

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего
образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

Кафедра

вычислительной техники

«Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация автоматизированных
систем специального назначения»
Специализация: «Эксплуатация вычислительных машин, комплексных
систем и сетей специального назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

ПРОГРАММА ОПТИМИЗАЦИИ ЛОГИСТИКИ
ПЕРЕДВИЖЕНИЯ

Студент

Ник

11.06.2024

Никитин Даниил Дмитриевич.

(подпись, дата)

(ФИО полностью)

Руководитель

[Подпись]

11.06.2024

Федюнин Р.Н.

(подпись, дата)

(фамилия, инициалы)

Нормоконтролёр

[Подпись]

11.06.2024

Бычков А.С.

(подпись, дата)

(фамилия, инициалы)

Рецензент

Г.А. Селезнёв
АО НПП "Робит"

Васильев

Васильев А.В.

(должность, место работы)

(подпись, дата)

(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 27.05.2024 г. № 11)

Заведующий кафедрой

[Подпись]

Митрохин М.А.

(подпись)

(фамилия, инициалы)

Работа защищена с отметкой отлично

(протокол заседания ГЭК от 19.06.2024 № 1)

Секретарь ГЭК

Е

(подпись)

Гурин Е.И.

(фамилия, инициалы)

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования

Пензенский государственный университет

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

Вычислительная техника

«Утверждаю»

Заведующий кафедрой

Митрохин М.А.

«19» апреля 2024 г.

ЗАДАНИЕ НА ДИПЛОМНЫЙ ПРОЕКТ

1. Студент Никитин Даниил Дмитриевич гр. 19ВОЭ1 факультета
вычислительной техники.

2. Тема работы: «Программа оптимизации логистики передвижения».

Тема утверждена приказом ПГУ № 405/о от 19.04.2024 г.

3. Руководитель работы к.т.н., доцент каф. ВТ «ПГУ» Федюнин Р.Н.

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

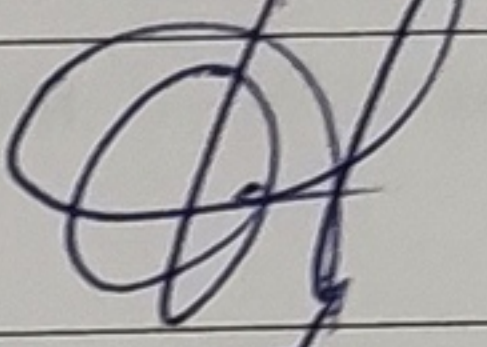
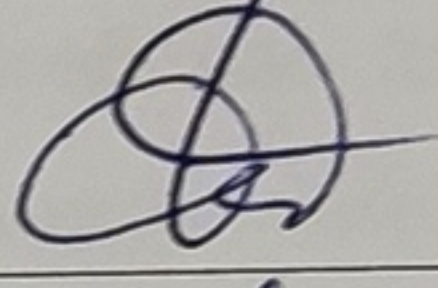
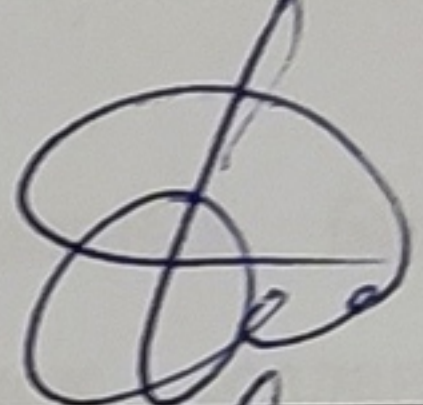
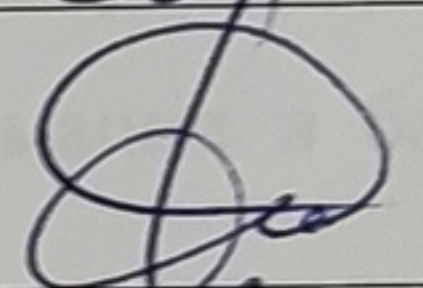
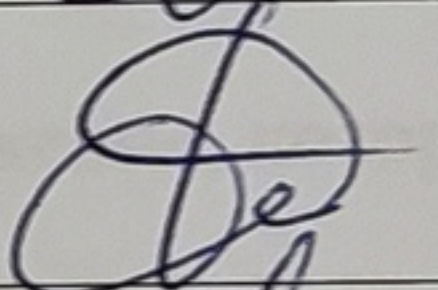
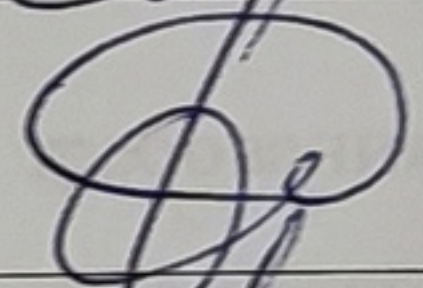
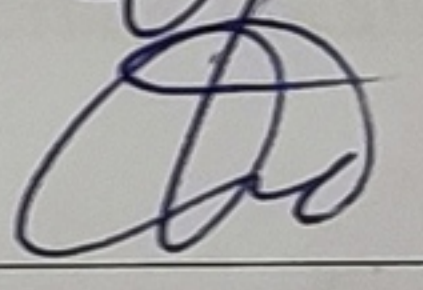
Назначение разработки – Программа предназначена для оптимизации логистики передвижения на примере поиска попутчиков для междугородних поездок. В программе необходимо предусмотреть возможность регистрации и авторизации пользователя. Каждый из пользователей должен иметь возможность создать свою поездку, присоединиться к поездкам других пользователей и просматривать построенный маршрут поездки.

Программа должна быть разработана с помощью среды Android Studio с помощью инструментов, имеющих лицензию свободного распространения.

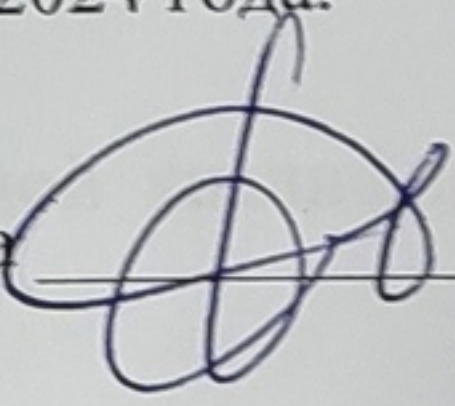
5. Перечень подлежащих разработке вопросов.

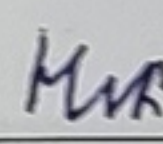
Требуется произвести анализ предметной области, определить функциональный состав программы, выбрать инструменты реализации программы и развертывания приложения в облачной среде исполнения. Необходимо разработать алгоритм работы программы, разработать UML диаграммы с последующим написанием программного кода.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10	19.04.2024 – 26.04.2024	
2.	Выбор инструментов для разработки программы	10	26.04.2024 – 3.05.2024	
3.	Проектирование и составление UML диаграмм для реализации программы	20	3.05.2024 - 10.05.2024	
4.	Реализация базы данных	10	10.05.2024 - 17.05.2024	
5.	Реализация программы	10	17.05.2024 - 24.05.2024	
6.	Развертывание приложения в облачной среде и отладка	10	24.05.2024 - 31.05.2024	
7.	Оформление выпускной работы	30	31.05.2024 - 05.06.2024	

Дата выдачи задания 19 «апреля» 2024 года.

Руководитель дипломного проекта  Федюнин Р.Н.
(подпись, дата)

Задание к исполнению принял студент  Никитин Д.Д.
(подпись, дата)

Работу к защите допустить
Декан факультета  Фионова Л.Р.
17.06.2024

Реферат

Работа содержит 112 листов, 33 рисунков, 4 таблицы, 6 источника, 3 приложения

ПРИЛОЖЕНИЕ, ANDROID, MAPKIT SDK, FIREBASE, КАРТЫ, МАРШРУТ, JAVA, ANDROID STUDIO, ЯНДЕКС

Цель ВКР – разработка приложения для оптимизации логистики передвижения на примере поиска попутчика для междугородних поездок. Построение междугороднего маршрута с помощью Yandex Mapkit SDK. Для авторизации и хранения информации пользователя и информации о поездках используется Firebase.

В работе произведен анализ и подбор инструментов разработки для реализации приложения. Выполнено проектирование, в результате которого были разработаны *UML* диаграммы разрабатываемого ПО.

Разработаны макеты окон приложения, структура базы данных. Были разработаны функции авторизации, регистрации, поиска поездок, создания поездок, построения поездок, внесения всей необходимой информации в базу данных и вывод необходимой информации из неё.

В ходе выполнения ВКР было разработано мобильное приложение, которое удовлетворяет поставленной задаче.

В разделе безопасности жизнедеятельности были описаны требования к офисным помещениям и рабочему месту для минимизации воздействия вредных факторов на организм человека.

В экономическом разделе был произведен расчет затрат на разработку программного обеспечения.

Графическая часть выпускной квалификационной работы содержит *UML*-диаграммы.

					ПГУ 1.090501.06.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Программа оптимизации логистики передвижения Пояснительная записка	Лит.	Лист	Листов
Разраб.		Никитин Д.Д.		11.06.24				
Провер.		Федюнин Р.Н.		11.06.24			2	112
Н. контр.		Бычков А.С.		11.06.24		ФВТ, каф. ВТ гр.19В031		
Утв.		Митрохин М.А.		11.06.24				

Содержание

Перечень сокращений и условных обозначений.....	5
Введение.....	6
1 Анализ предметной области.....	8
1.1 Анализ темы.....	8
1.2 Анализ аналогов.....	8
1.3 Постановка задачи.....	13
2 Архитектура приложения.....	16
2.1 Выбор архитектуры приложения.....	16
2.2 Разработка UML-диаграммы.....	17
3 Выбор инструментов разработки.....	23
3.1 Выбор языка программирования.....	23
3.2 Выбор сервиса для отображения карт.....	24
3.3 Выбор среды программирования.....	25
3.4 Выбор сервиса облачной СУБД.....	26
4 Разработка приложения.....	27
4.1 Подключение карт.....	27
4.2 Построение маршрута.....	27
4.3 Подключение БД.....	29
4.4 Авторизация и регистрация пользователя.....	30
4.5 Страницы пользователя.....	35
4.6 Создание поездки.....	42
4.7 Бронирование поездки.....	47
5 Технико-экономическое обоснование.....	51
5.1 Введение.....	51
5.2 Капитальные вложения.....	51
5.3 Затраты на разработку программы.....	53
5.4 Заработная плата.....	53
5.5 Итоговые расчеты.....	54

6 Безопасность жизнедеятельности.....	55
6.1 Характеристика опасных факторов при работе с ПЭВМ.....	55
6.2 Рабочее помещение.....	58
6.3 Санитарно-гигиеническая характеристика.....	58
6.4 Электробезопасность.....	60
Заключение.....	62
Список используемых источников.....	63
Приложение А – UML диаграммы.....	64
Приложение Б – Листинг программы.....	70
Приложение В – Презентация.....	107

Перечень сокращений и обозначений

В данном проекте применяются следующие сокращения и обозначения

БД - база данных

ОС - операционная система

ПО - программное обеспечение

ПК - персональный компьютер

СанПин - санитарно-эпидемические нормы

СУБД - система управления базами данных

API — программный интерфейс, то есть описание способов взаимодействия одной компьютерной программы с другими

IDE (Integrated Development Environment) - интегрированная среда разработки

JSON (JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript

NoSQL - обозначение широкого класса разнородных систем управления базами данных

SDK - набор инструментов для разработки программного обеспечения, объединённый в одном пакете

UML (Unified Modeling Language) - унифицированный язык моделирования, язык графического описания для объектного моделирования

UID (user identifier) - идентификаторами пользователя

Введение

Логистика передвижения [1] представляет собой различные организационные процессов для обеспечения передвижения индивида или группы лиц по оптимальному маршруту из одной точки в другую. Построение пути и организация перемещения для одного человека не является сложной задачей, проблема возникает в первую очередь при наличие большой группы людей. Данную проблему встречается в различных сферах человеческой деятельности в том числе и специализированных. В рамках данной выпускной квалификационной работы данная проблема будет рассмотрена на примере поиска попутчиков для организации междугородних поездок.

Поиск попутчиков для междугородних поездок является сложной задачей. Как и большинство проблем в современном мире проблемы подобного рода можно решить с помощью мобильного приложения. Мобильные приложения легко распространяются, что приведет к наличию большого количества пользователей, что в свою очередь приведет к разнообразию создаваемых поездок и упрощению поиска попутчиков.

Любой из пользователей приложения может занять и свободно менять роль, которую они играют в организации поездки. Первая роль — это люди без транспортного средства, которых можно назвать попутчики. Вторая роль — это люди, у которых есть транспортное средство, которых можно назвать водители. Попутчики ищут водителей, которые отвезут их в необходимое место за определенную плату, а водители ищут попутчиков для уменьшения затраченных средств на прохождение пути.

Самое сложное при поиске попутчика, это найти водителей, которые начинают свой путь из того же города что и вы, отправляется в город, в который нужно вам и будет делать в это в необходимый вам промежуток времени. Данный процесс можно упростить с помощью мобильных приложений и интернет-форумов, при условии наличия необходимого функционала. Приложение должно обладать системой поиска поездок, которые удовлетворяют условиям попутчика.

Так же приложение должно в удобной форме помочь водителям создать свою поездку для более быстрого поиска попутчиков. Самое главное, чтобы приложение предоставляло всю необходимую пользователю информацию в удобной и простой форме.

					<i>ПГУ 1.090501.06.001 ПЗ</i>	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

1 Анализ предметной области

1.1 Анализ темы

В современных условиях вопрос перемещения очень важен. Для перемещения по городу существует общественный транспорт и такси, которые являются удобными и бюджетными решениями проблемы перемещения. Но у междугородних поездок нет такого очевидного решения. Конечно, можно использовать ранее указанное такси, но такое решение нельзя назвать бюджетным. Выгоднее всего перемещаться между городами в компании, но найти себе попутчиков может быть довольно тяжело. Для этого лучше всего использовать мобильные приложения. С помощью мобильного приложения можно быстро найти себе попутчиков.

Разрабатываемое приложение поможет найти попутчиков в соответствии с указанными требованиями, что будет полезно как пассажирам, так и водителям.

1.2 Анализ аналогов

Перед разработкой приложения был проведен сравнительный анализ аналогичных приложений в Google Play и интернет-форумов, которые могут помочь пользователям в организации совместных поездок. Был проведен анализ следующих приложений: «BlaBlaCar», «Едем.рф», и следующих интернет-форумов: «Форум Винского», «Форум Попутчик.ру».

Приложение «BlaBlaCar» представляет пользователю возможность поиска и создание совместной поездки. Для нахождения попутчиков необходимо указать место отправки, пункт назначения, дату поездки и количество человек, затем будут представлены доступные поездки, а также их цены и время выезда. При создании поездки необходимо указать начало и конец поездки, дату отправки и количество свободных мест. Интерфейс приложения приложен на рисунке 1.1.

