        }

        private fun openMapScreen() {
            MainActivity?.openScreen(MapScreenFragment(), false)
        }

        private fun showAuthorizationError() {
            view?.let {
                Snackbar.make(it, R.string.error_authorization,
Snackbar.LENGTH_LONG).show()
            }
        }

        override fun onDestroyView() {
            subscriptions.clear()
            activity?.hideKeyboard()
            super.onDestroyView()
        }

        private companion object {
            private const val CAR_NUMBER_INPUT_MASK =
"[A][000][AA][009]"
        }
    }
}

```

Файл «DaggerMapFragment.kt»

					ПГУ1.090301.47.001 ПЗ	Лист
						59
Изм.	Лист	№ докум.	Подпись	Дата		

```

package ru.agattrans.ui.map

import android.os.Bundle
import android.view.LayoutInflater
import androidx.annotation.CallSuper
import android.view.View
import android.view.ViewGroup
import com.yandex.mapkit.MapKitFactory
import dagger.android.support.DaggerFragment
import ru.agattrans.databinding.FragmentMapBinding

open class DaggerMapFragment : DaggerFragment() {
    private var fragmentMapBinding: FragmentMapBinding? = null

    override fun onCreateView(inflater: LayoutInflater, container:
    ViewGroup?, savedInstanceState: Bundle?): View {
        val bindingMap = FragmentMapBinding.inflate(inflater,
        container, false)
        fragmentMapBinding = bindingMap
        return bindingMap.root
    }

    @CallSuper
    override fun onViewCreated(view: View, savedInstanceState:
    Bundle?) {
        super.onViewCreated(view, savedInstanceState)
    }

    override fun onStart() {
        super.onStart()
        MapKitFactory.getInstance().onStart()
        fragmentMapBinding?.mapView?.onStart()
    }

    override fun onStop() {
        super.onStop()
        MapKitFactory.getInstance().onStop()
        fragmentMapBinding?.mapView?.onStop()
    }
}

```

Файл «MapScreenFragment.kt»

```

package ru.agattrans.ui.map

import android.Manifest.permission.*
import android.annotation.SuppressLint
import android.app.Activity
import android.content.ActivityNotFoundException
import android.content.Intent
import android.graphics.BitmapFactory
import android.net.Uri
import android.os.Bundle

```

					ПГУ1.090301.47.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

```

import android.util.Base64
import android.util.Log
import android.view.*
import android.view.inputmethod.InputMethodManager
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import com.f2prateek.rx.preferences2.Preference
import com.google.android.gms.maps.model.LatLng
import com.google.android.material.snackbar.Snackbar
import com.tbruyelle.rxpermissions2.RxPermissions
import com.yandex.mapkit.*
import com.yandex.mapkit.Animation.Type.LINEAR
import com.yandex.mapkit.Animation.Type.SMOOTH
import com.yandex.mapkit.directions.DirectionsFactory
import com.yandex.mapkit.directions.driving.*
import com.yandex.mapkit.geometry.Geometry
import com.yandex.mapkit.geometry.Point
import com.yandex.mapkit.location.LocationListener
import com.yandex.mapkit.location.LocationManager
import com.yandex.mapkit.location.LocationStatus
import com.yandex.mapkit.map.*
import com.yandex.mapkit.map.Map
import com.yandex.runtime.Error
import com.yandex.runtime.image.ImageProvider
import io.reactivex.Completable
import io.reactivex.Single
import io.reactivex.android.schedulers.AndroidSchedulers
import io.reactivex.disposables.CompositeDisposable
import io.reactivex.rxkotlin.Observables
import io.reactivex.rxkotlin.Singles
import ru.agattrans.BuildConfig
import ru.agattrans.R
import ru.agattrans.data.DirectionsRepository
import ru.agattrans.data.NotificationsRepository
import ru.agattrans.data.ObjectTypesRepository
import ru.agattrans.data.StationsRepository
import ru.agattrans.data.api.AgatTransApi
import ru.agattrans.data.model.Notification
import ru.agattrans.data.model.ObjectType
import ru.agattrans.data.model.Station
import ru.agattrans.data.preference.EnabledObjectTypesPref
import ru.agattrans.data.preference.StationsTimestampPref
import ru.agattrans.data.preference.TokenPref
import ru.agattrans.databinding.FragmentMapBinding
import ru.agattrans.databinding.ViewDirectionsBinding
import ru.agattrans.databinding.ViewStationInfoBinding
import ru.agattrans.location.LocationProvider
import ru.agattrans.ui.login.LoginFragment
import ru.agattrans.ui.notification.NotificationsFragment

```

```

import ru.agattrans.ui.settings.ObjectTypesChooserDialog
import ru.agattrans.util.CustomSnackbar
import ru.agattrans.util.mainActivity
import ru.agattrans.util.visible
import timber.log.Timber
import javax.inject.Inject

class MapScreenFragment : DaggerMapFragment(),
    InputListener,
    LocationListener,
    MapObjectTapListener,
    DrivingSession.DrivingRouteListener {
    @Inject
    lateinit var api: AgatTransApi
    @Inject
    lateinit var directionsRepository: DirectionsRepository
    @Inject
    lateinit var locationProvider: LocationProvider
    @Inject
    lateinit var notificationsRepository:
NotificationsRepository
    @Inject
    lateinit var objectTypesRepository: ObjectTypesRepository
    @Inject
    lateinit var stationsRepository: StationsRepository
    @Inject
    @field:TokenPref
    lateinit var tokenPref: Preference<String>
    @Inject
    @field:StationsTimestampPref
    lateinit var stationsTimestampPref: Preference<Long>
    @Inject
    @field:EnabledObjectTypesPref
    lateinit var enabledObjectTypesPref: Preference<List<Int>>

    private val subscriptions = CompositeDisposable()

    private var userLocationMarker: Point? = null
    private var userLocationPlacemark: PlacemarkMapObject? =
null
    private var followUserLocation: Boolean = true
    private var stationInfoSnackbar: CustomSnackbar? = null
    private var routePolylineEncodedPath: String? = null

    private var directionsFrom: LatLng? = null
    private var directionsTo: LatLng? = null

    private lateinit var rxPermissions: RxPermissions

    private var fragmentMapBinding: FragmentMapBinding? = null
    private var viewDirectionBinding: ViewDirectionsBinding? =

```

```

null
    private var viewStationInfoBinding: ViewStationInfoBinding?
= null

    private lateinit var drivingRouter: DrivingRouter
    private var drivingSession: DrivingSession? = null
    private var polylineMap: PolylineMapObject? = null
    private var placemarks: ArrayList<PlacemarkMapObject>? =
null

    private lateinit var locationManager: LocationManager

    private var progressRouteDialog: AlertDialog? = null

    @SuppressLint("ResourceType")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        MapKitFactory.initialize(this.context)
        followUserLocation =
savedInstanceState?.getBoolean(STATE_FOLLOW_USER_LOCATION) ?:
true
        routePolylineEncodedPath =
savedInstanceState?.getString(STATE_ROUTE_POLYLINE_ENCODED_PATH)
        directionsFrom =
savedInstanceState?.getParcelable(STATE_DIRECTIONS_FROM)
        directionsTo =
savedInstanceState?.getParcelable(STATE_DIRECTIONS_TO)
        rxPermissions = RxPermissions(this)
        rxPermissions.setLogging(BuildConfig.DEBUG)
        locationManager =
MapKitFactory.getInstance().createLocationManager()
        drivingRouter =
DirectionsFactory.getInstance().createDrivingRouter(DrivingRoute
rType.COMBINED)
        placemarks = arrayListOf()
    }

    override fun onCreateView(inflater: LayoutInflater,
container: ViewGroup?, savedInstanceState: Bundle?): View {
        val bindingMap = FragmentMapBinding.inflate(inflater,
container, false)
        fragmentMapBinding = bindingMap
        viewDirectionBinding =
ViewDirectionsBinding.inflate(inflater, container, false)
        viewStationInfoBinding =
ViewStationInfoBinding.inflate(inflater, container, false)
        return bindingMap.root
    }

    override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
        super.onViewCreated(view, savedInstanceState)

```

```

        setHasOptionsMenu(true)
        setUpToolbar()
        setUpMap()
        setUpDirectionsForm()
        observeNewNotifications()
        observeStationsUpdates()

        showUserLocation()
        showStations()
    }

    override fun onCreateOptionsMenu(menu: Menu, inflater:
MenuInflater) {
        inflater.inflate(R.menu.map, menu)
        super.onCreateOptionsMenu(menu, inflater)
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean
    {
        when (item.itemId) {
            R.id.menu_directions -> openDirectionsScreen()
            R.id.menu_notifications -> openNotificationsScreen()
            R.id.menu_object_types -> openObjectTypesChooser()
            R.id.menu_sign_out -> showSignOutConfirmation()
            else -> return super.onOptionsItemSelected(item)
        }
        return true
    }

    private fun openObjectTypesChooser() = fragmentManager?.let
    {
        ObjectTypesChooserDialog().show(it,
"object_types_chooser")
    }

    private fun openDirectionsScreen() {
        fragmentMapBinding?.directionsFormLayout?.visible =
        if
        (fragmentMapBinding?.directionsFormLayout?.visible == true)
        false else true
    }

    private fun setUpToolbar() {
        (activity as
AppCompatActivity).setSupportActionBar(fragmentMapBinding?.toolb
ar)

        fragmentMapBinding?.toolbar?.title =
"$getString(R.string.app_name) v${BuildConfig.VERSION_NAME}"
    }

    private fun setUpMap() {
        fragmentMapBinding?.myLocationButton?.setOnClickListener

```

```

{ animateToUserLocation() }
    fragmentMapBinding?.myDelRouteBtn?.setOnClickListener {
deleteRoute() }
    fragmentMapBinding?.myDelRouteBtn?.visible = false
    fragmentMapBinding?.mapView?.map?.addInputListener(this)
    }

    private fun setUpDirectionsForm() {
        val btn =
fragmentMapBinding?.directionsFormLayout?.findViewById<Button>(R
.id.directionsButton)
        btn?.setOnClickListener {
            val pDirectionsTo =

fragmentMapBinding?.directionsFormLayout?.findViewById<SearchLay
out>(R.id.placeToDivider)?.point
            val pDirectionsFrom =

fragmentMapBinding?.directionsFormLayout?.findViewById<SearchLay
out>(R.id.placeFromDivider)?.point
            val searchLine =
fragmentMapBinding?.directionsFormLayout?.findViewById<EditText>
(R.id.placeView)

                if (pDirectionsFrom?.latitude != 0.0 &&
pDirectionsFrom?.longitude != 0.0 &&
                    pDirectionsTo?.latitude != 0.0 &&
pDirectionsTo?.longitude != 0.0 &&
                        searchLine?.text.toString() != ""
                    ) {
                        showRoute(pDirectionsFrom!!, pDirectionsTo!!)