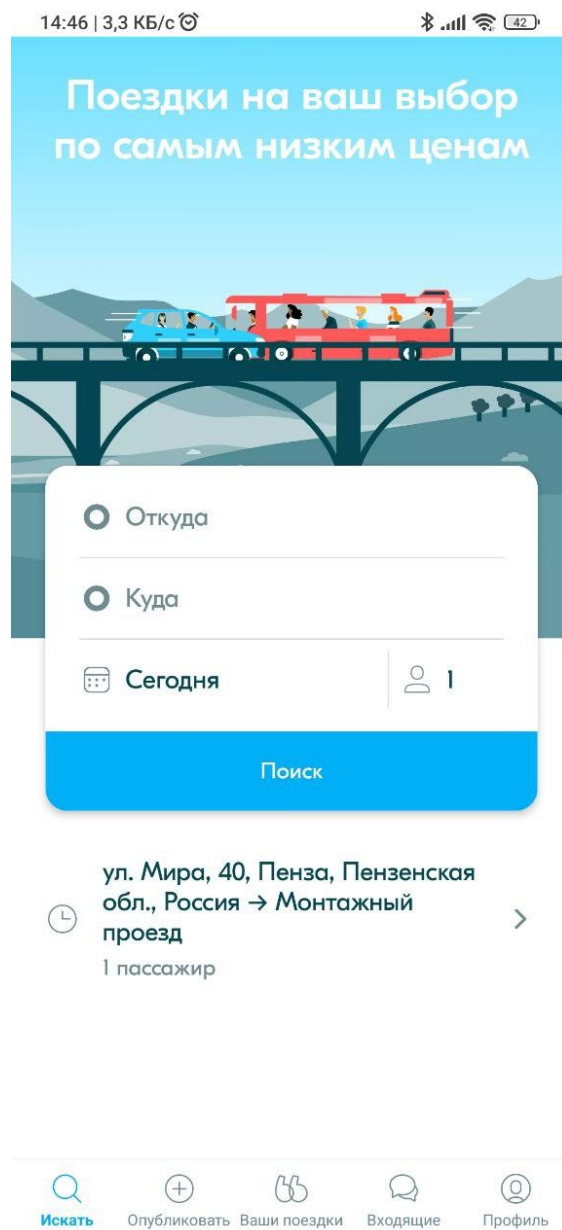


Рисунок 1.1 – Приложение «BlaBlaCar»

Приложение «Едем.рф» представляет пользователю возможность поиска и создание совместной поездки. Для начала поиска пользователю необходимо указать всю нужную информацию. При создании собственной поездки после указания места отправки и пункта назначения и ввести информацию об автомобиле. Интерфейс приложения приставлен на рисунке 1.2.

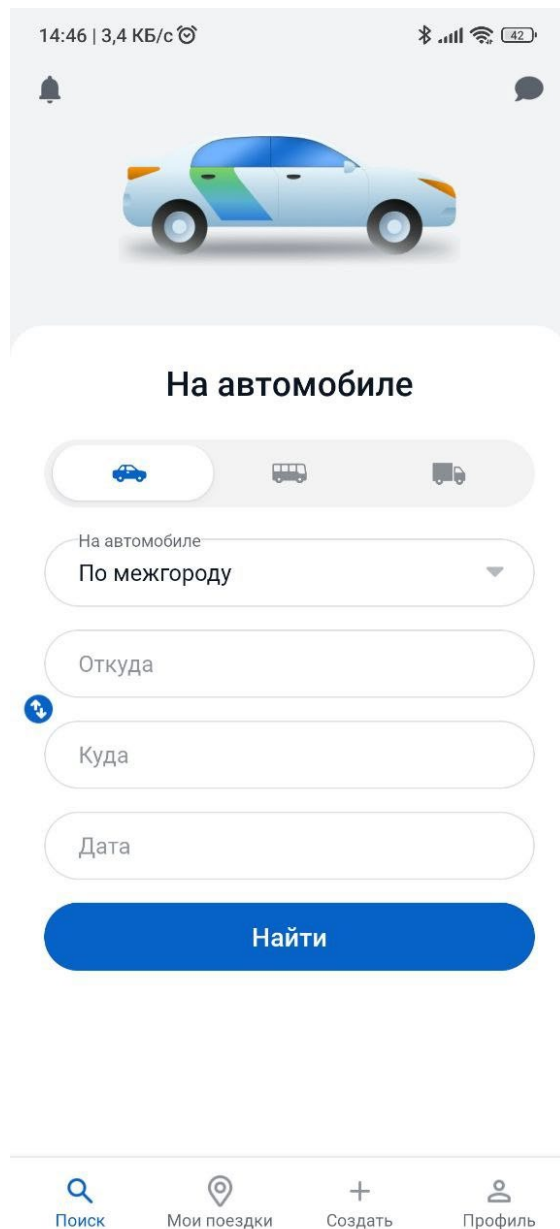


Рисунок 1.2 – Приложение «Едем.рф»

Форум «Форум Винского» представляет пользователю возможность создать обсуждения в интересующей его категории для поиска попутчика. Для создания темы достаточно придумать заголовок и описание. После этого другие пользователи смогут оставить свои комментарии с вопросами или предложениями. Пользователь может осуществить поиск интересующих его тем по ключевым словам. Интерфейс форума приставлен на рисунке 1.3

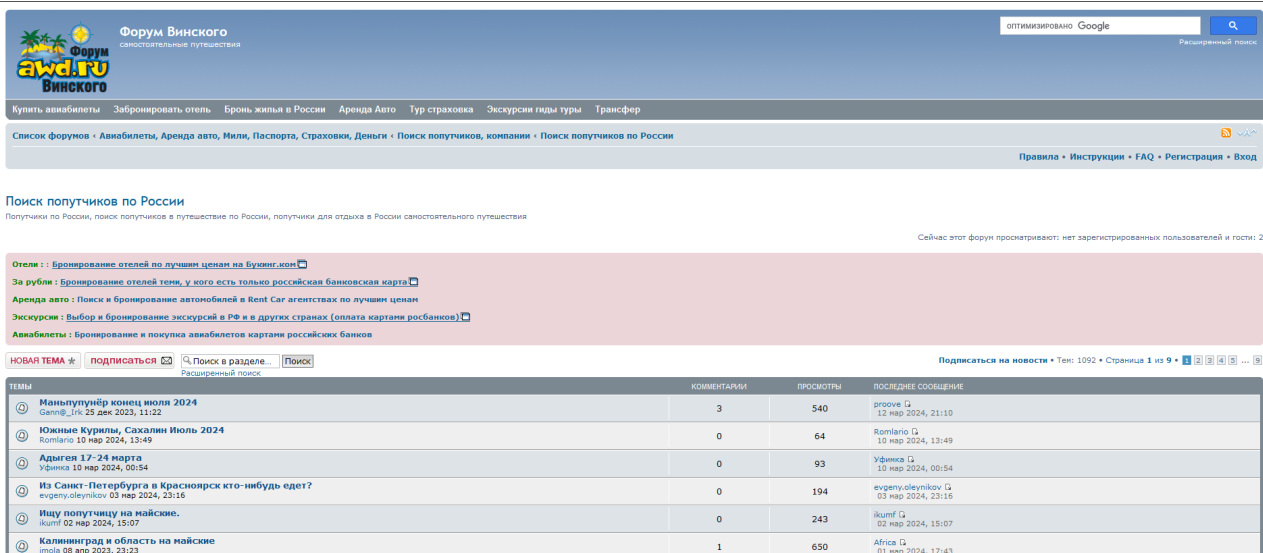


Рисунок 1.3 – Форум «Форум Винского»

Форум «Попутчик.ру» представляет пользователю возможность создать обсуждения в интересующей его категории для поиска попутчика. Для создания темы достаточно придумать заголовок и описание. После этого другие пользователи смогут оставить свои комментарии с вопросами или предложениями. У пользователя нет способа поиска тем удовлетворяющих его требованиям. Интерфейс форума представлен на рисунке 1.4.

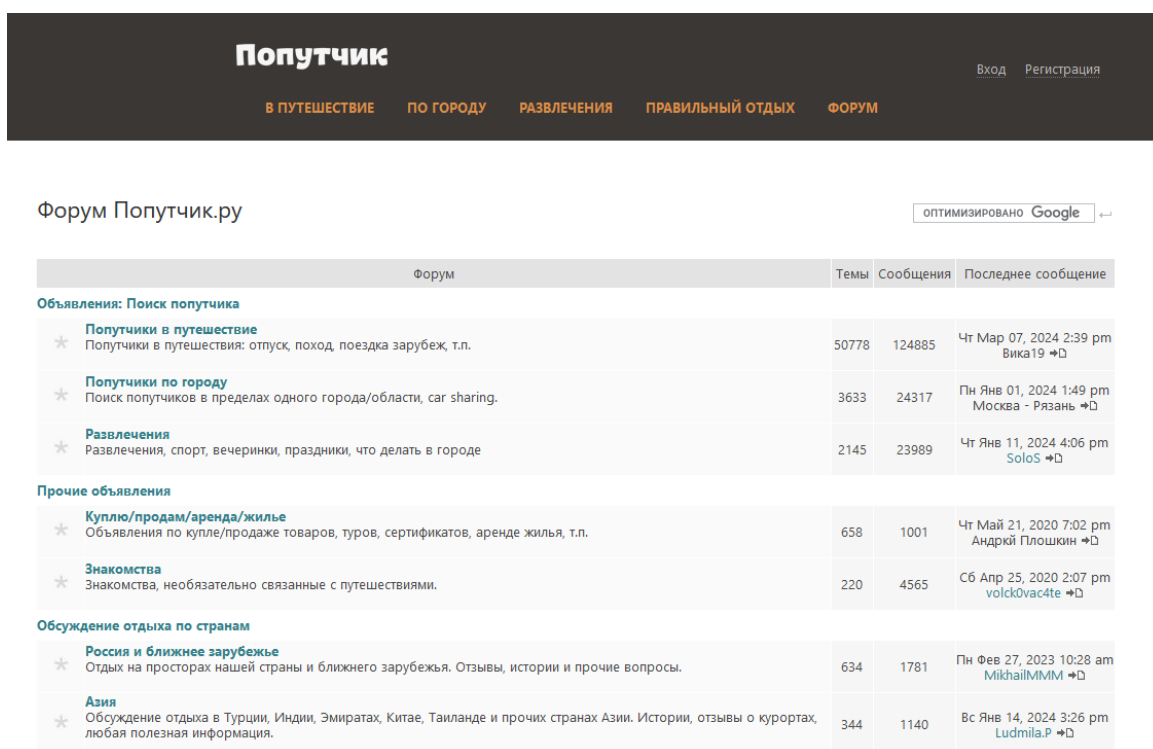


Рисунок 1.4 – Форум «Форум Попутчик.ру»

В ходе сравнительного анализа было установлено, что приложения имеют похожий функционал. Приложение «Едем.рф» предоставляет больше необходимой информации для тех кто ищет водителя, а приложение «BlaBlaCar» более дружелюбно к неопытным пользователям за счет собственного интерфейса. Форум «Форум Винского» и форум «Попутчик.ру» оказались крайне неэффективны в помощи поиска попутчика из-за малого функционала и неудобного интерфейса.

У приложений и форумов отсутствует возможность указать наличие и размер багажа у пассажиров и объёмы свободного места для багажа у водителя, а также возможности посмотреть маршрут не только водителю, но и пассажирам. Исходя из этого анализа становится ясно, что есть необходимость разработать приложение, которое будет обладать функцией указания размера багажа и вместительности автомобиля.

Сравнение функционала вышеописанных приложений представлен в таблице 1.1.

Таблица 1.1 – Результаты сравнительного анализа приложений.

Функция	Название приложения			
	BlaBlaCar	Едем.рф	Форум Винского	Попутчик.ру
Поиск попутчика	+	+	+	+
Создание поездки	+	+	+	+
Подбор оптимального маршрута	+	-	-	-
Интуитивный интерфейс	+	-	-	-
Задать размер багажа	-	-	-	-
Задать размер свободного места	-	-	-	-

Функция	Название приложения			
	BlaBlaCar	Едем.рф	Форум Винского	Попутчик.ру
Просмотреть маршрут как водитель	+	-	-	-
Просмотреть маршрут как пассажир	-	-	-	-

1.3 Постановка задачи

Задачей данного дипломного проекта является разработкой мобильного приложения. Приложение будет разрабатываться под ОС Android, так как она является самой популярной ОС для мобильных устройств. Приложение должно обладать следующим функционалом:

- регистрация и авторизация пользователей;
- поиск попутчиков по указанным параметрам:
 - 1) место отправки;
 - 2) пункт назначения;
- создание поездок с определенными параметрами:
 - 1) место отправки;
 - 2) пункт назначения;
 - 3) дата выезда;
 - 4) объем багажника;
 - 5) вместительность автомобиля;
- бронирование поездок и отмена бронирования поездок;
- создание и просмотр маршрута.

На основе поставленной задачи была разработана диаграмма вариантов использования (рисунок 1.5).

автомобиля для того, чтобы попутчики смогли определить нужное транспортное средства в месте сбора, количество пассажиров, которые могут присоединиться к поездке, объем багажника для багажа попутчиков и размер платы за поездку с каждого попутчика. После подтверждения информации о поездке, пользователь может просмотреть её маршрут.

					<i>ПГУ 1.090501.06.001 ПЗ</i>	Лист
						15
Изм.	Лист	№ докум.	Подпись	Дата		

2 Архитектура приложения

2.1 Выбор архитектуры приложения

Приложение состоит из серверной и клиентской части. Серверная часть – это система управления базами данных (СУБД), на ней будет храниться информация о пользователях и их аккаунтах, поездках и координаты мест отправки и пунктов назначения. Клиентская часть – это приложение с монолитной архитектурой, которое будет отправлять запросы на сервер на прямую, а не через микросервисы.

Монолитная архитектура была выбрана из-за следующих преимуществ:

- приложение просто разворачивать;
- приложение, созданное с одной базы кода легче разработать;
- более высокая производительность по сравнению с микросервисной архитектурой;
- в приложениях с монолитной архитектурой легче производить сквозное тестирование, чем при работе с распределенным приложением, так как оно является единым централизованным модулем;
- весь код находится в одном месте, благодаря чему становится легче выполнять запросы и находить проблемы во время отладки.

Архитектура проекта представлена на рисунке 2.1

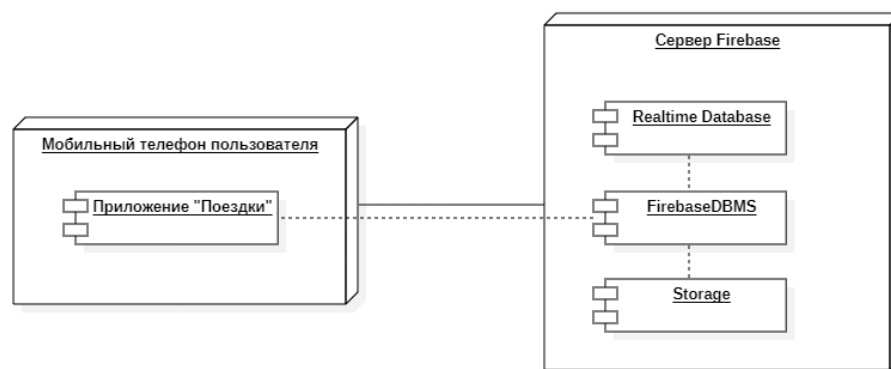


Рисунок 2.1 – Архитектура проекта

2.2 Разработка UML-диаграмм

На рисунках 2.2 – 2.4 изображены диаграммы деятельности, которые показывает, как пользователь будет взаимодействовать с приложением во время авторизации, поиска поездки и создания поездки.