fragmentMapBinding?.directionsFormLayout?.visible = false
                    } else if (pDirectionsFrom?.latitude == 0.0 &&
pDirectionsFrom?.longitude == 0.0 ||
                        searchLine?.text.toString() == ""
                    ) {
                        showRoute(userLocationMarker!!, pDirectionsTo!!)

fragmentMapBinding?.directionsFormLayout?.visible = false
                    }
                    val inputMethodManager =
context?.getSystemService(Activity.INPUT_METHOD_SERVICE) as
InputMethodManager

inputMethodManager.hideSoftInputFromWindow(view?.windowToken, 0)
                }
            }

            override fun onActivityResult(requestCode: Int, resultCode:
Int, data: Intent?) {}

```

					ПГУ1.090301.47.001 ПЗ	Лист
						65
Изм.	Лист	№ докум.	Подпись	Дата		

```

        private fun showRouteTo(station: Station) {
            userLocationMarker?.let { showRoute(it,
Point(station.latitude, station.longitude)) }
        }

        private fun showRoute(pointFrom: Point, pointTo: Point) {
            val drivingOptions = DrivingOptions().setRoutesCount(3)
            val requestPoints = listOf(
                RequestPoint(pointFrom, RequestPointType.WAYPOINT,
null, null, null),
                RequestPoint(pointTo, RequestPointType.WAYPOINT,
null, null, null)
            )
            drivingSession?.let { drivingSession?.cancel() }
            drivingSession = drivingRouter.requestRoutes(
                requestPoints,
                drivingOptions,
                VehicleOptions().setVehicleType(VehicleType.TRUCK),
                this
            )
            progressRouteDialog =
AlertDialog.Builder(context!!).setMessage(R.string.progress_buil
ding_route).show()
        }

        @SuppressWarnings("CheckResult")
        override fun onDrivingRoutes(routes:
MutableList<DrivingRoute>) {
            subscriptions.add(
                Completable.create {
                    try { it.onComplete() }
                    catch (ex: Exception) { it.onError(ex) }
                }
            ).observeOn(AndroidSchedulers.mainThread())
            .doAfterTerminate { progressRouteDialog?.dismiss() }
            .subscribe(
                {
                    val map = fragmentMapBinding?.mapView?.map
polylineMap?.let {
map?.mapObjects?.remove(it) }
                    polylineMap =
map?.mapObjects?.addPolyline(routes.last().geometry)
                    val geometry =
Geometry.fromPolyline(routes.last().geometry)
                    val position = map?.cameraPosition(
                        geometry, 0f, 0f,
                        ScreenRect(
                            ScreenPoint(100f, 100f),
                            ScreenPoint(
fragmentMapBinding?.mapView?.width?.toFloat()!! - 100,

```

```

fragmentMapBinding?.mapView?.height?.toFloat()!! - 100,
        )
    )
    map?.move(position!!, Animation(SMOOTH,
0.5f), null)
        fragmentMapBinding?.myDelRouteBtn?.visible =
true
    },
    { error ->
        Timber.w(error, "Failed to get route")
        Toast.makeText(this.activity,
R.string.error_no_route, Toast.LENGTH_LONG).show()
    }
    )
    )
}

override fun onDrivingRoutesError(p0: Error) {
    progressRouteDialog?.dismiss()
    Toast.makeText(this.activity, "Нет подключения к
сервисам Yandex Map", Toast.LENGTH_LONG).show()
}

private fun onMapReady() {

MapKitFactory.getInstance().resetLocationManagerToDefault()
    locationManager.requestSingleUpdate(this)
}

override fun onLocationUpdated(location:
com.yandex.mapkit.location.Location) {
    userLocationMarker = Point(location.position.latitude,
location.position.longitude)

    if (userLocationPlacemark == null) {
        userLocationMarker?.let {
            userLocationPlacemark =
fragmentMapBinding?.mapView?.map?.mapObjects?.addPlacemark()?.ap
ply {
                this.geometry = it

setIcon(ImageProvider.fromResource(mainActivity,
R.drawable.marker_user_location))
            }
        } else {
            userLocationMarker?.let {
userLocationPlacemark?.geometry = it }
        }
    }
}

```

					ПГУ1.090301.47.001 ПЗ	Лист
						67
Изм.	Лист	№ докум.	Подпись	Дата		

```

        override fun onLocationStatusUpdated(p0: LocationStatus) {}

        private fun showStationInfo(station: Station) {
            fragmentMapBinding?.mapView?.map?.move(
                CameraPosition(Point(station.latitude,
station.longitude), 13.0f, 0.0f, 0.0f),
                Animation(SMOOTH, 0.5f),
                null
            )

            stationInfoSnackbar = CustomSnackbar(
                parentView =
fragmentMapBinding?.coordinatorLayout!!,
                layout = R.layout.view_station_info,
                duration = Snackbar.LENGTH_INDEFINITE,
                background = R.color.window_background,
                swipeToDismiss = true
            ).apply {
                (contentView as StationInfoView).bindTo(
                    station = station,
                    viewStationInfoBinding =
viewStationInfoBinding!!,
                    onDirectionsButtonClick = {
                        hideStationInfo()
                        showRouteTo(station)
                    },
                    onMapsButtonClick = { showOnMaps(station) }
                )
                show()
            }
        }

        private fun hideStationInfo() {
            stationInfoSnackbar?.dismiss()
        }

        private fun animateToUserLocation() {
            locationManager.requestSingleUpdate(this)
            userLocationMarker?.let {
                fragmentMapBinding?.mapView?.map?.move(
                    CameraPosition(it, 14.0f, 0.0f, 0.0f),
                    Animation(SMOOTH, 1f),
                    null
                )
                followUserLocation = true
            }
        }

        private fun deleteRoute() {
            if (polylineMap != null) {
                fragmentMapBinding?.mapView?.map?.mapObjects?.remove(polylineMap

```

					ПГУ1.090301.47.001 ПЗ	Лист
						68
Изм.	Лист	№ докум.	Подпись	Дата		

```

!!)
        polylineMap = null
        fragmentMapBinding?.myDelRouteBtn?.visible = false
    }
}

@SuppressLint("MissingPermission", "RxSubscribeOnError")
private fun showUserLocation() =
    rxPermissions.requestEachCombined(ACCESS_FINE_LOCATION,
ACCESS_COARSE_LOCATION, INTERNET)
        .subscribe { permission ->
            if (permission.granted) {
                onMapReady()
            }
        }
}

private fun showStations() {
    subscriptions.add(
        Observables.combineLatest(
            stationsRepository.stations(),
            objectTypesRepository.objectTypes(),
            enabledObjectTypesPref.asObservable()
        )
        .observeOn(AndroidSchedulers.mainThread())
        .subscribe(
            { (stations, objectTypes,
enabledObjectTypes) ->

fragmentMapBinding?.mapView?.map?.setStations(stations,
objectTypes, enabledObjectTypes)
                },
            { error -> Timber.w(error, "Failed to show
stations") }
        )
    )
    if (!stationsTimestampPref.isSet) {
        updateStations()
    }
}

@Suppress("unused")
private fun Map.setStations(stations: List<Station>,
objectTypes: List<ObjectType>, enabledObjectTypes: List<Int>) {
    val map = fragmentMapBinding?.mapView?.map
    placemarks?.let {
        map?.mapObjects?.clear()
        hideStationInfo()
        polylineMap = null
        fragmentMapBinding?.myDelRouteBtn?.visible = false
        userLocationMarker?.let {
            userLocationPlacemark =

```

```

fragmentMapBinding?.mapView?.map?.mapObjects?.addPlacemark()?.apply {
    this.geometry = it

    setIcon(ImageProvider.fromResource(mainActivity,
        R.drawable.marker_user_location))
    }
    it.clear()
}
stations.filter {
enabledObjectTypes.contains(it.objectTypeId) }
    .map {
        map?.mapObjects?.addPlacemark()?.apply {
            this.geometry = Point(it.latitude,
it.longitude)
            this.userData = it

            val objectType = objectTypes.find {
objectType -> it.objectTypeId == objectType.id }
            val decodedString =
Base64.decode(objectType?.image, Base64.DEFAULT)
            if (objectType != null &&
objectType.image.isNotEmpty()) {
                val bitmap =
BitmapFactory.decodeByteArray(decodedString, 0,
decodedString.size)

this.setIcon(ImageProvider.fromBitmap(bitmap))
            } else {
                when (it.priority) {
                    1 ->
setIcon(ImageProvider.fromResource(mainActivity,
R.drawable.mark_green))
                    2 ->
setIcon(ImageProvider.fromResource(mainActivity,
R.drawable.mark_yellow))
                    else ->
setIcon(ImageProvider.fromResource(mainActivity,
R.drawable.mark_red))
                }
            }

            this.addTapListener(this@MapScreenFragment)
        }?.let { it1 -> placemarks?.add(it1) }
    }
}

override fun onMapObjectTap(marker: MapObject, point:
Point): Boolean {
    showStationInfo((marker as PlacemarkMapObject).userData
as Station)
}

```