При авторизации если у пользователя уже есть аккаунт, то ему достаточно ввести электронную почту, привязанную к аккаунту и пароль, если все данные верны, то произойдет вход в аккаунт и получен доступ ко всем функциям приложения. Если у пользователя нет аккаунта, то он должен будет зарегистрировать его. Для этого необходимо ввести фамилию, имя и отчество, которые можно будет изменить на страничке профиля после создания аккаунта, а также электронную почту и пароль, по которым будет осуществляться вход в аккаунт. После этого аккаунт будет создан, и пользователь сразу войдет в него и получит доступ ко всем функциям приложения.

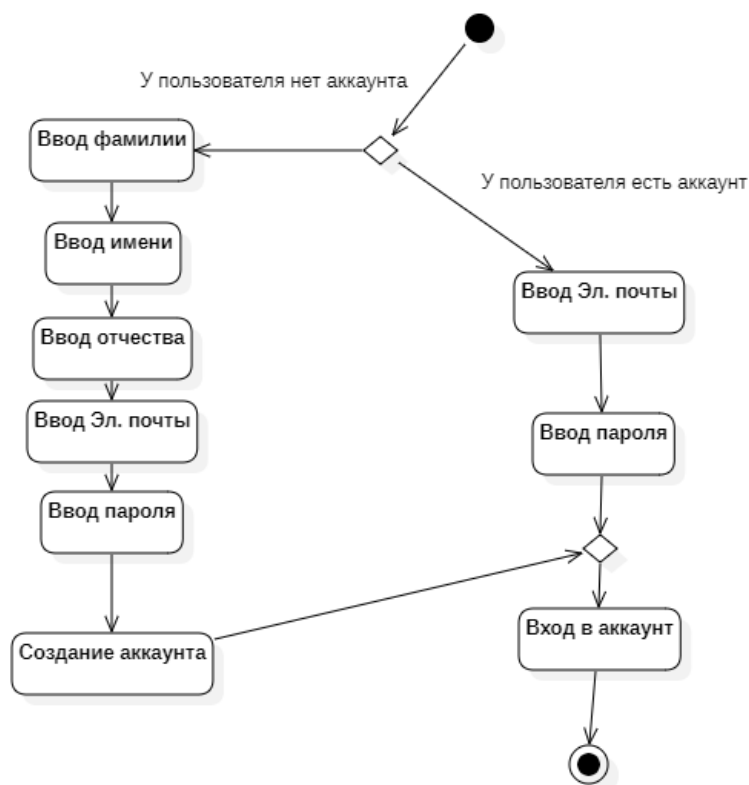


Рисунок 2.2 – Диаграмма деятельности: Авторизация пользователя

Для поиска поездки пользователь должен ввести название места отправки и название пункта назначения. После этого произойдет вывод списка поездок, удовлетворяющих условия пользователя, и информация о них.

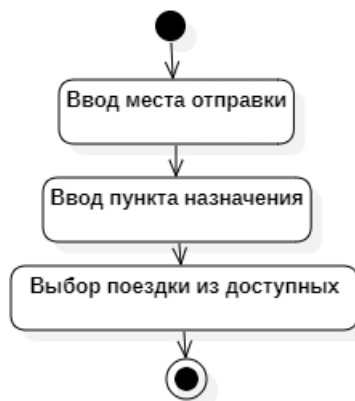


Рисунок 2.3 – Диаграмма деятельности: Поиск поездки

Для создания поездки пользователь должен ввести всю необходимую информацию. С начало нужно ввести названия места отправки и пункта назначения, потому что по ним другие пользователи смогут найти создаваемую поездку и будет построен маршрут. Затем необходима ввести комментарий, в нем может содержаться различная информация, которая может дать другим пользователям понять цели поездки, место встречи и другие детали. Затем необходимо ввести дату отправки. После этого пользователь должен внести информацию об своём транспортном средстве: марку, номер и цвет автомобиля. Затем ввести количество пассажиров и объем багажника, с помощью этой информации пользователи смогут понять могут ли они присоединиться к той или иной поездки. Затем необходимо ввести сколько должен будет заплатить каждый из пассажиров водителю за поездку. После подтверждения информации пользователь может создать поездку.



Рисунок 2.4 – Диаграмма деятельности: Создание поездки

На рисунке 2.5 изображена диаграмма данных на основе, которой будет разработана база данных. Разработанная база данных будет состоять из 6 сущностей: Auto, Pоеzdki, Coordinates, PU, User, Info.

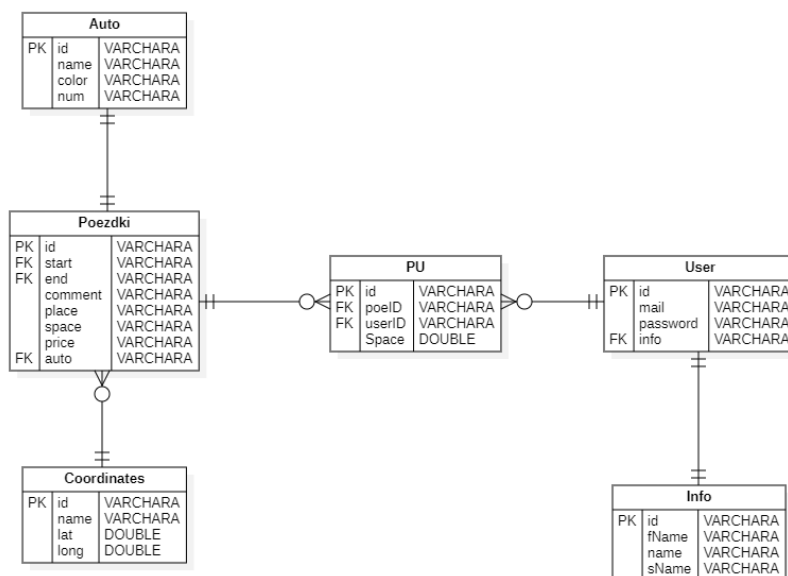


Рисунок 2.5 – Диаграмма данных

Таблица Auto содержит информацию о автомобилях. В таблице есть следующие поля:

- id – первичный ключ строкового типа;
- name – марка автомобиля строкового типа;
- color – цвет автомобиля строкового типа;
- num – номер автомобиля строкового типа.

Таблица Coordinates содержит наименования городов и их координаты. В таблице есть следующие поля:

- id – первичный ключ строкового типа;
- name – название города строкового типа;
- lat – географическая широта вещественного типа;
- long – географическая долгота вещественного типа.

Таблица Pоеzdki содержит информацию о поездках. Таблица содержит следующие поля:

- id – первичный ключ строкового типа;
- start – внешний ключ строкового типа на таблицу Coordinates;
- end – внешний ключ строкового типа на таблицу Coordinates;
- comment – комментарий строкового типа;
- place – количество свободных мест целочисленного типа;
- space – объём багажа вещественного типа;
- price – цена за поездку вещественного типа;
- auto - внешний ключ строкового типа на таблицу Auto.

Таблица Info содержит информацию пользователя. Таблица содержит следующие поля:

- id – первичный ключ строкового типа;
- fName – фамилия пользователя строкового типа;
- name – имя пользователя строкового типа;
- sName – отчество пользователя строкового типа.

Таблица user содержит информацию о аккаунтах пользователей. В таблице содержит следующие поля:

- id – первичный ключ строкового типа;
- mail – почта пользователя строкового типа;
- password – пароль пользователя строкового типа
- info – внешний ключ строкового типа на таблицу Info.

Таблица PU содержит информацию о том какой пользователь относится к какой поездке, и размер багажа пользователя. В таблице присутствуют следующие поля:

- id - первичный ключ строкового типа;
- poeID - внешний ключ строкового типа на таблицу Poezdki;
- userID - внешний ключ строкового типа на таблицу User;
- Space – объем багажа вещественного типа.

На рисунке 2.6 изображена диаграмма классов. На ней показано, как классы будут взаимодействовать между собой и поля этих классов. Классы Info, Coordinates, Poezdki, PU, User, Auto необходимы для получения данных из таблиц базы данных. Классы MainActivity, ProfileActivity, HomeActivity необходимы для отображения информации пользователя. Класс SearchActivity необходим для поиска поездок с помощью класса PoezdkiAdapter. Класс CreateActivity необходим для создания поездки. Класс RouteActivity необходим для построения и отображения маршрута.

На рисунке 2.7 изображена диаграмма последовательности. На ней показано взаимодействие клиента и сервера для создания аккаунта, авторизации, поиска поездки и создания поездки.

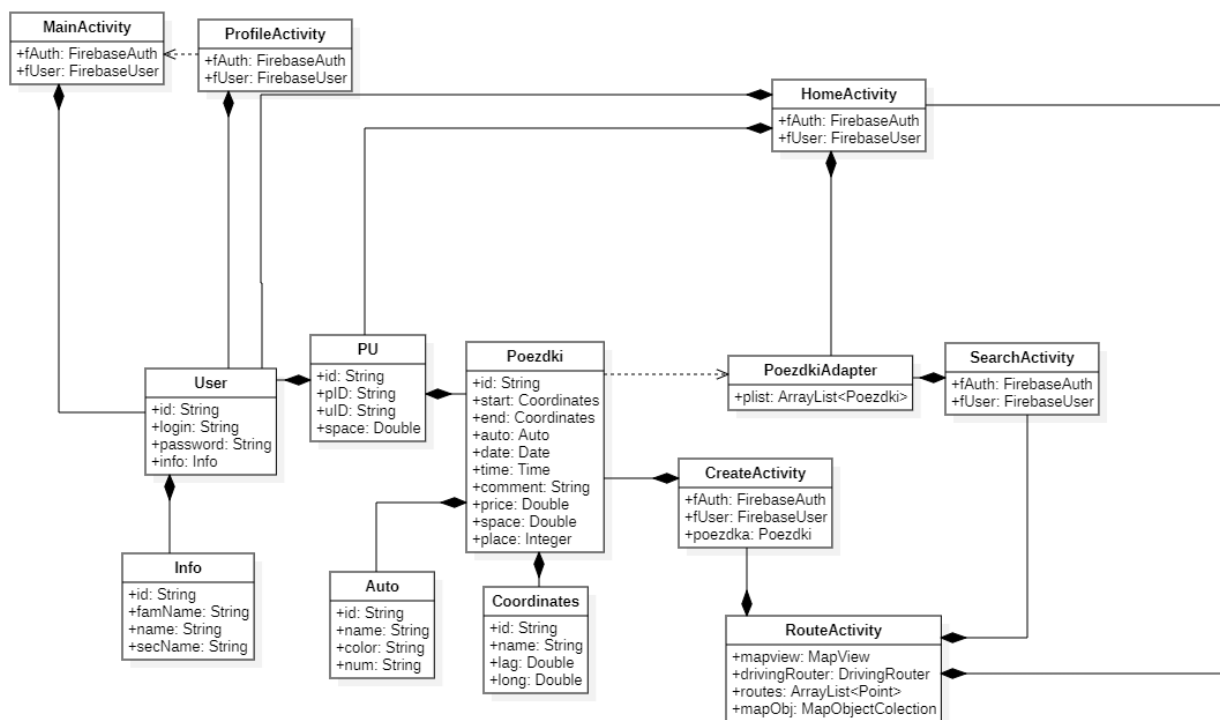


Рисунок 2.6 – Диаграмма классов

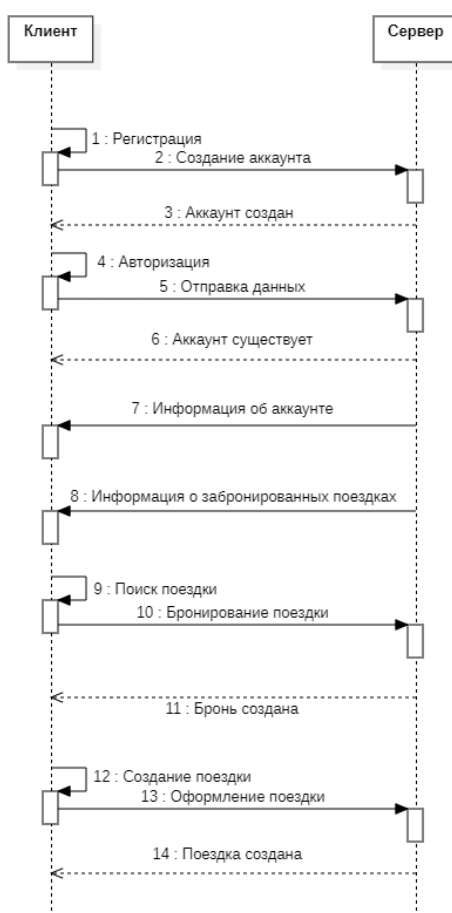


Рисунок 2.7 – Диаграмма последовательности

3 Выбор инструментов разработки

3.1 Выбор языка программирования

Для создания мобильного приложения для ОС Android можно использовать несколько разных языков программирования, но двумя самыми популярными являются Java и Kotlin [2]. У этих языков программирования есть как сходства, так и различия.

Из сходств можно выделить следующие особенности:

- Java и Kotlin являются языками со статической типизацией;
- оба языка являются бесплатными и распространяются как открытое программное обеспечение;
- Java и Kotlin являются объектно-ориентированными языками программирования.

Из основных различий можно выделить:

- Java превосходит Kotlin на 12–15 % по скорости компиляции кода;
- код, написанный на Kotlin, намного компактнее по сравнению с Java на 30–40 %;
- Java является более независим от платформы, так как программы, написанные на Java, могут запускаться на любой платформе, что упрощает создание кроссплатформенного приложения, что нельзя сказать про только набирающий популярность Kotlin;
- Java считается более защищённым языком программирования;
- Java считается более стабильным языком программирования;
- Java является более популярным языком разработки, чем Kotlin, что можно пронаблюдать на рисунке 3.1.

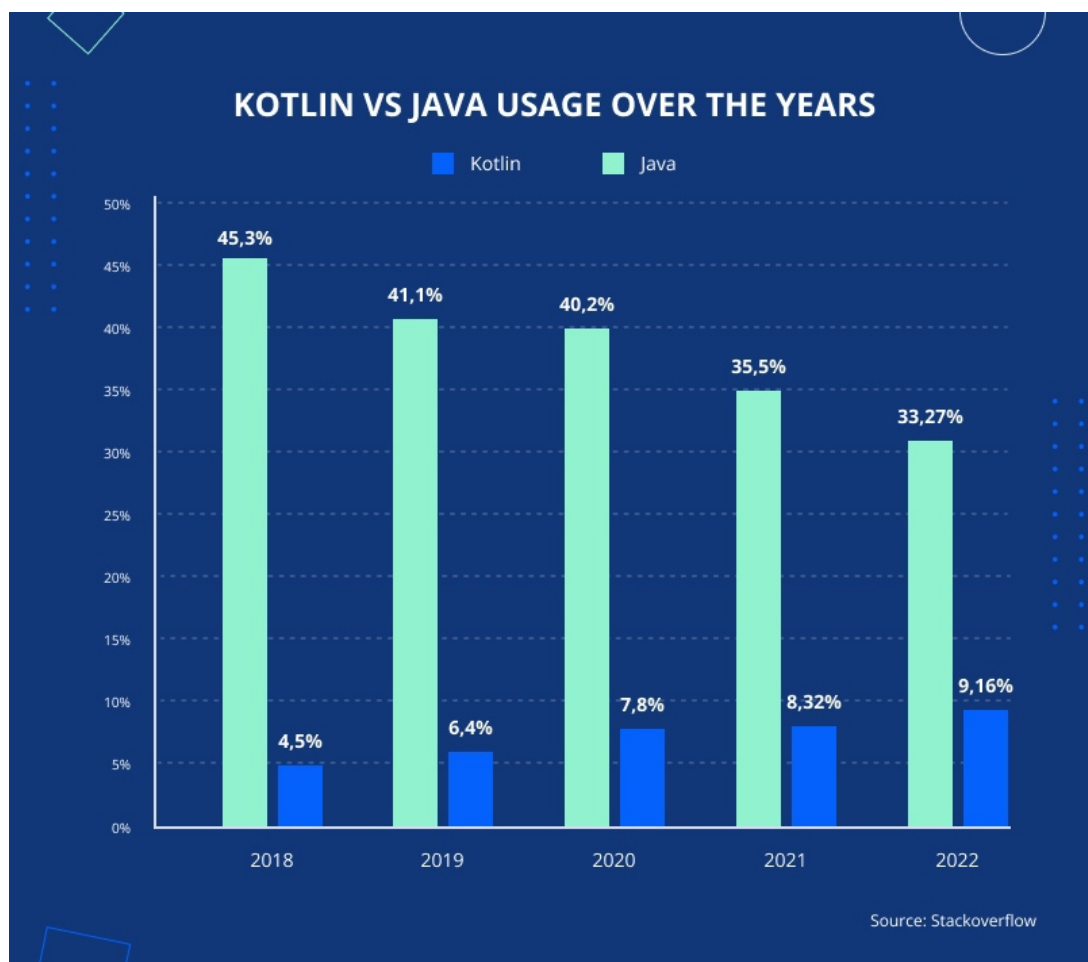


Рисунок 3.1 – Популярность Java и Kotlin

Учитывая преимущества и недостатки данных языков разработки, был выбран язык Java.

3.2 Выбор сервиса для отображения карт

Для создания мобильного приложения, которое будет использовать карты нужно выбрать SDK для работ с ними. Самыми популярными SDK для работы с картами в настоящее время являются Yandex MapKit, 2GIS, Google Maps SDK и OpenStreetMap.

С помощью Google Maps SDK можно добавлять в приложения для Android карты на основе данных сервиса "Google Карты" и настраивать их внешний вид. Кроме того, можно опубликовать дополнительные сведения о местах на карте и обеспечить возможность взаимодействия с этими данными, добавив маркеры,

многоугольники и наложения. Карты всего мира предоставляемые хорошо детализированы, но на карте России могут отсутствовать некоторые города и другие неточности. Помимо этого, данная SDK недоступно для разработчиков из России.

2GIS SDK позволяет добавить карту 2GIS в Android-приложение. С помощью этого SDK можно отобразить карту на экране, добавить на неё маркеры, рисовать геометрические фигуры, строить маршруты, получать информацию об объектах, управлять камерой и так далее. 2GIS SDK отличается от остальных SDK наличием высокой детализацией городов и наличием детальной количества информации о их инфраструктуре. У данной SDK есть бесплатная версия, но она имеет ограниченный функционал по сравнению с полной версией.

Yandex MapKit позволяет создать приложение с картами Яндекса для ОС Android. В таком приложении можно использовать поиск по организациям и топонимам, строить автомобильные и пешеходные маршруты с учетом актуальной дорожной ситуации, а также отображать информацию о пробках или панорамы. Карты Российской Федерации в Яндекс Картах являются одними из самых детализированных по сравнению с конкурентами. У данной SDK есть бесплатная версия, которая содержит функционал, не отличающийся от платной версии.

OpenStreetMap это бесплатный SDK, разработка которого поддерживается сообществом картографов для поддержания информации о дорогах, тропах, кафе, вокзалах и о многих других объектах по всему миру. Карты обладают большой точностью и имеют информацию, которой не располагают другие карты. Данная SDK не является популярной, из-за этого в интернете тяжело найти информацию об работы с ней.

Исходя из вышеперечисленного было решено использовать Yandex MapKit.

3.3 Выбор среды программирования

Существует огромное количество различных IDE для разработки мобильных приложений, самыми популярными из них являются: Android Studio и Eclipse [3].

Android Studio официальная среда программирования, разработанная компанией Google. Является одной самой популярной средой для разработки мобильных приложений для ОС Android и обладает инструментами упрощающую разработку мобильного приложения. Одним таким инструментом является встроенный эмулятор, который позволяет моментально проверит работу приложения.

Eclipse является бесплатной IDE имеющей минимальный функционал для разработки приложений на различных языках программирования. Данная среда программирования поддерживает большое количество различных плагинов для расширения доступного функционала. Несколько таких плагинов необходимы для создания мобильного приложения.

Из результатов анализа ясно, что лучше всего использовать среду разработки Android Studio. Именно поэтому в проекте было решено использовать ее.

3.4 Выбор сервиса облачной СУБД

Для хранения информации о различных пользователях необходимо использовать облачную СУБД. Самыми распространенными сервисами облачной СУБД являются Firebase и MySQL.

Firebase — это база данных NoSQL, которая хранит и синхронизирует данные в режиме реального времени. Данная база данных эффективно работает с большим набором данных и поддерживает серверные языки разработки приложений (такие как JSON и Node.js). Firebase имеет бесплатную и платную версии, которые не сильно отличаются своим функционалом.

MySQL — это система управления реляционными базами данных. Данная база данных имеет открытый исходный код и полностью бесплатная. Поддерживает большое количество языков программирования и хорошо обрабатывает сложные наборы данных.

Исходя из вышеперечисленного было решено использовать Firebase.

4 Разработка приложения

4.1 Подключение карт

Для того чтобы начать работу с Mapkit SDK [4], необходимо на сайте Яндекса получить API ключ. Для этого нужно зарегистрироваться на сервисе Яндекс, войти в личный кабинет для разработчиков, выбрать необходимую SDK и сгенерировать уникальный ключ (рисунок 4.1).

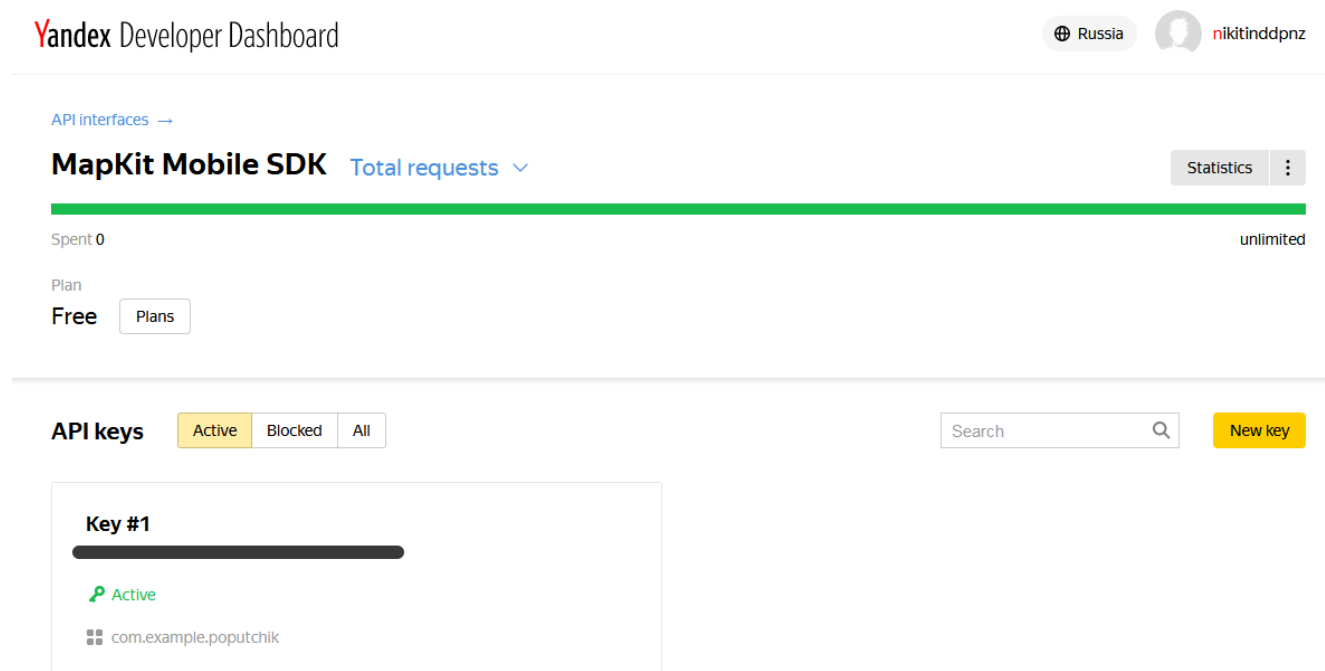


Рисунок 4.1 – Кабинет разработчика Яндекс

Для того чтобы приложение могла отображать карты Яндекса, перед инициализацией карт необходимо установить API ключ с помощью команды `MapKitFactory.setApiKey(MARKIT_API_KEY)`, где `MARKIT_API_KEY` – это строковая переменная, хранящая сгенерированный ключ. Затем нужно произвести инициализацию карт с помощью команды `MapKitFactory.initialize(this)`.

4.2 Построение маршрута

Для того чтобы построить маршрут с помощью Mapkit SDK необходимо создать запрос на автомобильные маршруты с помощью команды

DirectionsFactory.getInstance.createDrivingRouter(DrivingRouterType.COMBINED).

После чего происходит создание маршрута по заданным точкам:

```
private void submitRequest() {
    DrivingOptions drivingOptions = new DrivingOptions();
    VehicleOptions vehicleOptions = new VehicleOptions();
    ArrayList<RequestPoint> requestPoints = new ArrayList<>();
    requestPoints.add(
        new RequestPoint(start, RequestPointType.WAYPOINT, null, null));
    requestPoints.add(
        new RequestPoint(end, RequestPointType.WAYPOINT, null, null));
    drivingSession = drivingRouter.requestRoutes(
        requestPoints, drivingOptions, vehicleOptions, this);
}
```

Для того что отрисовать маршруты и расположить его в центре отображаемой карты используется следующий фрагмент кода:

```
@Override
    public void onDrivingRoutes(@NonNull List<DrivingRoute> routes)
    {
        if (routes != null && !routes.isEmpty()) {
            DrivingRoute route = routes.get(0);
            mapOb.addPolyline(route.getGeometry());
            mapView.getMapWindow().setFocusRect(new ScreenRect(new
                ScreenPoint(100f, 100f), new
                ScreenPoint(mapView.getMapWindow().width() - 100f,
                mapView.getMapWindow().height() - 100f)));
            BoundingBox box =
                BoundingBoxHelper.getBounds(route.getGeometry());
            boundingBoxPosition =
                mapView.getMap().cameraPosition(Geometry.fromBoundingBox(box));
            mapView.getMap().move(boundingBoxPosition);
        }
    }
```

После этого на карте будет отображен маршрут. Пользователь может изменить масштаб карты и передвигать её. На рисунке 4.2 представлен пример построенного маршрута из Москвы в Пензу.

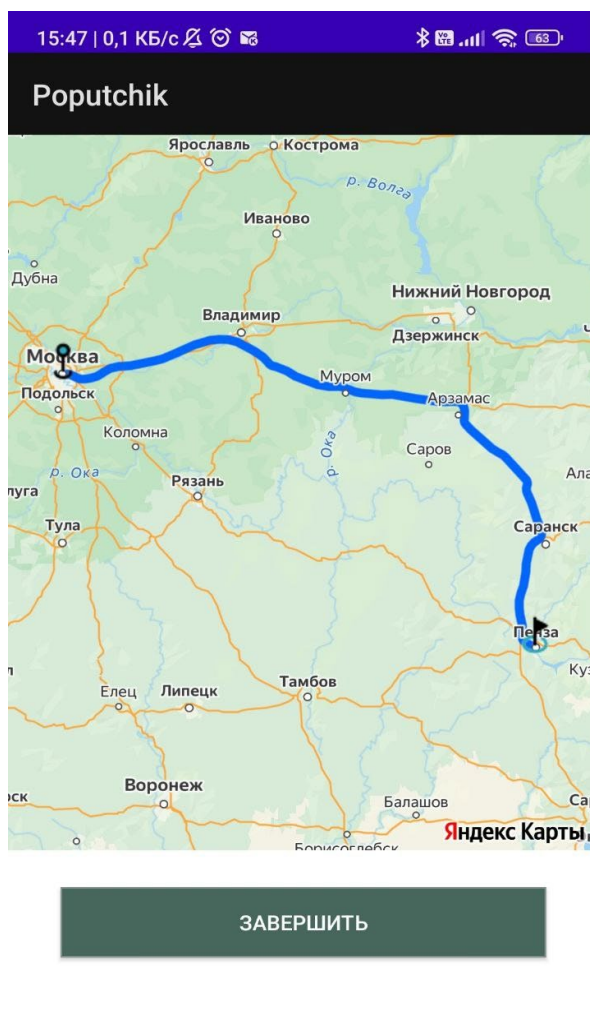


Рисунок 4.2 – Построенный маршрут

4.3 Подключение БД

Для того чтобы начать работать с Firebase [5], необходимо зарегистрироваться на их сайте и создать проект. После этого добавить к этому проекту разрабатываемую программу. Затем будет создан json файл, который необходимо добавить в программу с помощью вкладки Project. Далее необходимо инициализировать Firebase с помощью команды `FirebaseApp.initializeApp()`. На рисунке 4.3 изображен интерфейс проекта Firebase.