```

        return true
    }

    override fun onMapTap(p0: Map, p1: Point) {
        hideStationInfo()
    }

    override fun onMapLongTap(p0: Map, p1: Point) {}

    override fun onDestroyView() {
        subscriptions.clear()
        super.onDestroyView()
    }

    override fun onSaveInstanceState(outState: Bundle) {
        super.onSaveInstanceState(outState)
        outState.putBoolean(STATE_FOLLOW_USER_LOCATION,
followUserLocation)
        outState.putString(STATE_ROUTE_POLYLINE_ENCODED_PATH,
routePolylineEncodedPath)
        outState.putParcelable(STATE_DIRECTIONS_FROM,
directionsFrom)
        outState.putParcelable(STATE_DIRECTIONS_TO,
directionsTo)
    }

    private fun openNotificationsScreen() {
        mainActivity?.addScreen(NotificationsFragment())
    }

    private fun showSignOutConfirmation() {
        AlertDialog.Builder(this.requireActivity()).also {
            it.setMessage(R.string.alert_sign_out_confirmation)
            it.setPositiveButton(R.string.button_yes) { _, _ ->
signOut() }
            it.setNegativeButton(R.string.button_no, null)
        }.show()
    }

    private fun signOut() {
        val progressDialog =
AlertDialog.Builder(this.requireActivity()).setMessage(R.string.
progress_sign_out).show()
        subscriptions.add(
            api.logout(tokenPref.get())
                .doAfterTerminate {
                    tokenPref.delete()
                }
        )

        notificationsRepository.removeAllNotifications()
    }

    .subscribe(

```

```

        { Timber.i("User was successfully sign out")
    },
        { error -> Timber.w(error, "Failed to sign
out") }
    )
    )
    progressDialog.dismiss()
    mainActivity?.openScreen(LoginFragment(), false)
}

private fun observeNewNotifications() {
    subscriptions.add(
        notificationsRepository.newNotifications()
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(
                { notification ->
showNotificationAlert(notification) },
                { error -> Timber.w(error) }
            )
    )
}

private fun showNotificationAlert(notification:
Notification) {
    AlertDialog.Builder(this.requireActivity()).also {

it.setMessage(getString(R.string.alert_title_notification))
        it.setPositiveButton(R.string.button_ok) { _, _ ->
readNotification(notification) }
        it.setNegativeButton(R.string.button_postpone, null)
    }.show()
}

private fun readNotification(notification: Notification) {
    subscriptions.add(

notificationsRepository.readNotification(notification)
        .subscribe(
            {},
            { error -> Timber.w(error, "Failed to read
notification") }
        )
    )
}

private fun observeStationsUpdates() {
    subscriptions.add(
        stationsRepository.stationsUpdates()
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(
                { showStationsUpdateAlert() },
                { error -> Timber.w(error) }
            )
    )
}

```

```

        )
    )
}

private fun showStationsUpdateAlert() {
    AlertDialog.Builder(this.requireActivity()).also {
        it.setMessage(R.string.alert_message_has_stations_update)
        it.setPositiveButton(R.string.button_update) { _, _
        -> updateStations() }
        it.setNegativeButton(R.string.button_cancel, null)
    }.show()
}

private fun updateStations() {
    val progressDialog =
    AlertDialog.Builder(this.requireActivity()).setMessage(R.string.
    progress_loading_stations).show()
    subscriptions.add(

    Singles.zip(objectTypesRepository.updateObjectTypes(),
    stationsRepository.updateStations())
        .observeOn(AndroidSchedulers.mainThread())
        .subscribe(
            {},
            { error ->
                Timber.w(error, "Failed to update
stations")
                showLoadingStationsError()
            }
        )
    )
    progressDialog.dismiss()
}

private fun showLoadingStationsError() {
    AlertDialog.Builder(this.requireActivity()).also {
        it.setMessage(R.string.error_loading_stations)
        it.setPositiveButton(R.string.button_retry) { _, _ -
        > updateStations() }
        it.setNegativeButton(R.string.button_cancel, null)
    }.show()
}

private fun showOnMaps(station: Station) {
    val uri =
    Uri.parse("geo:0,0?q=${station.latitude},${station.longitude} (${
    station.title})")
    val intent = Intent(Intent.ACTION_VIEW, uri)
    try {
        startActivity(intent)
    } catch (e: ActivityNotFoundException) {

```

```

        Toast.makeText(this.activity,
R.string.error_no_maps, Toast.LENGTH_LONG).show()
    }

    }

    private companion object {
        private const val STATE_FOLLOW_USER_LOCATION =
"follow_user_location"
        private const val STATE_ROUTE_POLYLINE_ENCODED_PATH =
"route_polyline_encoded_path"
        private const val STATE_DIRECTIONS_FROM =
"directions_from"
        private const val STATE_DIRECTIONS_TO = "directions_to"
    }
}

```

Файл «SearchAdapter.kt»

```

package ru.agattrans.ui.map

import android.content.Context
import android.widget.ArrayAdapter
import android.widget.Filter

class SearchAdapter(context: Context?, resource: Int, objects:
ArrayList<String>?) : ArrayAdapter<String>(
    context!!, resource, objects!!
) {
    override fun getFilter(): Filter {
        return object : Filter() {
            override fun performFiltering(constraint:
CharSequence?): FilterResults? = null

            override fun publishResults(constraint:
CharSequence?, results: FilterResults?) {}
        }
    }
}

```

Файл «SearchLayout.kt»

```

package ru.agattrans.ui.map

import android.annotation.SuppressLint
import android.content.Context
import android.text.Editable
import android.text.TextWatcher
import android.util.AttributeSet
import android.view.LayoutInflater
import android.widget.*
import com.yandex.mapkit.geometry.BoundingBox
import com.yandex.mapkit.geometry.Point
import com.yandex.mapkit.search.*
import com.yandex.runtime.Error

```

```

@SuppressLint("ResourceType")
class SearchLayout :
    LinearLayout,
    SuggestSession.SuggestListener
{
    private val RESULT_NUMBER_LIMIT = 5
    private var searchManager: SearchManager
    private var suggestSession: SuggestSession

    private var searchLine: AutoCompleteTextView
    private var suggestResult: ArrayList<String> = arrayListOf()
    private var suggestItems: ArrayList<SuggestItem> =
arrayListOf()
    private var suggestResultAdapter: SearchAdapter

    var point: Point = Point(0.0, 0.0)
        get() = field
        set(value) { field = value }

    private val BOX_SIZE = 0.2
    private val BOUNDING_BOX = BoundingBox(
        Point(69.402021 - BOX_SIZE, 32.444884 - BOX_SIZE),
        Point(35.681729 + BOX_SIZE, 139.753927 + BOX_SIZE)
    )
    private val SEARCH_OPTIONS: SuggestOptions =
SuggestOptions().setSuggestTypes(
        SuggestType.GEO.value or
        //SuggestType.BIZ.value or
        SuggestType.TRANSIT.value
    )

    constructor(context: Context) : super(context) {}
    constructor(context: Context, attrs: AttributeSet) :
super(context, attrs) {}
    constructor(context: Context, attrs: AttributeSet,
defStyleAttr: Int) : super(context, attrs, defStyleAttr) {}

    init {

LayoutInflater.from(context).inflate(ru.agattrans.R.layout.searc
h_address_layout, this)
        searchManager =
SearchFactory.getInstance().createSearchManager(SearchManagerTyp
e.COMBINED)
        suggestSession = searchManager.createSuggestSession()

        suggestResultAdapter = SearchAdapter(
            this.context,
            android.R.layout.simple_list_item_1,
            suggestResult
        )
    }
}

```

```

        suggestResultAdapter.setNotifyOnChange(true)

        searchLine =
this.findViewById<ru.agattrans.R.id.placeView>()
        searchLine.addTextChangedListener(object : TextWatcher {
            override fun afterTextChanged(editable: Editable) {
                suggestSession.suggest(editable.toString(),
BOUNDING_BOX, SEARCH_OPTIONS, this@SearchLayout)
            }
            override fun beforeTextChanged(s: CharSequence?,
start: Int, count: Int, after: Int) {}
            override fun onTextChanged(s: CharSequence?, start:
Int, before: Int, count: Int) {}
        })
        searchLine.setOnFocusChangeListener { _, hasFocus ->
            if (!hasFocus) {
                if (suggestResult.isNotEmpty()) {
                    setSuggestText(0)
                } else {
                    searchLine.setText("")
                    searchLine.setSelection(0)
                }
                suggestResult.clear()
                suggestItems.clear()
                suggestResultAdapter.notifyDataSetChanged()
            }
        }
        searchLine.setAdapter(suggestResultAdapter)

        when (this.id) {
            ru.agattrans.R.id.placeFromDivider ->
searchLine.hint = "Откуда"
            ru.agattrans.R.id.placeToDivider -> searchLine.hint
= "Куда"
        }

        searchLine.setOnItemClickListener { _, _, position, _ ->
            setSuggestText(position)
            suggestSession.suggest(suggestResult[position],
BOUNDING_BOX, SEARCH_OPTIONS, this@SearchLayout)
        }
    }

    private fun setSuggestText(position: Int) {
        searchLine.setText(suggestResult[position])
        searchLine.setSelection(suggestResult[position].length)
        point = suggestItems[position].center!!
    }

    override fun onResponse(suggest: SuggestResponse) {
        suggestResult.clear()
        suggestItems.clear()
    }

```

```

        run {
            suggest.items.forEachIndexed { index, item ->
                if (index == RESULT_NUMBER_LIMIT) return@run
                suggestResult.add(item.displayText!!)
                suggestItems.add(item)
            }
        }
        suggestResultAdapter.notifyDataSetChanged()
        suggestResultAdapter.setNotifyOnChange(true)
    }

    override fun onError(p0: Error) {}
}

```

Файл «SationInfoView.kt»

```

package ru.agattrans.ui.map

import android.content.Context
import androidx.core.content.ContextCompat
import android.util.AttributeSet
import android.widget.FrameLayout
import android.widget.TextView
import ru.agattrans.databinding.ViewStationInfoBinding
import ru.agattrans.R
import ru.agattrans.data.model.Station
import ru.agattrans.util.visible

class StationInfoView(context: Context, attrs: AttributeSet) :
    FrameLayout(context, attrs) {
    fun bindTo(station: Station, viewStationInfoBinding:
        ViewStationInfoBinding,
            onDirectionsButtonClick: () -> Unit,
            onMapsButtonClick: () -> Unit) {
        with(station) {
            this@StationInfoView.findViewById<TextView>(R.id.titleView).text
            = title

            this@StationInfoView.findViewById<TextView>(R.id.addressView).te
            xt = address

            this@StationInfoView.findViewById<TextView>(R.id.priorityView).s
            etText(
                when (priority) {
                    1 -> R.string.priority_high
                    2 -> R.string.priority_medium
                    else -> R.string.priority_low
                }
            )

            this@StationInfoView.findViewById<TextView>(R.id.priorityView).s
            etTextColor(ContextCompat.getColor(context,

```

					ПГУ1.090301.47.001 ПЗ	Лист
						77
Изм.	Лист	№ докум.	Подпись	Дата		

```

        when (priority) {
            1 -> R.color.priority_high
            2 -> R.color.priority_medium
            else -> R.color.priority_low
        }
    })

    this@StationInfoView.findViewById<TextView>(R.id.limitView).text
    =
        if (limit > 0)
            resources.getString(R.string.format_limit, limit)
        else resources.getString(R.string.no_limit)

    this@StationInfoView.findViewById<TextView>(R.id.commentView).te
    xt = comment

    this@StationInfoView.findViewById<TextView>(R.id.commentView).vi
    sible = !comment.isNullOrBlank()

    this@StationInfoView.findViewById<TextView>(R.id.directionButton
    ).setOnClickListener {
        onDirectionsButtonClick.invoke()
    }

    this@StationInfoView.findViewById<TextView>(R.id.mapsButton).set
    OnClickListener {
        onMapsButtonClick.invoke()
    }
}
}
}
}

```

Файл «NotificationsFragment.kt»

```

package ru.agattrans.ui.notification

import android.graphics.Canvas
import android.graphics.Paint
import android.os.Bundle
import androidx.core.content.ContextCompat
import androidx.recyclerview.widget.DividerItemDecoration
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import androidx.recyclerview.widget.RecyclerView.ViewHolder
import androidx.recyclerview.widget.ItemTouchHelper
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.LinearLayout
import dagger.android.support.DaggerFragment

```

```

import io.reactivex.android.schedulers.AndroidSchedulers
import io.reactivex.disposables.CompositeDisposable
import ru.agattrans.R
import ru.agattrans.data.NotificationsRepository
import ru.agattrans.data.model.Notification
import ru.agattrans.databinding.FragmentNotificationsBinding
import ru.agattrans.util.bitmapFromVectorDrawable
import ru.agattrans.util.dpToPx
import ru.agattrans.util.mainActivity
import timber.log.Timber
import javax.inject.Inject

class NotificationsFragment : DaggerFragment() {
    @Inject lateinit var notificationsRepository:
NotificationsRepository

    private lateinit var notificationsAdapter:
NotificationsAdapter
    private val subscriptions = CompositeDisposable()

    private val notificationsTouchCallback = object :
ItemTouchHelper.SimpleCallback(0, ItemTouchHelper.LEFT or
ItemTouchHelper.RIGHT) {
        override fun onMove(recyclerView: RecyclerView, viewHolder:
ViewHolder, target: ViewHolder) = false

        override fun onSwiped(viewHolder: ViewHolder, direction:
Int) {
            val position = viewHolder.adapterPosition
            readNotification(notificationsAdapter.items[position])
        }

        override fun onChildDraw(c: Canvas, recyclerView:
RecyclerView, viewHolder: ViewHolder, dX: Float, dY: Float,
actionState: Int, isCurrentlyActive: Boolean) {
            val itemView = viewHolder.itemView
            val paint = Paint()
            paint.color = ContextCompat.getColor(context!!,
R.color.notification_done)
            val doneIcon =
context!!.bitmapFromVectorDrawable(R.drawable.ic_done_white_24dp
)
            if (dX > 0) {
                c.drawRect(itemView.left.toFloat(),
itemView.top.toFloat(), dX, itemView.bottom.toFloat(), paint)
                c.drawBitmap(doneIcon,
                    itemView.left.toFloat() + context!!.dpToPx(16),
                    itemView.top.toFloat() + (itemView.bottom.toFloat()
- itemView.top.toFloat() - doneIcon.height) / 2,
                    paint)
            } else {
                c.drawRect(itemView.right.toFloat() + dX,

```

					ПГУ1.090301.47.001 ПЗ	Лист
						79
Изм.	Лист	№ докум.	Подпись	Дата		

```

itemView.top.toFloat(), itemView.right.toFloat(),
itemView.bottom.toFloat(), paint)
    c.drawBitmap(doneIcon,
        itemView.right.toFloat() - context!!.dpToPx(16) -
doneIcon.width,
        itemView.top.toFloat() + (itemView.bottom.toFloat()
- itemView.top.toFloat() - doneIcon.height) / 2,
        paint)
    }
    super.onChildDraw(c, recyclerView, viewHolder, dX, dY,
actionState, isCurrentlyActive)
}
}

private var fragmentNotificationBinding:
FragmentNotificationsBinding? = null

override fun onCreateView(inflater: LayoutInflater, container:
ViewGroup?, savedInstanceState: Bundle?): View {
    val binding =
FragmentNotificationsBinding.inflate(inflater, container, false)
    fragmentNotificationBinding = binding
    return binding.root
}

override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    fragmentNotificationBinding?.toolbar?.setNavigationOnClickListen
er {
        mainActivity?.onBackPressed()
    }
    notificationsAdapter = NotificationsAdapter()
    fragmentNotificationBinding?.recyclerView?.run {

ItemTouchHelper(notificationsTouchCallback).attachToRecyclerView
(this)
        addItemDecoration(DividerItemDecoration(context,
LinearLayout.VERTICAL))
        layoutManager = LinearLayoutManager(context)
        adapter = notificationsAdapter
        setEmptyView(fragmentNotificationBinding?.emptyView)
    }
    loadNotifications()
}

private fun loadNotifications() {
    subscriptions.add(
        notificationsRepository.notifications()
            .map { it.sortedByDescending { it.date } }
            .observeOn(AndroidSchedulers.mainThread())
    )
}

```

```

        .subscribe(
            { notifications ->
notificationsAdapter.swapItems(notifications) },
            { error -> Timber.w(error, "Failed to load
notifications") }
        )
    )
}

private fun readNotification(notification: Notification) {
    subscriptions.add(
        notificationsRepository.readNotification(notification)
            .subscribe(
                {},
                { error -> Timber.w(error, "Failed to read
notification") }
            )
    )
}

override fun onDestroyView() {
    subscriptions.clear()
    super.onDestroyView()
}
}

```

Файл «NotificationsDiffCallback.kt»

```

package ru.agattrans.ui.notification

import androidx.recyclerview.widget.DiffUtil
import ru.agattrans.data.model.Notification

class NotificationsDiffCallback(
    private val oldList: List<Notification>,
    private val newList: List<Notification>
) : DiffUtil.Callback() {

    override fun areItemsTheSame(oldItemPosition: Int,
newItemPosition: Int) =
        oldList[oldItemPosition].id == newList[newItemPosition].id

    override fun areContentsTheSame(oldItemPosition: Int,
newItemPosition: Int) =
        oldList[oldItemPosition] == newList[newItemPosition]

    override fun getOldListSize() = oldList.size

    override fun getNewListSize() = newList.size
}

```

Файл «NotificationsAdapter.kt»

					ПГУ1.090301.47.001 ПЗ	Лист
						81
Изм.	Лист	№ докум.	Подпись	Дата		

```

package ru.agattrans.ui.notification

import androidx.recyclerview.widget.DiffUtil
import androidx.recyclerview.widget.RecyclerView
import android.view.LayoutInflater
import android.view.ViewGroup
import org.threeten.bp.ZoneId
import org.threeten.bp.format.DateTimeFormatter
import ru.agattrans.data.model.Notification
import ru.agattrans.databinding.ItemNotificationBinding
import
ru.agattrans.ui.notification.NotificationsAdapter.ViewHolder

class NotificationsAdapter() :
RecyclerView.Adapter<ViewHolder>() {
    private val dateFormatter =
DateTimeFormatter.ofPattern("dd.MM.yyyy HH:mm")
    var items: List<Notification> = emptyList()
    private set

    fun swapItems(items: List<Notification>) {
        val diffResult =
DiffUtil.calculateDiff(NotificationsDiffCallback(this.items,
items))
        this.items = items
        diffResult.dispatchUpdatesTo(this)
    }

    inner class ViewHolder(val binding: ItemNotificationBinding) :
RecyclerView.ViewHolder(binding.root)

        override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): ViewHolder {
            val inflater = LayoutInflater.from(parent.context)
            val binding = ItemNotificationBinding.inflate(inflater,
parent, false)
            return ViewHolder(binding)
        }

        override fun onBindViewHolder(holder: ViewHolder, position:
Int) {
            with(holder) {
                with(items[position]) {
                    binding.messageView.text = this.message
                    binding.dateView.text =
this.date.atZoneSameInstant(ZoneId.systemDefault()).format(dateF
ormatter)
                }
            }
        }
    }
}

```

```

        override fun getItemCount() = items.size
    }

```

Файл «ObjectTypesChooserDialog.kt»

```
package ru.agattrans.ui.settings
```

```

import android.app.AlertDialog
import android.app.Dialog
import android.os.Bundle
import com.f2prateek.rx.preferences2.Preference
import dagger.android.support.DaggerAppCompatActivity
import ru.agattrans.R
import ru.agattrans.data.ObjectTypesRepository
import ru.agattrans.data.preference.EnabledObjectTypesPref
import javax.inject.Inject

```

```

class ObjectTypesChooserDialog : DaggerAppCompatActivity() {
    @Inject lateinit var objectTypesRepository:
ObjectTypesRepository
    @Inject @field:EnabledObjectTypesPref lateinit var
enabledObjectTypesPref: Preference<List<Int>>

    override fun onCreateDialog(savedInstanceState: Bundle?):
Dialog {
        val objectTypes =
objectTypesRepository.objectTypes().blockingFirst()
        val enabledObjectTypes =
enabledObjectTypesPref.get().toMutableSet()
        return AlertDialog.Builder(context).apply {
            setTitle(R.string.object_types)
            setMultiChoiceItems(
                objectTypes.map { it.name }.toTypedArray(),
                objectTypes.map { enabledObjectTypes.contains(it.id)
}.toBooleanArray()
            ) { _, position, isChecked ->
                objectTypes[position].id.let { if (isChecked)
enabledObjectTypes.add(it) else enabledObjectTypes.remove(it) }
            }
            setPositiveButton(R.string.button_ok) { _, _ ->
                enabledObjectTypesPref.set(enabledObjectTypes.toList())
            }
            setNegativeButton(R.string.button_cancel, null)
        }.create()
    }
}

```

Файл «MainActivity.kt»

```
package ru.agattrans.ui
```

```

import android.os.Bundle
import androidx.fragment.app.Fragment

```

					ПГУ1.090301.47.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		83