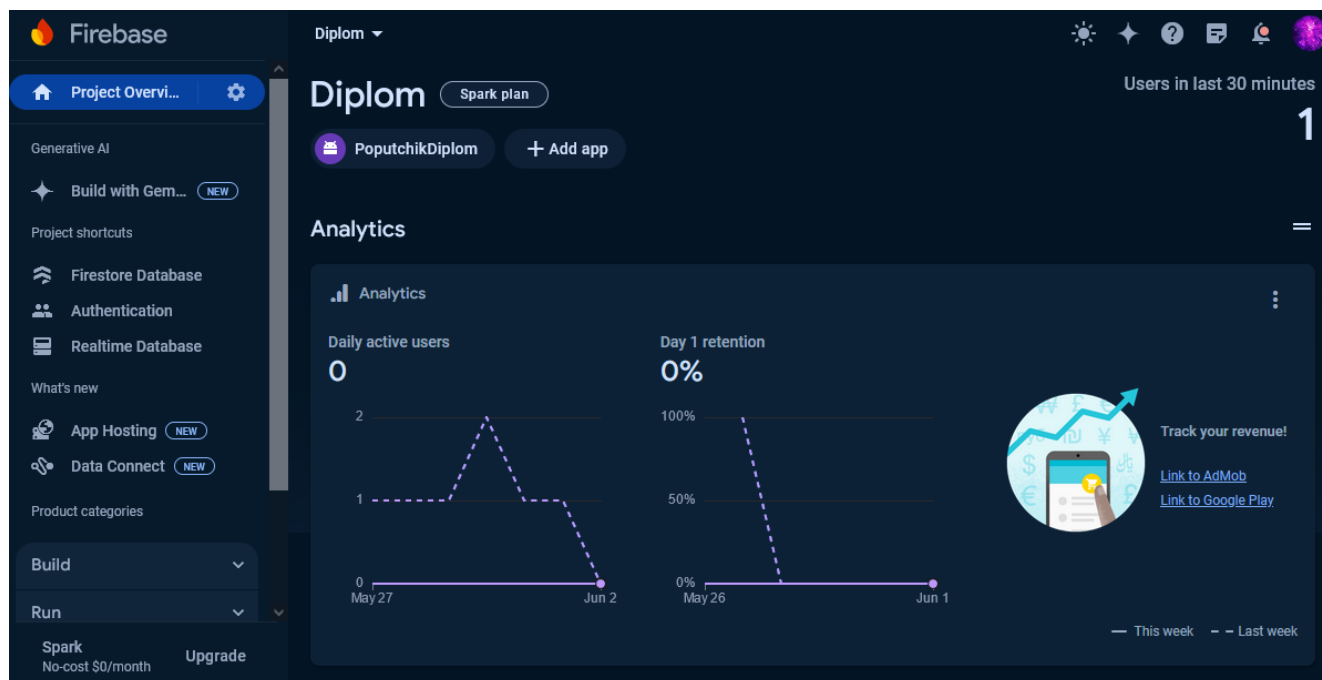


Рисунок 4.3 – Проект Firebase

4.4 Авторизация и регистрация пользователя

Для реализации авторизации необходимо использовать аутентификацию Firebase. Чтобы начать использовать аутентификацию Firebase необходимо добавить вкладку Authentication в окне проекта. Затем с помощью метода `signInWithEmailAndPassword()`, который принимает две строковые переменные для ввода электронной почты и пароля пользователя. Чтобы убедиться, что пользователь ввел необходимую информацию, проверяется были ли заполнены поля `EditText`.

Код функции авторизации пользователя приведен ниже:

```
btnLog.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String email = ((EditText) findViewById(
(R.id.idLogin))).getText().toString();
        String password = ((EditText) findViewById(
(R.id.idPassword))).getText().toString();
        if (!email.equals("") && !password.equals("")) {
            signIn(email, password);
        }
    }
});
```

```

    }
}

});

private void signIn(String email, String password) {
    Log.d("sign", "signIn:" + email);
    mAuth.signInWithEmailAndPassword(email,
password).addOnCompleteListener(Profile.this, new
OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if (task.isSuccessful()) {
            getUserInfo();
        } else {
            Toast.makeText(cnt, "Authentication failed.",
Toast.LENGTH_SHORT).show();
            getUserInfo();
        }
    }
});
}

```

Интерфейс формы с авторизацией приведен на рисунке 4.4. на форме авторизации располагаются два текстовых поля EditText для ввода логина и пароля и 2 кнопки Button для входа в профиль и для перехода на форму регистрации



Авторизация

Почта

Пароль

ВОЙТИ

РЕГИСТРАЦИЯ

Рисунок 4.4 – Авторизация

					ПГУ 1.090501.06.001 ПЗ	Лист
						31
Изм.	Лист	№ докум.	Подпись	Дата		

Для реализации регистрации используется метод `createUserWithEmailAndPassword()`, который принимает две строковые переменные для ввода электронной почты и пароля пользователя, после чего происходит ввод остальной информации пользователя.

Интерфейс формы регистрации приведен на рисунке 4.5. На форме регистрации располагается 5 объектов для ввода текста `EditText` необходимых для ввода информации пользователя, 5 текстовых объекта `Text` необходимых для подсказки пользователю, какую именно информацию он должен ввести для регистрации и 1 кнопка `Button` для завершения регистрации.

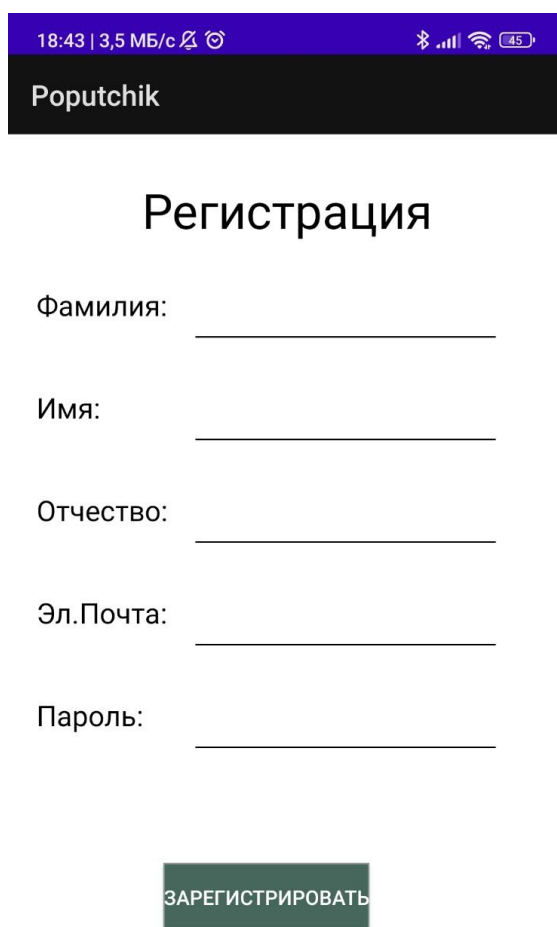


Рисунок 4.5 – Регистрация

После этого на страничке проекта Firebase во вкладках Authentication будет внесена информация о почте и пароле пользователя, необходимые для его авторизации (рисунок 4.6), а во вкладке Realtime Database будет сохранена

информация нового пользователя, которая будет отображена на странице его профиля (рисунок 4.7).

Identifier	Providers	Created ↓	Signed In	User UID
kool@mail.ru	✉	Jun 6, 2024	Jun 6, 2024	kAEyvEhKVshvL8ff5n450dEW...
kva@mail.ru	✉	Jun 4, 2024	Jun 4, 2024	n10VYVEbUuX7PVfxKgAXYC6...
alex@gmail.com	✉	Jun 4, 2024	Jun 4, 2024	Aj0E3VPCifgHa12pSqAOSBz...
alina@yandex.ru	✉	Jun 4, 2024	Jun 4, 2024	el2qFzDyFJ05pvj987zD4gYo3I...
belpen@yandex.ru	✉	Jun 4, 2024	Jun 4, 2024	qxZqjypVopdrLGDyvxX6XITKQ...
paadom@mail.ru	✉	Jun 4, 2024	Jun 4, 2024	IlRCrjjUBrQfqnz7aogwnlZbAF...
prokmoscow@gmail.co...	✉	Jun 4, 2024	Jun 4, 2024	jOTfr99QJ8VEFBegZZWITnXl...
aspenza@mail.ru	✉	May 24, 2024	May 24, 2024	zkvduxUDg0ZZ8KEAZwSVUMf...
nddpnz@gmail.com	✉	May 22, 2024	Jun 4, 2024	BeiQg1NDm9M4ZQogxFJ20U...

Rows per page: 50 1 - 9 of 9

Рисунок 4.6 – Вкладка *Authentication*

```

User
├── Aj0E3VPCifgHa12pSqAOSBzMjrK2
├── BeiQg1NDm9M4ZQogxFJ20UBTW82
├── IlRCrjjUBrQfqnz7aogwnlZbAFG3
├── el2qFzDyFJ05pvj987zD4gYo3Ih1
├── jOTfr99QJ8VEFBegZZWITnXlegq1
└── kAEyvEhKVshvL8ff5n450dEWHde2
    ├── email: "kool@mail.ru"
    ├── fName: "Куликов"
    ├── name: "Дмитрий"
    ├── password: "234567890kdbewgsibdj"
    └── sName: "Владимирович"
  
```

Рисунок 4.7 – Вкладка *Realtime Database*

Код функции регистрации пользователя приведен ниже:

```
btnReg.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String email = ((EditText) findViewById
(R.id.editTextTextEmailAddress)).getText().toString();
        String password = ((EditText) findViewById
(R.id.editTextTextPassword)).getText().toString();
        String fName = ((EditText) findViewById
(R.id.idFN)).getText().toString();
        String name = ((EditText) findViewById
(R.id.idName)).getText().toString();
        String sName = ((EditText)
findViewById(R.id.idSN)).getText().toString();
        if (!email.equals("") && !password.equals("") &&
!fName.equals("") && !name.equals("") && !sName.equals("")){
            createAccount(fName, name, sName, email,
password);}}});
private void createAccount(String fName, String name, String sName,
String email, String password) {
    Log.d("create", "createAccount:" + email);
    mAuth.createUserWithEmailAndPassword(email, password)
.addOnCompleteListener(Profile.this, task -> {
    if (task.isSuccessful()) {
        Log.d("create", "createUserWithEmail:success");
        user = mAuth.getCurrentUser();
        database = FirebaseDatabase.getInstance();
        DatabaseReference myRef = database.getReference
("User/" + user.getId() + "/name");
        myRef.setValue(name);
        myRef = database.getReference("User/" +
user.getId() + "/fName");
        myRef.setValue(fName);
        myRef = database.getReference("User/" +
user.getId() + "/sName");
```

					ПГУ 1.090501.06.001 ПЗ	Лист
						34
Изм.	Лист	№ докум.	Подпись	Дата		

```

        myRef.setValue(sName);
        myRef = database.getReference("User/" +
user.getId() + "/eMail");
        myRef.setValue(email);
        myRef = database.getReference("User/" +
user.getId() + "/password");
        myRef.setValue(password);
        getUserInfo();
    } else {
        Log.d("create", "createUserWithEmail:abortively");
    }
});}

```

4.5 Страницы пользователя

После авторизации или регистрации пользователь получает доступ к трем главным формам приложения. Форма профиля (рисунок 4.8), на которой располагается информация пользователя, форма поездок (рисунок 4.9), на которой выводятся все забронированные или созданные поездки пользователя, форма для поиска (рисунок 4.10), на которой можно ввести критерии для поиска и создать свою поездку. На всех трех формах присутствуют 3 кнопки Button внизу экрана для осуществления быстрого перехода между формами тремя главными формами приложения.

Форма профиля содержит четыре текстовых поля EditText для вывода и редактирования информации о пользователе, три кнопки Button, одна из которых становится видимой только после того, как пользователь начнет редактировать собственный профиль и позволяет сохранить новую информацию о профиле, вторая для начала редактирования и третья для осуществления выхода из профиля и приложения. Так же на этой форме есть четыре текстовых объекта Text для обозначения информации пользователя.

Форма поездок содержит список ListView необходимы для просмотра списка забронированных или созданных поездок пользователя, если у пользователя нет забронированных или созданных поездок, то эта страница будет пуста.

Форма поиска содержит два тестовых поля EditText с функцией автозаполнения, для быстрого ввода наименования города выезда и города пункта назначения, две кнопки Button для начала поиска и для создания своей поездки. Также на этой форме присутствует список ListView для вывода результата поиска, который будет пустым до начала поиска.

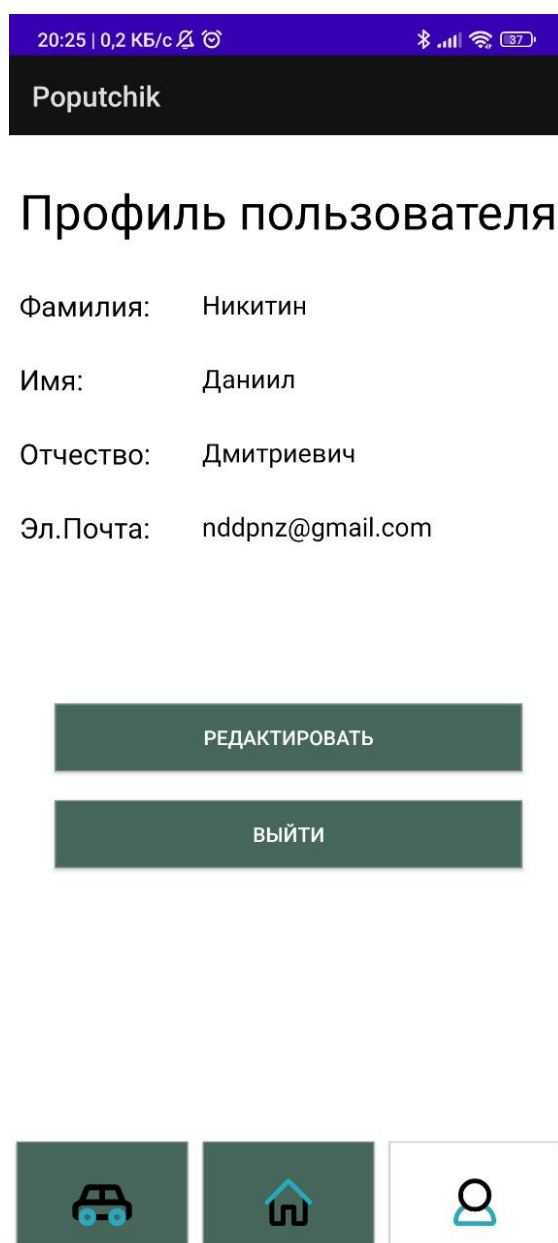


Рисунок 4.8 – Интерфейс формы профиля

Ваши поездки

Пенза -> Пермь

Дата	Время
14.09.2024	18:00
Свободные места: 2	
Объем багажника: 20	600 р



Рисунок 4.9 – Интерфейс формы поездок

Москва

Пенза

ПОИСК СОЗДАТЬ

Москва -> Пенза

Дата	Время
12.09.2024	17:50
Свободные места: 1	
Объем багажника: 10	600 р



Рисунок 4.10 – Интерфейс формы поиска

На форме профиля пользователь может изменить свою информацию. Для этого необходимо нажать на кнопку “Редактировать”. После чего поля ввода изменят свой цвет и их можно будет редактировать для ввода новой информации, также кнопка “Сохранить” перестанет быть невидимой, а кнопка “Редактировать” пропадет (рисунок 4.11). Чтобы закончить редактирование необходимо нажать на кнопку “Сохранить”. Нажав на кнопку “Выйти”, произойдет выход из профиля и приложения.