```

import com.f2prateek.rx.preferences2.Preference
import dagger.android.support.DaggerAppCompatActivity
import ru.agattrans.R
import ru.agattrans.data.preference.TokenPref
import ru.agattrans.ui.login.LoginFragment
import ru.agattrans.ui.map.MapScreenFragment
import javax.inject.Inject

class MainActivity : DaggerAppCompatActivity() {
    @Inject @field:TokenPref lateinit var tokenPref:
    Preference<String>

    override fun onCreate(savedInstanceState: Bundle?) {
        setTheme(R.style.Theme_Agattrans)
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        if (currentScreen() == null) {
            openLaunchScreen()
        }
    }

    private fun openLaunchScreen() {
        val launchScreen = if (tokenPref.isSet) MapScreenFragment()
    else LoginFragment()
        openScreen(launchScreen, false)
    }

    fun openScreen(fragment: Fragment, addToBackStack: Boolean =
    true) {
        supportFragmentManager.beginTransaction()
            .replace(R.id.contentFrame, fragment)
            .apply { if (addToBackStack) addToBackStack(null) }
            .commitAllowingStateLoss()
    }

    fun addScreen(fragment: Fragment) {
        supportFragmentManager.beginTransaction()
            .add(R.id.contentFrame, fragment)
            .addToBackStack(null)
            .commitAllowingStateLoss()
    }

    private fun currentScreen() =
    supportFragmentManager.findFragmentById(R.id.contentFrame)
    }

```

Файл «UIModule.kt»

```

package ru.agattrans.ui

import dagger.Module
import dagger.android.ContributesAndroidInjector
import ru.agattrans.ui.login.LoginFragment

```

```
import ru.agattrans.ui.map.MapScreenFragment
import ru.agattrans.ui.notification.NotificationsFragment
import ru.agattrans.ui.settings.ObjectTypesChooserDialog
```

```
@Module
abstract class UiModule {

    @ContributesAndroidInjector
    abstract fun bindMainActivity(): MainActivity

    @ContributesAndroidInjector
    abstract fun bindLoginFragment(): LoginFragment

    @ContributesAndroidInjector
    abstract fun bindMapScreenFragment(): MapScreenFragment

    @ContributesAndroidInjector
    abstract fun bindNotificationsFragment():
NotificationsFragment

    @ContributesAndroidInjector
    abstract fun bindObjectTypesChooserDialog():
ObjectTypesChooserDialog
}
```

Файл «StationsRepository.kt»

```
package ru.agattrans.data

import com.f2prateek.rx.preferences2.Preference
import io.reactivex.Observable
import io.reactivex.Single
import io.reactivex.schedulers.Schedulers
import ru.agattrans.data.api.AgatTransApi
import ru.agattrans.data.db.AppDatabase
import ru.agattrans.data.model.Station
import ru.agattrans.data.preference.StationsTimestampPref
import ru.agattrans.data.preference.TokenPref
import timber.log.Timber
import java.util.concurrent.TimeUnit
import javax.inject.Inject
import javax.inject.Singleton

@Singleton
class StationsRepository @Inject constructor(
    private val api: AgatTransApi,
    private val db: AppDatabase,
    @TokenPref private val tokenPref: Preference<String>,
    @StationsTimestampPref private val stationsTimestampPref:
Preference<Long>
) {

    fun stations(): Observable<List<Station>> =
```

```

        db.stationsDao().getAll().toObservable()

fun updateStations(): Single<List<Station>> =
    api.stations(tokenPref.get())
        .doOnSuccess {
            db.stationsDao().run {
                deleteAll()
                insertAll(it.stations)
            }
            stationsTimestampPref.set(it.timestamp)
        }
        .map { it.stations }

fun stationsUpdates(): Observable<Unit> =
    Observable.interval(0, 6, TimeUnit.HOURS,
Schedulers.computation())
        .flatMapSingle {
            api.hasStationsUpdate(tokenPref.get(),
stationsTimestampPref.get())
                .map { it.hasUpdates }
                .onErrorReturn { error ->
                    Timber.w(error, "Failed to get stations
update")
                        false
                    }
        }
        .filter { it }
        .map { Unit }
}

```

Файл «ObjectTypesRepository.kt»

```

package ru.agattrans.data

import com.f2prateek.rx.preferences2.Preference
import io.reactivex.Observable
import io.reactivex.Single
import ru.agattrans.data.api.AgatTransApi
import ru.agattrans.data.db.AppDatabase
import ru.agattrans.data.model.ObjectType
import ru.agattrans.data.preference.EnabledObjectTypesPref
import ru.agattrans.data.preference.TokenPref
import javax.inject.Inject
import javax.inject.Singleton

@Singleton
class ObjectTypesRepository @Inject constructor(
    private val api: AgatTransApi,
    private val db: AppDatabase,
    @TokenPref private val tokenPref: Preference<String>,
    @EnabledObjectTypesPref private val enabledObjectTypesPref:
Preference<List<Int>>
) {

```

```

fun objectTypes(): Observable<List<ObjectType>> =
    db.objectTypesDao().getAll().toObservable()

fun updateObjectTypes(): Single<List<ObjectType>> =
    api.objectTypes(tokenPref.get())
        .doOnSuccess { objectTypes ->
            setUpEnabledObjectTypes(objectTypes)
            replaceAllObjectTypes(objectTypes)
        }

private fun replaceAllObjectTypes(objectTypes:
List<ObjectType>) = db.objectTypesDao().run {
    deleteAll()
    insertAll(objectTypes)
}

private fun setUpEnabledObjectTypes(objectTypes:
List<ObjectType>) {
    val enabledObjectTypes =
        if (enabledObjectTypesPref.isSet) {
            enabledObjectTypesPref.get().filter {
enabledObjectTypeId ->
                objectTypes.any { it.id == enabledObjectTypeId }
            }
        } else {
            objectTypes.map { it.id }
        }
    enabledObjectTypesPref.set(enabledObjectTypes)
}
}

```

Файл «NotificationsRepository.kt»

```

package ru.agattrans.data

import com.f2prateek.rx.preferences2.Preference
import io.reactivex.Completable
import io.reactivex.Observable
import io.reactivex.schedulers.Schedulers
import org.threeten.bp.OffsetDateTime
import org.threeten.bp.format.DateTimeFormatter
import ru.agattrans.data.api.AgatTransApi
import ru.agattrans.data.db.AppDatabase
import ru.agattrans.data.model.Notification
import ru.agattrans.data.preference.TokenPref
import timber.log.Timber
import java.util.concurrent.TimeUnit
import javax.inject.Inject

class NotificationsRepository @Inject constructor(
    private val api: AgatTransApi,
    private val db: AppDatabase,

```

```

        @TokenPref private val tokenPref: Preference<String>
    ) {

        fun newNotifications(): Observable<Notification> =
            Observable.interval(0, 1, TimeUnit.HOURS,
                Schedulers.computation())
                .doOnNext { Timber.i("new notifications #\$it") }
                .flatMapSingle {
                    api.notifications(tokenPref.get())
                        .onErrorReturn { error ->
                            Timber.w(error, "Failed to get new
notifications")
                                emptyList()
                            }
                }
                .doOnNext { notifications ->
                    db.notificationsDao().insertAll(notifications)
                }
                .flatMapIterable { it }

        fun notifications(): Observable<List<Notification>> =
            db.notificationsDao().getAll().toObservable()

        fun readNotification(notification: Notification): Completable
        =
            api.notificationRead(tokenPref.get(), notification.id,
                OffsetDateTime.now().format(DateTimeFormatter.ISO_OFFSET_DATE_TI
                ME))
                .doAfterTerminate {
                    db.notificationsDao().delete(notification) }

        fun removeAllNotifications() =
            db.notificationsDao().deleteAll()
    }

```

Файл «TokenPref.kt»

```

package ru.agattrans.data.preference

import javax.inject.Qualifier
import kotlin.annotation.AnnotationRetention.RUNTIME

@Qualifier
@Retention(RUNTIME)
annotation class TokenPref

```

Файл «StationsTimestampPref.kt»

```

package ru.agattrans.data.preference

import javax.inject.Qualifier
import kotlin.annotation.AnnotationRetention.RUNTIME

@Qualifier

```

```

@Retention(RUNTIME)
annotation class StationsTimestampPref

    Файл «PreferenceModule.kt»

package ru.agattrans.data.preference

import android.app.Application
import android.content.SharedPreferences
import android.preference.PreferenceManager
import com.f2prateek.rx.preferences2.Preference
import com.f2prateek.rx.preferences2.RxSharedPreferences
import com.squareup.moshi.Moshi
import com.squareup.moshi.Types
import dagger.Module
import dagger.Provides
import ru.agattrans.data.preference.converter.MoshiConverter
import javax.inject.Singleton

@Module
class PreferencesModule {

    @Provides
    @Singleton
    fun provideSharedPreferences(app: Application):
    SharedPreferences =
        PreferenceManager.getDefaultSharedPreferences(app)

    @Provides
    @Singleton
    fun provideRxSharedPreferences(prefs: SharedPreferences):
    RxSharedPreferences =
        RxSharedPreferences.create(prefs)

    @Provides
    @TokenPref
    fun provideTokenPref(rxPrefs: RxSharedPreferences):
    Preference<String> =
        rxPrefs.getString("token")

    @Provides
    @EnabledObjectTypesPref
    fun provideEnabledObjectTypesPref(rxPrefs:
    RxSharedPreferences, moshi: Moshi): Preference<List<Int>> =
        rxPrefs.getObject("enabled_object_types", emptyList()),
    MoshiConverter<List<Int>>(moshi,
    Types.newParameterizedType(List::class.java,
    Int::class.javaObjectType)))

    @Provides
    @StationsTimestampPref
    fun provideStationsTimestampPref(rxPrefs:
    RxSharedPreferences): Preference<Long> =

```

					ПГУ1.090301.47.001 ПЗ	Лист
						89
Изм.	Лист	№ докум.	Подпись	Дата		

```

        rxPrefs.getLong("stations_timestamp")
    }

```

Файл «EnabledObjectTypesPref.kt»

```

package ru.agattrans.data.preference

import javax.inject.Qualifier
import kotlin.annotation.AnnotationRetention.RUNTIME

@Qualifier
@Retention(RUNTIME)
annotation class EnabledObjectTypesPref

```

Файл «MoshiConverter.kt»

```

package ru.agattrans.data.preference.converter

import com.f2prateek.rx.preferences2.Preference
import com.squareup.moshi.JsonAdapter
import com.squareup.moshi.Moshi
import java.lang.reflect.Type

class MoshiConverter<T : Any>(private val adapter:
JsonAdapter<T>) : Preference.Converter<T> {

    constructor(moshi: Moshi, type: Class<T>) :
this(moshi.adapter(type))

    constructor(moshi: Moshi, type: Type) :
this(moshi.adapter<T>(type))

    override fun serialize(value: T): String =
adapter.toJson(value)

    override fun deserialize(serialized: String): T =
checkNotNull(adapter.fromJson(serialized))
}

```

Файл «Station.kt»

```

package ru.agattrans.data.model

import androidx.room.Entity
import androidx.room.PrimaryKey
import com.squareup.moshi.Json

@Entity(tableName = "stations")
data class Station(
    @PrimaryKey val id: Int,
    val title: String,
    val address: String,
    val latitude: Double,
    val longitude: Double,
    val priority: Int,

```

					ПГУ1.090301.47.001 ПЗ	Лист
						90
Изм.	Лист	№ докум.	Подпись	Дата		

```

        val limit: Int,
        val comment: String?,
        @Json(name = "objecttype_id") val objectTypeId: Int
    )

```

Файл «ObjectType.kt»

```

package ru.agattrans.data.model

import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "object_types")
data class ObjectType(
    @PrimaryKey val id: Int,
    val name: String,
    val image: String
)

```

Файл «Notification.kt»

```

package ru.agattrans.data.model

import androidx.room.Entity
import androidx.room.PrimaryKey
import org.threeten.bp.OffsetDateTime

@Entity(tableName = "notifications")
data class Notification(
    @PrimaryKey val id: Int,
    val message: String,
    val date: OffsetDateTime
)

```

Файл «AppDatabase.kt»

```

package ru.agattrans.data.db

import androidx.room.Database
import androidx.room.RoomDatabase
import androidx.room.TypeConverters
import ru.agattrans.data.model.Notification
import ru.agattrans.data.model.ObjectType
import ru.agattrans.data.model.Station
import ru.agattrans.util.room.OffsetDateTimeConverter

@Database(entities = [Notification::class, ObjectType::class,
    Station::class], version = 3)
@TypeConverters(OffsetDateTimeConverter::class)
abstract class AppDatabase : RoomDatabase() {

    abstract fun notificationsDao(): NotificationsDao

    abstract fun objectTypesDao(): ObjectTypesDao
}

```

```

    abstract fun stationsDao(): StationsDao
}

```

Файл «DatabaseModule.kt»

```

package ru.agattrans.data.db

```

```

import android.app.Application
import androidx.room.Room
import dagger.Module
import dagger.Provides
import javax.inject.Singleton

```

```

@Module
class DatabaseModule {

    @Provides
    @Singleton
    fun provideDatabase(app: Application): AppDatabase =
        Room.databaseBuilder(app, AppDatabase::class.java,
            "agattrans.db")
            .fallbackToDestructiveMigration()
            .build()
}

```

Файл «NotificationsDao.kt»

```

package ru.agattrans.data.db

```

```

import androidx.room.Dao
import androidx.room.Delete
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query
import io.reactivex.Flowable
import ru.agattrans.data.model.Notification

```

```

@Dao
interface NotificationsDao {

    @Query("SELECT * FROM notifications")
    fun getAll(): Flowable<List<Notification>>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun insertAll(notification: List<Notification>)

    @Delete
    fun delete(notification: Notification)

    @Query("DELETE FROM notifications")
    fun deleteAll()
}

```

Файл «ObjectTypesDao.kt»

					ПГУ1.090301.47.001 ПЗ	Лист
						92
Изм.	Лист	№ докум.	Подпись	Дата		

```

package ru.agattrans.data.db

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query
import io.reactivex.Flowable
import ru.agattrans.data.model.ObjectType

@Dao
interface ObjectTypesDao {

    @Query("SELECT * FROM object_types")
    fun getAll(): Flowable<List<ObjectType>>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun insertAll(stations: List<ObjectType>)

    @Query("DELETE FROM object_types")
    fun deleteAll()
}

```

Файл «StationsDao.kt»

```

package ru.agattrans.data.db

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query
import io.reactivex.Flowable
import ru.agattrans.data.model.Station

@Dao
interface StationsDao {

    @Query("SELECT * FROM stations")
    fun getAll(): Flowable<List<Station>>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    fun insertAll(stations: List<Station>)

    @Query("DELETE FROM stations")
    fun deleteAll()
}

```

Файл «AgatTransApi.kt»

```

package ru.agattrans.data.api

import io.reactivex.Completable
import io.reactivex.Single
import retrofit2.http.*
import ru.agattrans.data.model.Notification

```

					ПГУ1.090301.47.001 ПЗ	Лист
						93
Изм.	Лист	№ докум.	Подпись	Дата		

```

import ru.agattrans.data.model.ObjectType

interface AgatTransApi {

    @FormUrlEncoded
    @POST("login")
    fun login(
        @Field("car_number") carNumber: String,
        @Field("password") password: String
    ): Single<TokenResponse>

    @GET("logout")
    fun logout(
        @Query("token") token: String
    ): Completable

    @GET("stations")
    fun stations(
        @Query("token") token: String
    ): Single<StationsResponse>

    @GET("notifications")
    fun notifications(
        @Query("token") token: String
    ): Single<List<Notification>>

    @GET("notification_read")
    fun notificationRead(
        @Query("token") token: String,
        @Query("id") id: Int,
        @Query(value = "date", encoded = true) date: String
    ): Completable

    @GET("has_station_updates")
    fun hasStationsUpdate(
        @Query("token") token: String,
        @Query("timestamp") timestamp: Long
    ): Single<CheckStationsUpdateResponse>

    @GET("objecttype")
    fun objectTypes(
        @Query("token") token: String
    ): Single<List<ObjectType>>

    /*
    @GET("getpoints")
    fun points(
        @Query("token") token: String
    ): Single<StationsResponse>
    */
}

```

Файл «ApiModule.kt»

```
package ru.agattrans.data.api

import android.app.Application
import com.facebook.stetho.okhttp3.StethoInterceptor
import com.squareup.moshi.KotlinJsonAdapterFactory
import com.squareup.moshi.Moshi
import dagger.Module
import dagger.Provides
import io.reactivex.schedulers.Schedulers
import okhttp3.Cache
import okhttp3.OkHttpClient
import okhttp3.logging.HttpLoggingInterceptor
import okhttp3.logging.HttpLoggingInterceptor.Level
import okhttp3.logging.HttpLoggingInterceptor.Logger
import retrofit2.Retrofit
import retrofit2.adapter.rxjava2.RxJava2CallAdapterFactory
import retrofit2.converter.moshi.MoshiConverterFactory
import ru.agattrans.BuildConfig
import ru.agattrans.util.moshi.OffsetDateTimeAdapter
import timber.log.Timber
import java.io.File
import javax.inject.Singleton

@Module
class ApiModule {

    @Provides
    @Singleton
    fun provideAgatTransApi(client: OkHttpClient, moshi: Moshi):
    AgatTransApi =
        Retrofit.Builder()
            .baseUrl("http://85.234.35.203:49100/api/")
            .client(client)
            .validateEagerly(BuildConfig.DEBUG) // Fail early:
check Retrofit configuration at creation time

            .addCallAdapterFactory(RxJava2CallAdapterFactory.createWithScheduler(Schedulers.io()))

            .addConverterFactory(MoshiConverterFactory.create(moshi))
                .build()
                .create(AgatTransApi::class.java)

    @Provides
    @Singleton
    fun provideOkHttpClient(app: Application): OkHttpClient =
        OkHttpClient.Builder()
            .cache(httpCache(app))
            .apply {
                if (BuildConfig.DEBUG) {
```

```

        addInterceptor(httpLoggingInterceptor())
        addNetworkInterceptor(StethoInterceptor())
    }
}
.build()

@Provides
@Singleton
fun provideMoshi(): Moshi =
    Moshi.Builder()
        .add(OffsetDateTimeAdapter)
        .add(KotlinJsonAdapterFactory())
        .build()

private fun httpCache(app: Application): Cache {
    val cacheDir = File(app.cacheDir, "http")
    return Cache(cacheDir, DISK_CACHE_SIZE)
}

private fun httpLoggingInterceptor(): HttpLoggingInterceptor
=
    HttpLoggingInterceptor(object : Logger {
        override fun log(message: String) {
            Timber.tag("OkHttp").d(message)
        }
    }).apply { level = Level.BASIC }

private companion object {
    private const val DISK_CACHE_SIZE = 10485760L // 10MB
}

```

Файл «CheckStationsUpdate.kt»

```

package ru.agattrans.data.api

import com.squareup.moshi.Json

data class CheckStationsUpdateResponse(
    @Json(name = "has_updates") val hasUpdates: Boolean
)

```

Файл «StationsResponse.kt»

```

package ru.agattrans.data.api

import com.squareup.moshi.Json
import ru.agattrans.data.model.Station

data class StationsResponse(
    @Json(name = "stations") val stations: List<Station>,
    @Json(name = "timestamp") val timestamp: Long
)

```

Файл «TokenResponse.kt»

					ПГУ1.090301.47.001 ПЗ	Лист
						96
Изм.	Лист	№ докум.	Подпись	Дата		

```
package ru.agattrans.data.api

import com.squareup.moshi.Json

data class TokenResponse(
    @Json(name = "token") val token: String
)
```

Файл «OffsetDateTimeAdapter.kt»

```
package ru.agattrans.util.moshi

import com.squareup.moshi.FromJson
import com.squareup.moshi.ToJson
import org.threeten.bp.OffsetDateTime
import org.threeten.bp.format.DateTimeFormatter

object OffsetDateTimeAdapter {
    private val formatter = DateTimeFormatter.ISO_OFFSET_DATE_TIME

    @ToJson
    fun toJson(offsetDateTime: OffsetDateTime): String =
        offsetDateTime.format(formatter)

    @FromJson
    fun fromJson(value: String): OffsetDateTime =
        OffsetDateTime.parse(value, formatter)
}
```

Файл «OffsetDateTimeConverter.kt»

```
package ru.agattrans.util.room

import androidx.room.TypeConverter
import org.threeten.bp.OffsetDateTime
import org.threeten.bp.format.DateTimeFormatter

object OffsetDateTimeConverter {
    private val formatter = DateTimeFormatter.ISO_OFFSET_DATE_TIME