Рисунок 4.11 – Редактирование профиля

Код функции формы профиля приведен ниже:

```
bedit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
```

```

        EditText EName = (EditText) findViewById
(R.id.idName);

        EName.setEnabled(true);

EName.setBackgroundColor(Color.parseColor("#a8b5a6"));

        EditText EFN = (EditText) findViewById(R.id.idFN);
        EFN.setEnabled(true);
        EFN.setBackgroundColor(Color.parseColor("#a8b5a6"));
        EditText ESN = (EditText) findViewById(R.id.idSN);
        ESN.setEnabled(true);
        ESN.setBackgroundColor(Color.parseColor("#a8b5a6"));
        Button bedit2 = findViewById(R.id.buttonEdit);
        bedit2.setVisibility(View.INVISIBLE);
        Button bsave2 = findViewById(R.id.button4);
        bsave2.setVisibility(View.VISIBLE);
    }
});
bsave.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        EditText EName = (EditText)
findViewById(R.id.idName);
        EName.setEnabled(false);

EName.setBackgroundColor(Color.parseColor("#FFFFFF"));

        EditText EFN = (EditText) findViewById(R.id.idFN);
        EFN.setEnabled(false);
        EFN.setBackgroundColor(Color.parseColor("#FFFFFF"));
        EditText ESN = (EditText) findViewById(R.id.idSN);
        ESN.setEnabled(false);
        ESN.setBackgroundColor(Color.parseColor("#FFFFFF"));
        Button bedit2 = findViewById(R.id.buttonEdit);
        bedit2.setVisibility(View.VISIBLE);
        Button bsave2 = findViewById(R.id.button4);
        bsave2.setVisibility(View.INVISIBLE);
    }
});

```

```

        userName = EName.getText().toString();
        userFName = EFN.getText().toString();
        userSName = ESN.getText().toString();
        database = FirebaseDatabase.getInstance();
        DatabaseReference myRef =
database.getReference("User/" + user.getId() + "/name");
        myRef.setValue(userName);
        myRef = database.getReference("User/" +
user.getId() + "/fName");
        myRef.setValue(userFName);
        myRef = database.getReference("User/" +
user.getId() + "/sName");
        myRef.setValue(userSName);
    }
});
bexit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        mAuth.signOut();
        user = null;
        finishAffinity();
    }
});

```

При открытии формы поездок происходит вывод всех созданных пользователей поездок и забронированных пользователем поездок. Пользователь может нажать на поездку, чтобы вывести полную информацию о поездке (рисунок 4.12).

Код функции формы поездки приведен ниже:

```

@Override
    public void onDataChange(@NonNull DataSnapshot snapshot)
    {for(DataSnapshot poe: snapshot.getChildren()){
        String us = poe.child("u").getValue (String);
        String po = poe.child("p").getValue (String);
        if (Objects.equals(us, user.getId()))resPo(po);
    }
}

```

```

    }}

    listView.setOnItemClickListener((parent, view, position, id) -> {
        Poezdki p = poezdkis.get(position);
        String pID = p.ID;
        resultActivity(pID);});

```



Пенза->Пермь

14.09.2024 18:00

Кол-во свободных мест: 2

Свободный объем багажника: 20

Цена за пассажира: 600

Марка: волга

Цвет: красный

Номер: p207нк

Водитель: Никитин Д. Д.

тестовый комментарий



Рисунок 4.12 – Полная информация о поездке

На форме поиска, пользователь может создать свою поездку при нажатии кнопки “Создать” или произвести поиск по точки отправки и пункта назначения. Для этого пользователь должен заполнить поля EditText “Откуда?” и “Куда?” и после этого нажать кнопку “Поиск”. После этого произойдет вывод всех поездок, удовлетворяющих условия поиска. Так же, как и на форме поездок, пользователь может нажать на поездку для получения полной информации о ней.

Код функции формы поиска приведен ниже:

```

btn.setOnClickListener(new View.OnClickListener() {
    @Override

```

```

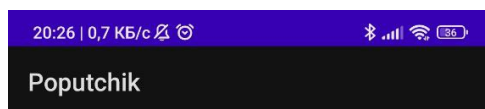
        public void onClick(View v) {
            start = autoTextView.getText().toString();
            end = autoTextView2.getText().toString();
            if (((!Objects.equals(start, "")) &&
(!end.equals(""))))) {
                searchPoezdki();
            }
        }
    });

    listView.setOnItemClickListener((parent, view, position, id) ->
{
        Poezdki p = poezdkis.get(position);
        String pID = p.ID;
        resultActivity(pID);
    });

```

4.6 Создание поездки

Для создания поездки пользователь должен заполнить всю информацию о поездке на пяти формах создания поездки. На первой форме необходимо ввести место отправки и пункт назначения в текстовые поля EditText (рисунок 4.13). На второй форме необходимо ввести в многострочное текстовое поле EditText комментарий о поездке, который описывает суть поездки (рисунок 4.14). На третьей форме необходимо ввести дату и время начала поездки в поля EditTextDate и EditTextTime (рисунок 4.15). На четвертой форме необходимо ввести информацию об автомобиле в пять текстовых полей EditText (рисунок 4.16). Пользователь должен ввести марку автомобиля, цвет автомобиля, номер автомобиля, количество пассажиров и объём для багажа. На пятой форме необходимо ввести в текстовое поле EditText цену, которую должен будет заплатить каждый из пассажиров после поездки (рисунок 4.17). На всех пяти формах есть две кнопки Button. При нажатии кнопки “Продолжить” пользователь перейдет на следующую форму. При нажатии кнопки “Отменить” пользователь перейдет на предыдущую форму.



Создание поездки

Укажите место выезда:

Откуда?

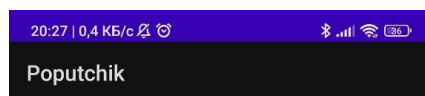
Укажите место приезда:

Куда?

ПРОДОЛЖИТЬ

ОТМЕНИТЬ

Рисунок 4.13 – Форма создания поездки



Комментарий

ПРОДОЛЖИТЬ

ОТМЕНИТЬ

Рисунок 4.14 – Форма комментария



Дата и время выезда

Дата

Время

ПРОДОЛЖИТЬ

ОТМЕНИТЬ

Рисунок 4.15 – Форма даты и времени



Информация об автомобиле

Марка автомобиля

Номер автомобиля

Цвет автомобиля

Кол-во пассажиров

Объем для багажа

ПРОДОЛЖИТЬ

ОТМЕНИТЬ

Рисунок 4.16 – Форма автомобиля



Цена

Укажите сколько должен будет
заплатить каждый из пассажиров:

Цена в рублях

ПРОДОЛЖИТЬ

ОТМЕНИТЬ

Рисунок 4.17 – Форма цены

Код функции формы создания приведен ниже:

```
bcreate.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        database = FirebaseDatabase.getInstance();  
        DatabaseReference myRef =  
database.getReference("Poezdki/");  
        String poezdkiID = myRef.push().getKey();  
        myRef = database.getReference("Poezdki/" +  
poezdkiID + "/user");  
        myRef.setValue(user.getId());  
        myRef = database.getReference("Poezdki/" +  
poezdkiID + "/start");  
        myRef.setValue(newCreate.Start);  
        myRef = database.getReference("Poezdki/" +  
poezdkiID + "/end");  
        myRef.setValue(newCreate.End);  
    }  
});
```

```

        myRef = database.getReference("Poezdki/" +
poezdkiID + "/date");
        myRef.setValue(newCreate.Date);
        myRef = database.getReference("Poezdki/" +
poezdkiID + "/time");
        myRef.setValue(newCreate.Time);
        myRef = database.getReference("Poezdki/" +
poezdkiID + "/autoname");
        myRef.setValue(newCreate.AutoName);
        myRef = database.getReference("Poezdki/" +
poezdkiID + "/autocolor");
        myRef.setValue(newCreate.AutoColor);
        myRef = database.getReference("Poezdki/" +
poezdkiID + "/autonum");
        myRef.setValue(newCreate.AutoNumber);
        myRef = database.getReference("Poezdki/" +
poezdkiID + "/place");
        myRef.setValue(newCreate.Place);
        myRef = database.getReference("Poezdki/" +
poezdkiID + "/space");
        myRef.setValue(newCreate.Space);
        myRef = database.getReference("Poezdki/" +
poezdkiID + "/price");
        myRef.setValue(newCreate.Price);
        myRef = database.getReference("Poezdki/" +
poezdkiID + "/com");
        myRef.setValue(newCreate.Comment);
        DatabaseReference myRef2 =
database.getReference("PU/");
        String puID = myRef2.push().getKey();
        myRef2 = database.getReference("PU/" + puID +
"/u");
        myRef2.setValue(user.getUid());
        myRef2 = database.getReference("PU/" + puID +
"/p");

```

```

        myRef2.setValue (poezdkiID);
        Intent i = new Intent(cnt, create5.class);
        i.putExtra("ID", poezdkiID);
        i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
        startActivity(i);
    }
});

```

4.7 Бронирование поездки

Для бронирования поездки необходимо перейти на форму с поной информацией о поездке. Если пользователь не бронировал или создавал эту поездку, то на форме будет кнопка “Присоединиться” (рисунок 4.18).

16:46 3.37 MB/c

Ропутчик

Москва->Еманжелинск

12.09.2024 9:50

Кол-во свободных мест: 9

Свободный объем багажника: 90

Цена за пассажира: 999

Марка: Газель

Цвет: белая

Номер: п345оп

Водитель: Зелепухин А. С.

Одному скучно

ПРИСОЕДИНИТЬСЯ

МАРШРУТ

НАЗАД

Рисунок 4.18 – Страница поездки

После нажатия на кнопку “Присоединиться” пользователь отправится на форму оформления бронирования, где ему необходимо ввести в текстовой поле EditText, объём своего багажа, если пользователь не введет объём багажа и оставит его пустым, то объём его багажа будет равен 0 (рисунок 4.19). Если пользователь введет объём больше, чем свободно, то пользователь не сможет присоединиться к поездке.

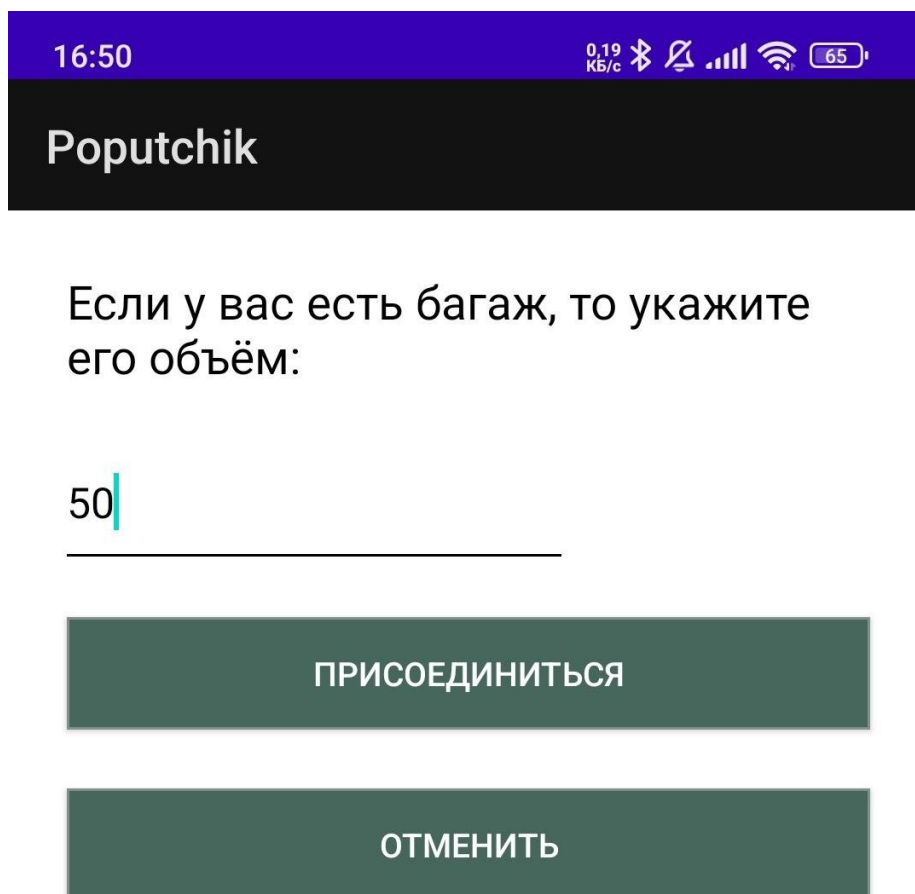


Рисунок 4.19 – Форма оформления бронирования

После этого, если пользователь зайдет на страницу с полной информацией о поездке, то количество свободных мест и свободный объем для багажа изменятся (рисунок 4.20). Также кнопка “Присоединиться” пропадет и вместо неё появится кнопка “Выйти”. Если пользователь “Выйти”, то количество свободных мест и свободное место для багажа вернутся в изначальное значение.

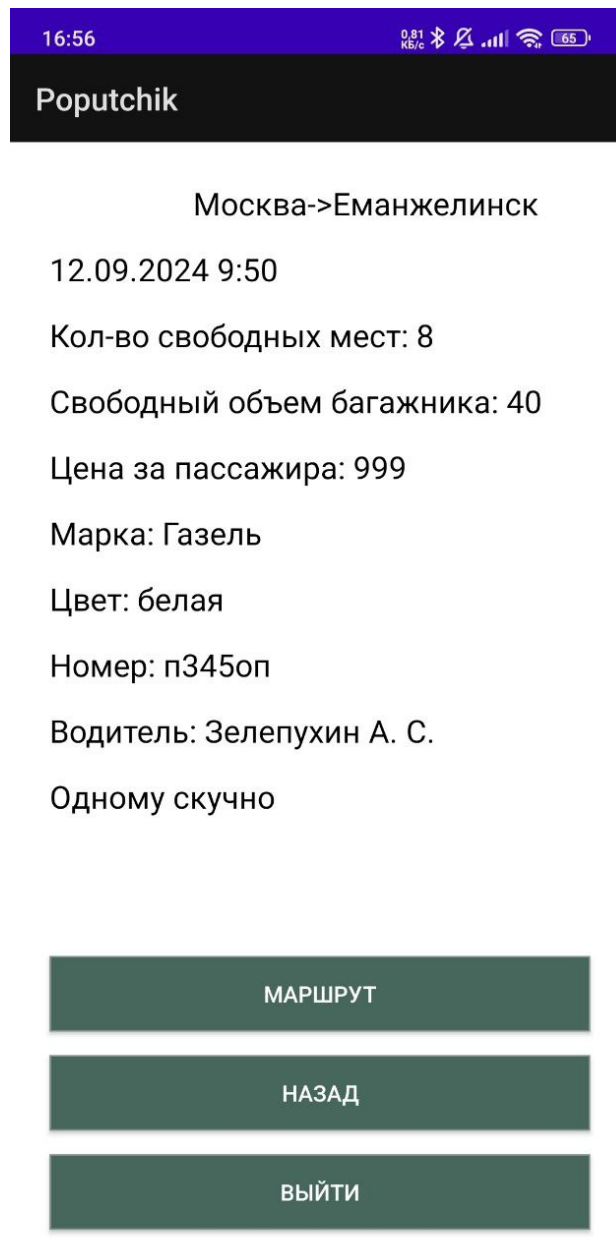


Рисунок 4.20 – Страница забронированной поездки

Код функции бронирования приведен ниже:

```
join.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String weight = nW.getText().toString();
        if (weight.equals("")) weight = "0";
        if (Integer.valueOf(sp) > Integer.valueOf
(weight)) {
            database = FirebaseDatabase.getInstance();
```

```

        DatabaseReference myRef =
database.getReference("PU/");

        String puID = myRef.push().getKey();
        myRef = database.getReference("PU/" + puID
+ "/u");

        myRef.setValue(user.getUid());
        myRef = database.getReference("PU/" + puID
+ "/p");

        myRef.setValue(pID);
        myRef = database.getReference("PU/" + puID
+ "/w");

        myRef.setValue(weight);
        Integer i = Integer.valueOf(pl);
        String npl = String.valueOf(i - 1);
        i = Integer.valueOf(sp);
        Integer j = Integer.valueOf(weight);
        String nsp = String.valueOf(i - j);
        database = FirebaseDatabase.getInstance();
        DatabaseReference myRef3 =
database.getReference("Poezdki/" + pID + "/place");
        myRef3.setValue(npl);
        myRef3 = database.getReference("Poezdki/" +
pID + "/space");

        myRef3.setValue(nsp);
        start = "";
        end = "";
        finish();
    }
}
});

```

5 Технико-экономическое обоснование

5.1 Введение

Технико-экономическое обоснование – это оформленные результаты расчетов экономической целесообразности разработки проекта, анализ результатов расчетов и их оценка. Для вычисления необходимо учесть все затраты, которые были произведены в ходе работы над программой.

Для определения экономической целесообразности необходимо рассчитать итоговые расходы, которые включают в себя:

- расходы на капитальные вложения;
- затраты на разработку программы;
- заработная плата разработчика.

5.2 Капитальные вложения

Для расчета капитальных вложений необходимо рассчитать:

- стоимость ЭВМ, на которой производилось разработка программы;
- стоимость лицензированного программного обеспечения, которое было необходимо для разработки программы.

В таблице 5.1 представлена информация о комплектующих ПК, на котором разрабатывалась программа, и цены на них.

Таблица 5.1 – Комплектующие ПК

Наименование комплектующей	Цена	Источник
Процессор Intel Core i5-10400F OEM	9500	«М Видео»
Видеокарта MSI GeForce RTX 3050	27000	«М Видео»
Клавиатура	4500	«М Видео»

Наименование комплектующей	Цена	Источник
Мышь	2000	«М Видео»
Оперативная память Kingston	15000	«М Видео»
SSD	8200	«М Видео»

Основываясь на данные, приведенные в таблице 5.1, можно произвести расчеты затраты на оборудование. Итоговые затраты на оборудование равны 66200 рублей.

В ходе разработки было использовано следующее ПО:

- Операционная система Microsoft Windows 10 Pro;
- Yandex Mapkit SDK;
- Firebase.

В таблице 5.2 представлена информация о лицензионном программном обеспечении, которое было использовано в ходе разработки.

Таблица 5.2 – Лицензионное программное обеспечение

Наименование	Цена/Срок	Источник
Операционная система Microsoft Windows 10 Pro	20000/бессрочно	“ДНС”
Yandex Mapkit SDK	45000/1 месяц	“Яндекс”
Firebase	29000/1 месяц	“Google”

Программное обеспечение, представленное в таблице 5.2 необходимо на все время разработке, которое заняло 2 месяца.

Основываясь на данных из таблицы 5.2, можно произвести расчеты затраты на программное обеспечение. Итоговые затраты на программное обеспечение равны 168000 рублей.

В результате произведенных расчетов можно произвести расчет капитальных вложений, которые равны сумме между затратами на программное обеспечение и затратами на ЭВМ. Капитальные вложения равны 234200 рублей.

5.3 Затраты на разработку программы

Для расчета необходимо определить затраты по заработной плате и накладные расходы. В нашем случае необходимо определить затраты по заработной плате инженера-программиста. Средняя заработная плата инженера-программиста равна 57000 рублей в месяц. Поскольку разработка программы заняла два месяца, то затраты по заработной плате инженера программиста равны 114000 рублей.

Для расчета накладных расходов необходимо перемножить среднюю заработную плату инженера-программиста, коэффициент дополнительной заработной платы инженера программиста равный 1,35 и коэффициент начисления органам социального страхования равный 1,28. Из этого следует, что накладные расходы равняются 98496 рублей.

Из всего вышеперечисленного можно рассчитать затраты на разработку. Итоговые затраты на разработку равны 212496 рублей.

5.4 Заработная плата

Для расчета основной заработной платы разработчика необходимо перемножить количество месяцев работы на среднюю заработную плату инженера-программиста.

На разработку программы потребовалось 2 месяца, а средняя заработная плата инженера-программиста приведенная ранее равна 57000 рублей. В результате основная заработная плата инженера программиста будет составлять 114000 рублей

Для вычисления итоговой заработной платы необходимо учесть затраты на дополнительную заработную плату, составляющую 15% от основной. Из этого можно сделать вывод, что итоговая заработная плата, которая будет выплачена разработчику равно 131100 рублей.

5.5 Итоговые расходы

В таблице 5.3 приведены итоговые расходы на разработку программы для улучшения логистики передвижения.

Таблица 5.3 – Итоговые расходы

Наименование	Значение (рубли)
Капитальные вложения	234200
Затраты на разработку программы	212496
Заработная плата	131100
Итоги	577796

6 Безопасность жизнедеятельности

6.1 Характеристика опасных факторов при работе с ПЭВМ

Во время разработки программы, приложения или любой другой деятельности за персональным компьютером, на разработчика могут влиять различные внешние факторы, которые могут негативно повлиять на его производительность. Взаимодействие инженера-программиста с этими факторами необходимо снизить на сколько это возможно путем создания комфортного рабочего места и оградить разработчика от внешних факторов различными методами. В первую очередь необходимо определить какие факторы могут влиять на работоспособность разработчика. Такие факторы часто называют опасными факторами и их можно разделить на четыре группы:

- психофизиологические;
- физические;
- химические;
- биологические.

Под группой психофизиологических факторов в первую очередь имеют ввиду следующие факторы:

- физическое перенапряжение организма;
- умственное перенапряжение человека;
- усталость глаз;
- головные боли;
- боли в спине;
- боли в шее.

Долгое нахождение за компьютером во время разработки или в свободное от разработки время может привести к сильному снижению работоспособности разработчика. Продолжительное нахождение в сидячем положении приводит к перенапряжению и сильной усталости, затеканию конечностей. Так же при неправильном нахождении за своим рабочим местом может привести к болям в

шее, спине и ухудшению осанки. При долгом просмотре в экран персонального компьютера у разработчика может проявиться сухость в глазах или слезоточивость, что может привести к ухудшению и потере зрения. Из всего вышеперечисленного можно сделать вывод о том, что долгое нахождение перед персональным компьютером может привести к сильному ухудшению здоровья разработчика, что в свою очередь повлияет на его работоспособность. Что бы предотвратить такой исход инженер-программист обязан периодически сменять умственную деятельность на физическую путем проведения зарядки или любой другой физической активности, так же стоит соблюдать правильную осанку во время работы и расположить экран монитора на расстоянии 45 – 60 сантиметров от разработчика и сам разработчик должен смотреть на экран сверху в низ.

К группе физических факторов относятся:

- рентгеновское излучение, влияющее на разработчика, выше допустимого уровня;
- инфракрасное излучение, влияющее на разработчика, выше допустимого уровня;
- статическое электричество, влияющее на разработчика, выше допустимого уровня;
- ультрафиолетовое излучение, влияющее на разработчика, выше допустимого уровня;
- слишком низкая температура в рабочей области;
- слишком высокая температура в рабочей области;
- большое количество пыли в воздухе рабочей области, на рабочих поверхностях и на поверхности персонального компьютера разработчика;
- недостаточная освещенность рабочей области;
- слишком высокая освещенность рабочей области;
- чрезмерное попадание солнечных лучей на экран монитора персонального компьютера разработчика или на лицо разработчика;

Некоторые из физических факторов затрудняют работу из-за своего воздействия на рабочее место, но большинство из них влияют непосредственно на самого разработчика. Воздействие этих факторов может не только ухудшить работоспособность, но и привести здоровье инженера-программиста в критическое состояние. Для избежания подобного исхода необходимо оградить рабочую область и самого разработчика от подобных факторов на сколько это возможно. Этого можно добиться как прямыми ограждениями разработчика от источников таких факторов, так и расположение рабочего места на приемлемом расстоянии от них.

Третьей группой опасных факторов является химические факторы. К подобным факторам относятся:

- содержание углекислого газа в воздухе рабочего пространства превышающее нормы;
- нахождение в рабочем пространстве с повышенной температурой большого количества некачественного пластика;
- чрезмерное количество вредных веществ, таких как формальдегид и озон в воздухе рабочего пространства.

Химические факторы в первую очередь связаны с воздушным пространством рабочего места. При некачественной вентиляции разработчик будет вынужден вдыхать вредные вещества во время нахождения за рабочим местом, что приведет к сильному ухудшению здоровья, головным болям и потери сознания во время работы.

Под биологическими факторами в первую очередь понимают нахождение микроорганизмов в воздушном пространстве рабочего места. Для того чтобы избежать подобного необходимо обеспечить рабочее место хорошей вентиляцией и заниматься периодическим проветриванием рабочего места.

6.2 Рабочее помещение

Рабочим помещением для программиста или разработчика в первую очередь является офисное помещение. В офисе проходит большая часть рабочей деятельности разработчика или инженера-программиста. Для обеспечения стабильности в работоспособности разработчиков в офисе необходимо обеспечить различными условиями комфорта и безопасности.

Одним из таких условий является обеспечение пожарной безопасности путем наличия нескольких пожарных выходов. В экстренной ситуации разработчики должны знать о наличии и расположении таких выходов для безопасной эвакуации из помещения.

Самое главное для обеспечения комфорта в помещении необходимо поддерживать допустимую температуру. По стандартам СанПиН 2.2.4.3359-16 [6] температура в офисе должна поддерживаться от 22 до 26 градусов. Подобная температура должна поддерживаться, как в холодное, так и в жаркое время года. Этого можно добиться с помощью кондиционеров, проветривания помещения, обогревателей и систем отопления помещения.

6.3 Санитарно-гигиеническая характеристика

Во время работы программист в большей части находится в сидячем положении, и сама работа не оказывает больших физических нагрузок. В связи с этим нормами СанПиН рабочее место программиста должно придерживаться следующих нормативов и требований:

- на рабочем месте программиста должна поддерживаться температура от 22 до 26 градусов цельсия на всем протяжении рабочей деятельности;
- на рабочем месте программиста относительная влажность не должна превышать 75% на всем протяжении рабочей деятельности;
- на рабочем месте программиста скорость движения воздуха не должна превышать 0,1 м/с.

В связи с тем, что большая часть рабочей деятельности разработчика связана со просмотром в экран монитора персонального компьютера, под конец рабочего дня у большинства разработчиков проявляется боли в глазах, повышенная слезоточивость, сухость глаз. Как сказано выше это вызывается в первую очередь продолжительным просмотром в монитор, но также это может быть вызвано несколькими факторами. К таким факторам можно отнести:

- редкое моргание;
- плохая освещенность рабочего места разработчика;
- слишком высокая освещённость рабочего места;
- попадание солнечных лучей в глаза разработчика в чрезмерном количестве;

Предотвратить боль в глазах является одной из первостепенных задач, так как ухудшение зрения или полная его потеря может значительно понизить работоспособность инженера-программиста.

Для защиты зрения программиста необходимо соблюдать определённые требования как к рабочему месту, так и к самому программисту. К основным требованиям относятся:

- не в коем случаи не производить работу за экраном монитора при низкой освещенности рабочего пространства;
- не в коем случаи не производить работу за экраном монитора при слишком высокой освещенности рабочего пространства;
- программист должен периодически отвлекаться от экрана монитора для того, чтобы дать глазам отдохнуть;
- программист должен периодически совершать физические упражнения для поддержания собственного здоровья;
- на рабочем месте необходимо поддерживать баланс освещенности;
- на рабочем месте необходимо поддерживать сочетание искусственного и натурального освещения;
- не допускать чрезмерного попадания солнечных лучей в лицо разработчика;

– не допускать чрезмерного попадания солнечных лучей на экран монитора персонального компьютера разработчика.

6.4 Электробезопасность

При работе разработчик постоянно на прямую взаимодействует с электроприборами. В связи с этим абсолютно необходимо соблюдать правила электробезопасности. Электрический ток представляет большую опасность для здоровья человека. От прямого взаимодействия с электрическим током у человека может возникнуть электрические ожоги в месте соприкосновения и по всему телу, повреждение внутренних органов. Одно из опасностей воздействия электрического тока, это то, что не всегда наблюдатель может определить наличие такого воздействия. В случае возникновения такого взаимодействия человеку необходимо как можно быстрее оказать первую медицинскую помощь и вызвать врачей.

При работе над приложением или программой у разработчика всегда есть вероятность попасть под влияние электрического тока. Уменьшение этой вероятности является первостепенной задачей. Для этого необходимо следовать определенным правилам и требованиям:

- необходимо поддерживать все электрическое оборудование в пригодном состоянии;
- необходимо периодически проверять целостность электрического оборудования;
- не позволять сбору большого количества пыли на оборудовании и рабочих поверхностях;
- поддержание нормальной температуры рабочего пространства для работы разработчиков и для нормального функционирования всего электрического оборудования;
- поддержания относительной влажности рабочего пространства для нормальной работы разработчиков и оборудования;

					ПГУ 1.090501.06.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

- не в коем случаи не допускать попадания каких-либо жидкостей на исправное или не исправное электрическое оборудование;
- не в коем случаи не допускать работу на электрическом оборудовании с какими-либо повреждениями;
- не в коем случаи не допускать взаимодействия с электропроводкой и другими кабелями, если они имеют внешние повреждения.

Помимо всего вышеперечисленного все разработчики и другие сотрудники, которые работают с любыми электроприборами должны знать правила электробезопасности. Для этого работодатель обязан периодически проводить организационные беседы в целях обеспечения сотрудников знаниями необходимыми для избежания взаимодействия с электрическим током и для оказания первой медицинской помощи в случаи если подобное взаимодействие произошло.

					ПГУ 1.090501.06.001 ПЗ	Лист
						61
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В процессе выполнения выпускной квалификационной работы была изучена предметная область, сформулировано техническое задание на разработку приложения оптимизации логистики передвижения на примере поиска попутчиков.

Был произведен анализ и подбор инструментов разработки для реализации мобильного приложения. Выполнено проектирование, в результате которого были разработаны UML-диаграммы разрабатываемого программного обеспечения.