    @JvmStatic
    @TypeConverter
    fun toOffsetDateTime(value: String): OffsetDateTime =
        OffsetDateTime.parse(value, formatter)

    @JvmStatic
    @TypeConverter
    fun fromOffsetDateTime(offsetDateTime: OffsetDateTime): String =
        offsetDateTime.format(formatter)
}
```

Файл «CrashlyticsTree.kt»

					ПГУ1.090301.47.001 ПЗ	Лист
						97
Изм.	Лист	№ докум.	Подпись	Дата		

```

package ru.agattrans.util

import android.util.Log
import com.crashlytics.android.Crashlytics
import timber.log.Timber.DebugTree

class CrashlyticsTree : DebugTree() {

    override fun isLoggable(tag: String?, priority: Int): Boolean
    {
        return priority >= Log.INFO
    }

    override fun log(priority: Int, tag: String?, message: String,
t: Throwable?) {
        Crashlytics.log(priority, tag, message)
        t?.let { if (priority == Log.ERROR)
Crashlytics.logException(it) }
    }
}

```

Файл «CustomSnackbar.kt»

```

package ru.agattrans.util

import android.annotation.SuppressLint
import androidx.annotation.LayoutRes
import androidx.coordinatorlayout.widget.CoordinatorLayout
import com.google.android.material.snackbar.Snackbar
import android.view.LayoutInflater
import android.view.View
import android.view.ViewTreeObserver
import android.widget.TextView

class CustomSnackbar(
    parentView: View,
    @LayoutRes layout: Int,
    duration: Int,
    background: Int = -1,
    swipeToDismiss: Boolean = true
) {
    private val snackbar: Snackbar = Snackbar.make(parentView, "",
duration)
    val contentView: View
    private var snackbarView: Snackbar.SnackbarLayout

    init {
        @SuppressLint("RestrictedApi")
        snackbarView = snackbar.view as Snackbar.SnackbarLayout

        if (background != -1) {
            snackbarView.setBackgroundResource(background)
        }
    }
}

```

```

        if (!swipeToDismiss) {
            snackbarView.viewTreeObserver.addOnPreDrawListener(object
: ViewTreeObserver.OnPreDrawListener {
                override fun onPreDraw(): Boolean {

snackbarView.viewTreeObserver.removeOnPreDrawListener(this)
                    (snackbarView.layoutParams as
CoordinatorLayout.LayoutParams).behavior = null
                    return true
                }
            })
        }
        snackbarView.setPadding(0, 0, 0, 0)

snackbarView.findViewById<TextView>(ru.agattrans.R.id.snackbar_t
ext).visibility = View.INVISIBLE

snackbarView.findViewById<TextView>(ru.agattrans.R.id.snackbar_a
ction).visibility = View.INVISIBLE
        contentView =
LayoutInflater.from(parentView.context).inflate(layout, null)
        snackbarView.addView(contentView, 0)
    }

    fun show() = snackbar.show()

    fun isShown() = snackbar.isShown

    fun dismiss() = snackbar.dismiss()
}

```

Файл «EmptyRecyclerView.kt»

```

package ru.agattrans.util

import android.content.Context
import android.util.AttributeSet
import android.view.View
import androidx.recyclerview.widget.RecyclerView

class EmptyRecyclerView(context: Context, attrs: AttributeSet) :
RecyclerView(context, attrs) {
    private var emptyView: View? = null
    private val emptyObserver = object : AdapterDataObserver() {
        override fun onChanged() {
            super.onChanged()
            setUpVisibility()
        }
    }

    override fun onItemRangeInserted(positionStart: Int,
itemCount: Int) {
        super.onItemRangeInserted(positionStart, itemCount)
        setUpVisibility()
    }
}

```

```

    }

    override fun onItemRangeRemoved(positionStart: Int,
itemCount: Int) {
        super.onItemRangeRemoved(positionStart, itemCount)
        setUpVisibility()
    }
}

fun setEmptyView(emptyView: View?) {
    this.emptyView = emptyView
}

override fun setAdapter(adapter: Adapter<*>?) {
    getAdapter()?.unregisterAdapterDataObserver(emptyObserver)
    adapter?.registerAdapterDataObserver(emptyObserver)
    super.setAdapter(adapter)
    setUpVisibility()
}

private fun setUpVisibility() {
    if (adapter != null && emptyView != null) {
        val showEmptyView = adapter!!.itemCount == 0
        emptyView!!.visibility = if (showEmptyView) View.VISIBLE
else View.GONE
        visibility = if (showEmptyView) View.GONE else
View.VISIBLE
    }
}
}

```

Файл «KExtentions.kt»

```

package ru.agattrans.util

import android.app.Activity
import android.content.Context
import android.graphics.Bitmap
import android.graphics.Canvas
import android.util.DisplayMetrics
import androidx.annotation.DrawableRes
import com.google.android.material.snackbar.Snackbar
import androidx.fragment.app.Fragment
import androidx.core.content.ContextCompat
import android.view.View
import android.view.inputmethod.InputMethodManager
import android.widget.EditText
import com.jakewharton.rxbinding2.view.clicks
import com.redmadrobot.inputmask.MaskedTextChangedListener
import io.reactivex.Observable
import io.reactivex.schedulers.Schedulers
//import org.jetbrains.anko.displayMetrics
import ru.agattrans.ui.MainActivity

```

					ПГУ1.090301.47.001 ПЗ	Лист
						100
Изм.	Лист	№ докум.	Подпись	Дата		

```

import java.util.concurrent.TimeUnit

private const val SINGLE_CLICKS_THROTTLE_DURATION = 300L

fun View.singleClicks(): Observable<Unit> =
    clicks().throttleFirst(SINGLE_CLICKS_THROTTLE_DURATION,
        TimeUnit.MILLISECONDS, Schedulers.computation())

fun EditText.applyInputMask(inputMask: String) {
    val maskedTextChangedListener =
        MaskedTextChangedListener(inputMask, true, this, null, null)
    addTextChangedListener(maskedTextChangedListener)
    onFocusChangeListener = maskedTextChangedListener
}

fun Activity.hideKeyboard() {
    val inputMethodManager =
        getSystemService(Activity.INPUT_METHOD_SERVICE) as
        InputMethodManager
    val view = currentFocus ?: View(this)
    inputMethodManager.hideSoftInputFromWindow(view.windowToken,
        0)
}

val Fragment.mainActivity: MainActivity?
    get() = activity as MainActivity?

fun permanentSnackbar(view: View, message: Int, actionText: Int,
    action: (View) -> Unit) = Snackbar
    .make(view, message, Snackbar.LENGTH_INDEFINITE)
    .setAction(actionText, action)
    .show()

fun View.show() = apply { visibility = View.VISIBLE }
fun View.gone() = apply { visibility = View.GONE }
fun View.invisible() = apply { visibility = View.INVISIBLE }

var View.visible: Boolean
    get() = visibility == View.VISIBLE
    set(value) {
        visibility = if (value) View.VISIBLE else View.GONE
    }

fun Context.bitmapFromVectorDrawable(@DrawableRes drawableId:
    Int): Bitmap {
    val drawable = checkNotNull(ContextCompat.getDrawable(this,
        drawableId))
    val bitmap = Bitmap.createBitmap(drawable.intrinsicWidth,
        drawable.intrinsicHeight, Bitmap.Config.ARGB_8888)
    val canvas = Canvas(bitmap)
    drawable.setBounds(0, 0, canvas.width, canvas.height)
    drawable.draw(canvas)
}

```

```

    return bitmap
}

```

```

fun Context.dpToPx(dp: Int): Long =
    Math.round(dp * DisplayMetrics().density).toLong()

```

Файл «activity_main.xml»

```

<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/contentFrame"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:keepScreenOn="true"
    tools:context="ru.agattrans.ui.MainActivity"/>

```

Файл «fragment_login.xml»

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    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="match_parent">

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="@dimen/content_max_width"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:gravity="center_horizontal"
        android:padding="20dp">

        <ImageView
            android:id="@+id/logoView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:contentDescription="@null"
            android:src="@drawable/img_logo"/>

        <com.google.android.material.textfield.TextInputLayout
            android:id="@+id/carNumberLayout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="24dp">

            <com.google.android.material.textfield.TextInputEditText
                android:id="@+id/carNumberEdit"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="@string/car_number"
                android:inputType="textVisiblePassword"/>

```

```

</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/passwordLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    app:passwordToggleEnabled="true">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/passwordEdit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/password"
        android:inputType="textPassword"/>

</com.google.android.material.textfield.TextInputLayout>

<Button
    android:id="@+id/signInButton"
    style="@style/Widget.AppCompat.Button.Colored"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:text="@string/sign_in"/>

</LinearLayout>

```

</ScrollView>

Файл «fragment_map.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:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="?android:windowBackground">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        style="@style/Widget.Toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:title="@string/app_name"/>

    <androidx.coordinatorlayout.widget.CoordinatorLayout
        android:id="@+id/coordinatorLayout"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

```

```

<com.yandex.mapkit.mapview.MapView
    android:id="@+id/mapView"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

```

```

<com.google.android.material.floatingactionbutton.FloatingAction
Button
    android:id="@+id/myDelRouteBtn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom|end"
    android:layout_marginBottom="100dp"
    android:layout_marginEnd="16dp"
    android:src="@drawable/ic_directions_off_24"
    app:backgroundTint="#FFFFFF"
    app:fabSize="normal"/>

```

```

<com.google.android.material.floatingactionbutton.FloatingAction
Button
    android:id="@+id/myLocationButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom|end"
    android:layout_marginBottom="16dp"
    android:layout_marginEnd="16dp"
    android:src="@drawable/ic_fab_my_location"
    app:backgroundTint="#FFFFFF"
    app:fabSize="normal"/>
<!-- app:useCompatPadding="true" -->

<androidx.cardview.widget.CardView
    android:id="@+id/directionsFormLayout"
    android:layout_width="@dimen/content_max_width"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:layout_margin="8dp"
    android:visibility="gone">

```

```

    <include layout="@layout/view_directions"/>

```

```

</androidx.cardview.widget.CardView>

```

```

</androidx.coordinatorlayout.widget.CoordinatorLayout>

```

```

</LinearLayout>

```

Файл «fragment_notifications.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"
xmlns:tools="http://schemas.android.com/tools"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="?android:windowBackground">

<androidx.appcompat.widget.Toolbar
    android:id="@+id/toolbar"
    style="@style/Widget.Toolbar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:navigationIcon="@drawable/ic_toolbar_back"
    app:title="@string/notifications"/>

<FrameLayout
    android:layout_width="@dimen/content_max_width"
    android:layout_height="match_parent"
    android:layout_gravity="center_horizontal">

    <ru.agattrans.util.EmptyRecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:clipToPadding="false"
        android:paddingBottom="8dp"
        android:paddingTop="8dp"
        tools:listitem="@layout/item_notification"/>

    <TextView
        android:id="@+id/emptyView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="@string/no_notifications"

        android:textAppearance="@style/TextAppearance.AppCompat.Subhead"
        android:textColor="?android:textColorSecondary"/>

</FrameLayout>

</LinearLayout>

```

Файл «search_address_layout.xml»

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout

    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="wrap_content"

```

					ПГУ1.090301.47.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		105

```

        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintRight_toRightOf="parent">
<AutoCompleteTextView
    android:id="@+id/placeView"
    android:layout_width="match_parent"
    android:layout_height="48dp"
    android:layout_gravity="center"
    android:layout_marginStart="25dp"
    android:background="?selectableItemBackground"
    android:gravity="center_vertical"
    android:dropDownVerticalOffset="15dp"
    android:dropDownWidth="wrap_content"
    android:textSize="18sp"
    android:inputType="text"/>
</androidx.coordinatorlayout.widget.CoordinatorLayout>

```

Файл «view_directions.xml»

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="16dp">

    <ImageView
        android:id="@+id/placeFromImage"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:src="@drawable/ic_place_from_24dp"
        android:layout_marginTop="10dp"

app:layout_constraintTop_toTopOf="@+id/placeFromDivider"
        tools:ignore="MissingConstraints"/>

    <ru.agattrans.ui.map.SearchLayout
        android:id="@+id/placeFromDivider"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        tools:ignore="MissingConstraints"/>

    <ImageView
        android:id="@+id/placeToImage"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:src="@drawable/ic_place_to_24dp"
        android:layout_marginTop="10dp"

app:layout_constraintTop_toBottomOf="@id/placeFromImage"

```

```

app:layout_constraintTop_toTopOf="@+id/placeToDivider"
    tools:ignore="MissingConstraints"/>

    <ru.agattrans.ui.map.SearchLayout
        android:id="@+id/placeToDivider"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"

app:layout_constraintTop_toBottomOf="@id/placeFromDivider"
    tools:ignore="MissingConstraints"/>

    <Button
        android:id="@+id/directionsButton"
        style="@style/Widget.AppCompat.Button.Colored"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:paddingEnd="24dp"
        android:paddingStart="24dp"
        android:text="@string/button_directions"
        app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintTop_toBottomOf="@id/placeToDivider"/>

</androidx.constraintlayout.widget.ConstraintLayout>
Файл «view_stations_info.xml»

<?xml version="1.0" encoding="utf-8"?>
<ru.agattrans.ui.map.StationInfoView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="@dimen/content_max_width"
        android:layout_height="match_parent"
        android:paddingBottom="12dp"
        android:paddingEnd="16dp"
        android:paddingStart="16dp"
        android:paddingTop="12dp">

        <TextView
            android:id="@+id/titleView"
            android:layout_width="0dp"
            android:layout_height="wrap_content"

            android:textAppearance="@style/TextAppearance.AppCompat.Subhead"

```

```

app:layout_constraintEnd_toStartOf="@id/directionButton"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    tools:text="Заправка лукойл"/>

<TextView
    android:id="@+id/addressView"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="2dp"

    android:textAppearance="@style/TextAppearance.AppCompat.Body1"
    android:textColor="?android:textColorSecondary"

    app:layout_constraintEnd_toStartOf="@id/directionButton"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/titleView"
    tools:text="Нижегородская область, г. Лукоянов"/>

<TextView
    android:id="@+id/priorityView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="2dp"

    android:textAppearance="@style/TextAppearance.AppCompat.Body2"
    android:textColor="#1B5E20"

    app:layout_constraintEnd_toStartOf="@id/directionButton"
    app:layout_constraintHorizontal_bias="0"
    app:layout_constraintStart_toStartOf="parent"

    app:layout_constraintTop_toBottomOf="@id/addressView"
    tools:text="@string/priority_high"/>

<TextView
    android:id="@+id/limitView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_toEndOf="@id/priorityView"
    android:singleLine="true"

    android:textAppearance="@style/TextAppearance.AppCompat.Body1"
    android:textColor="?android:textColorSecondary"

    app:layout_constraintBaseline_toBaselineOf="@id/priorityView"

    app:layout_constraintEnd_toStartOf="@id/directionButton"
    app:layout_constraintHorizontal_bias="0"

    app:layout_constraintStart_toEndOf="@id/priorityView"

```

```

        tools:text="до 10л"/>

<TextView
    android:id="@+id/commentView"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="2dp"

    android:textAppearance="@style/TextAppearance.AppCompat.Body1"
    android:textColor="?android:textColorSecondary"

    app:layout_constraintEnd_toStartOf="@id/directionButton"
    app:layout_constraintStart_toStartOf="parent"

    app:layout_constraintTop_toBottomOf="@id/priorityView"
    tools:text="Комментарий"/>

<Button
    android:id="@+id/directionButton"
    style="@style/Widget.AppCompat.Button.Colored"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:drawablePadding="8dp"

    android:drawableStart="@drawable/ic_directions_white_24dp"
    android:text="@string/button_directions"
    tools:ignore="MissingConstraints"
    app:layout_constraintEnd_toEndOf="parent"/>

<Button
    android:id="@+id/mapsButton"

    style="@style/Widget.AppCompat.Button.Borderless.Colored"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:drawablePadding="8dp"
    android:drawableStart="@drawable/ic_map_accent_24dp"
    android:text="@string/button_maps"
    app:layout_constraintEnd_toEndOf="parent"

    app:layout_constraintTop_toBottomOf="@id/directionButton"/>

</androidx.constraintlayout.widget.ConstraintLayout>

</ru.agattrans.ui.map.StationInfoView>

```

ПриложениеВ

(Обязательно)

Картографический сервис поиска АЗС

Презентация

					ПГУ1.090301.47.001 ПЗ	Лист
						110
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

Цель работы – разработка мобильного приложения на Kotlin.

Задачи:

- Запуск в режиме полной совместимости на Android
- Создание интерфейса
- Адаптация управления под различные типы операционных систем
- Создание системы случайной генерации NPC

Объект исследования — разработка мобильного приложения

Предмет исследования — готовое приложение разработанное на Kotlin

Актуальность проекта

Приложение разрабатывалось для транспортных компаний с целью оптимизации затрат на топливо:

- Разница в цене и качестве топлива на АЗС даже в пределах одной сети
- Договоры с разными АЗС помогают компаниям экономить
- Мобильное приложение для Android (70% рынка) показывает ближайшие АЗС с учетом:
 - акций и скидок
 - корпоративных условий
 - данных геолокации

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.47.001 ПЗ

Лист

111

Обзор аналогов

Сервис Т-Банка «Топливо»



«Яндекс.Заправки»



Выбор инструментов разработки



Yandex
MapKit

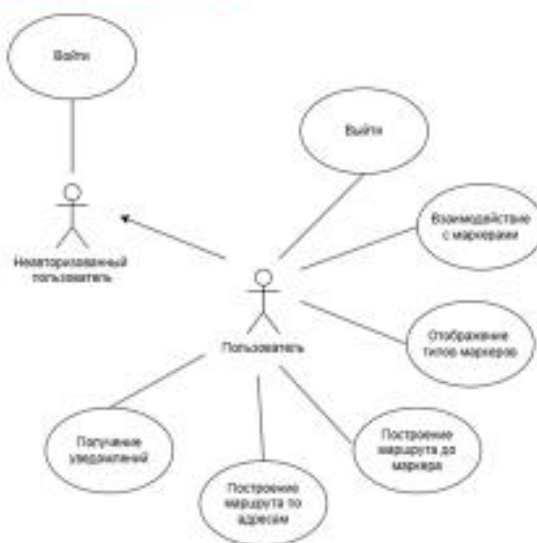
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.47.001 ПЗ

Лист

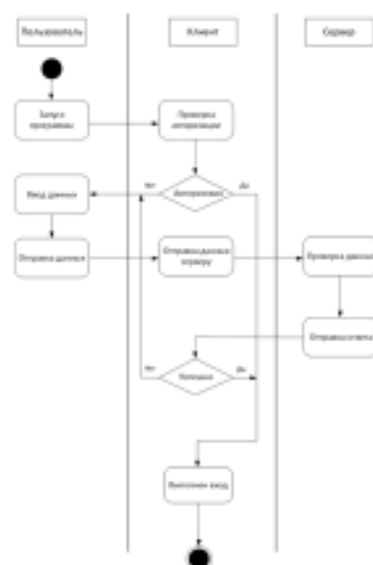
112

UML диаграмма прецедентов



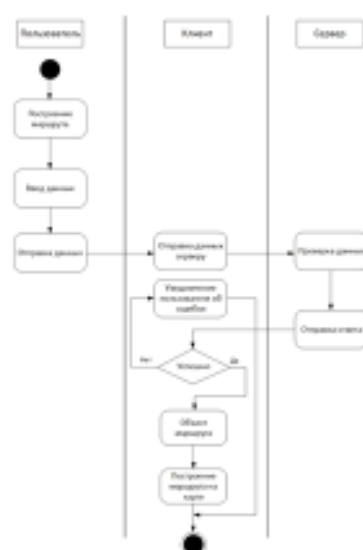
UML диаграмма деятельности авторизации

- Для авторизации пользователь вводит государственный номер транспортного средства и пароль.
- Затем, после нажатия на кнопку «Войти», вызывается метод авторизации, где из представлений для ввода берутся данные профиля.
- Далее, делается запрос на сервер, при успешной авторизации, открывается основной фрагмент, иначе пользователя информируют об ошибке авторизации.



UML диаграмма деятельности маршрутизации

- Сначала, приложение получает данные о местоположении пользователя путем запроса к серверу через менеджера геолокации.
- Затем, приходит ответ в виде объекта с данными о геолокации, после чего создается экземпляр класса, содержащий информацию о маршруте.
- Полученные данные о маршруте отображаются на карте.



Приложение «Картографический сервис поиска АЗС»



Авторизация пользователя



Информация о станции



Построение маршрута