Разработаны макеты окон приложения, структура базы данных. Были разработаны функции авторизации, регистрации, поиска поездок, создания поездок, построения поездок, внесения всей необходимой информации в базу данных и вывод необходимой информации из неё.

В разделе безопасности жизнедеятельности были описаны требования к офисным помещениям и рабочему месту для минимизации воздействия вредных факторов на организм человека.

В экономическом разделе был произведен расчет затрат на разработку программного обеспечения.

Таким образом, требования технического задания были выполнены в полном объёме и в установленные сроки.

					<i>ПГУ 1.090501.06.001 ПЗ</i>	Лист
						62
Изм.	Лист	№ докум.	Подпись	Дата		

Список используемой литературы

1. Логистика: персонал, технологии, практика: учебно-практическое пособие / Е. В. Панасенко // Москва: Инфра-Инженерия, 2011. - 222, [1] с.: ил., табл.; 21 см.; ISBN 978-5-9729-0034-3.

2. Kotlin vs Java: Breaking Down the Differences for Business Leaders [Электронный ресурс]: URL: <https://jaydevs.com/kotlin-vs-java/> (дата обращения 17.05.2024).

3. СРАВНЕНИЕ СОВРЕМЕННЫХ СРЕДСТВ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ / Чуриков Е.А., Зудилова Т.В., Ананченко И.В., Осипов Н.А., Иванов С.Е., Осетрова И.С. // ФГАОУ ВО «Национальный исследовательский университет ИТМО», Санкт-Петербург, 2022.

4. О MapKit | Mapkit Mobile SDK [Электронный ресурс]: URL: <https://yandex.ru/dev/mapkit/doc/ru/> (дата обращения 17.05.2024).

5. Документация | Firebase Documentation [Электронный ресурс]: URL: <https://firebase.google.com/docs> (дата обращения 17.05.2024).

6. Постановление государственного санитарного врача РФ от 00т 28.01.2021 N 2 (ред. от 30.12.2022) СанПиН 2.2.4.3359-16 "Санитарно-эпидемиологические требования к физическим факторам на рабочих местах".

					<i>ПГУ 1.090501.06.001 ПЗ</i>	Лист
						63
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А

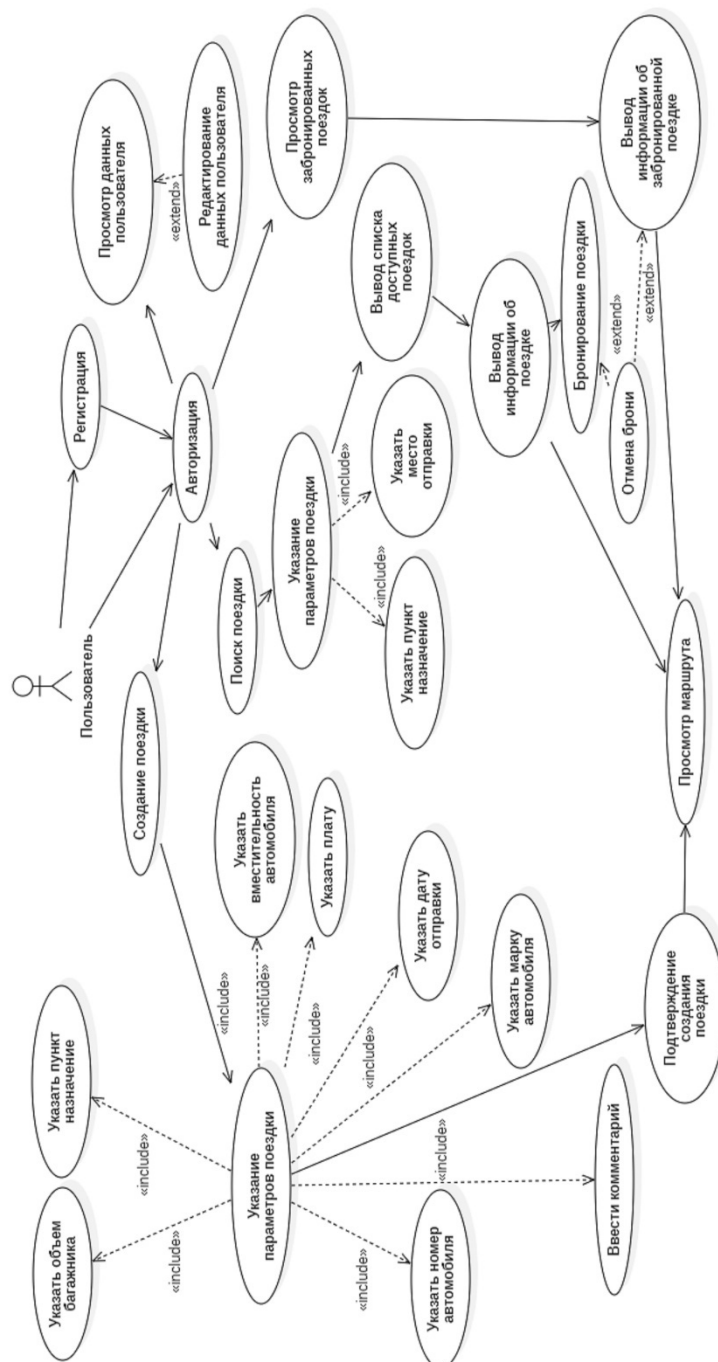
(обязательное)

Программа оптимизации логистики передвижения

UML-диаграммы

					<i>ПГУ 1.090501.06.001 ПЗ</i>	Лист
						64
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1 – Диаграмма вариантов использования

[illegible]

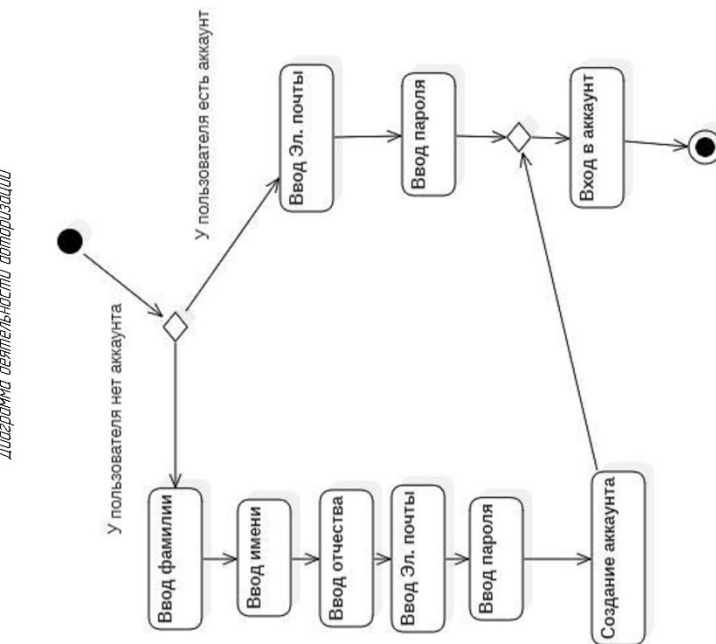
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.06.001 ПЗ

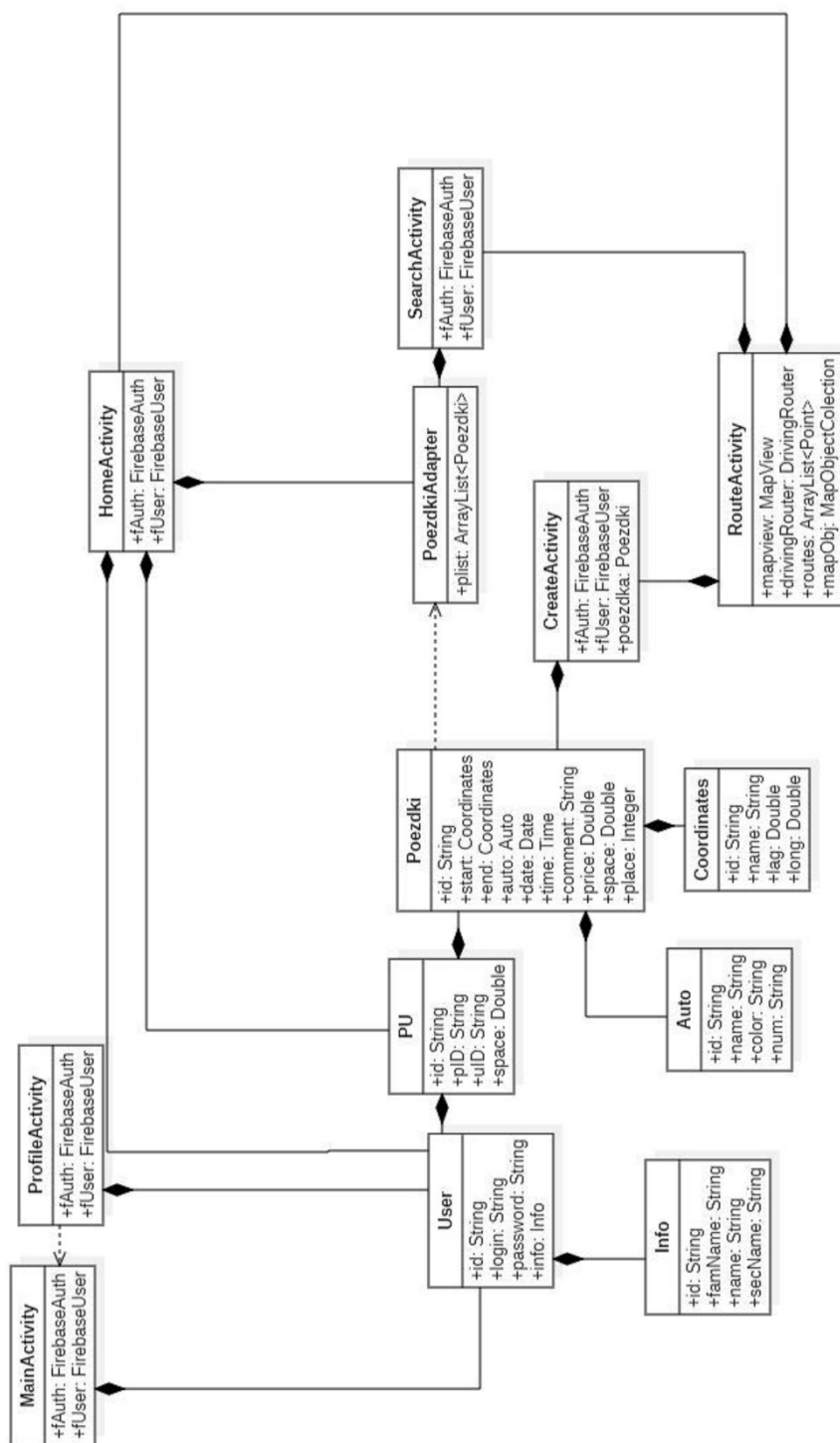
Приложение А.2 – Диаграмма последовательности

[illegible]

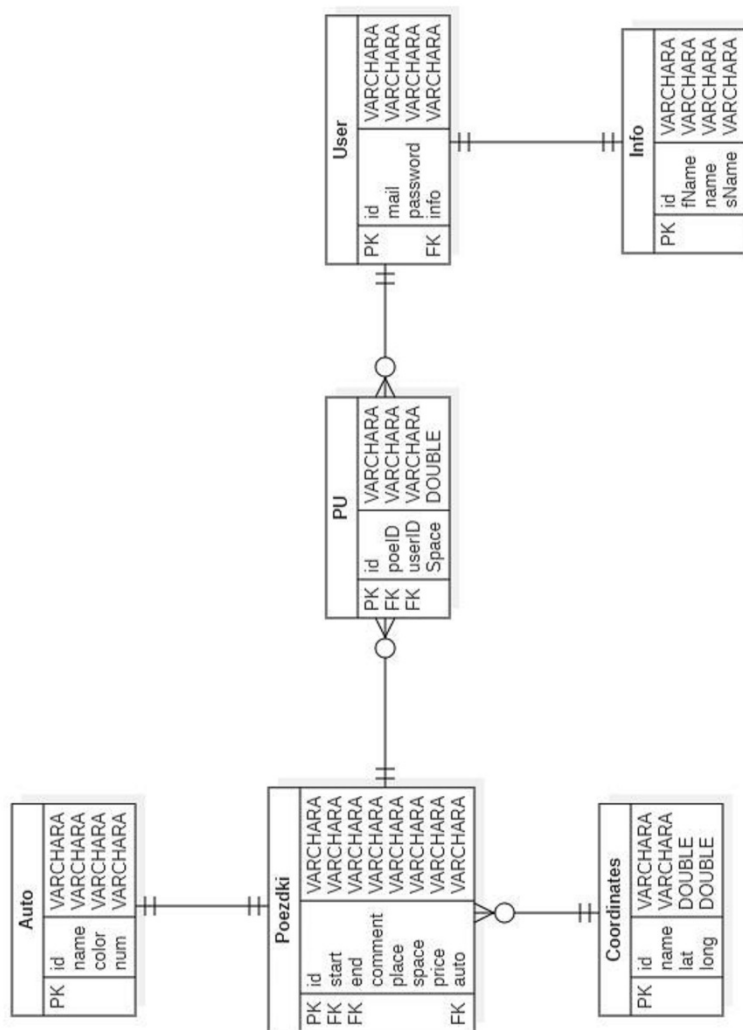
Приложение А.3 – Диаграммы деятельности

[illegible]

Приложение А.4 – Диаграмма классов

[illegible]

Приложение А.5 – Диаграмма данных

[illegible]

Приложение Б

(обязательное)

Программа оптимизации логистики передвижения

Листинг программы

					<i>ПГУ 1.090501.06.001 ПЗ</i>	Лист
						70
Изм.	Лист	№ докум.	Подпись	Дата		

MainActivity.java

```
package com.example.poputchik;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;

import com.google.firebase.FirebaseApp;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        FirebaseApp.initializeApp(getApplicationContext());

        Context cnt = getApplicationContext();
        Intent i = new Intent(cnt, Profile.class);
        startActivity(i);
    }
}
```

Profile.java

```
package com.example.poputchik;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class Profile extends AppCompatActivity {
```

					<i>ПГУ 1.090501.06.001 ПЗ</i>	Лист
						71
Изм.	Лист	№ докум.	Подпись	Дата		

```

private FirebaseAuth mAuth;
FirebaseUser user;
FirebaseDatabase database;
DatabaseReference myRef;

String userName;
String userFName;
String userSName;
String userMail;

Context cnt;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    mAuth = FirebaseAuth.getInstance();
    database = FirebaseDatabase.getInstance("https://diplom-
854a9-default-rtdb.europe-west1.firebaseio.com/");
    myRef = database.getReference();
    user = mAuth.getCurrentUser();
    if ((user != null)) {
        getUserInfo();
    }
    else authActivity();
}

public void regActivity() {
    setContentView(R.layout.activity_reg);

    Button btnReg = findViewById(R.id.button5);
    btnReg.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String email = ((EditText)
findViewById(R.id.editTextTextEmailAddress)).getText().toString();
            String password = ((EditText)
findViewById(R.id.editTextTextPassword)).getText().toString();
            String fName = ((EditText)
findViewById(R.id.idFN)).getText().toString();
            String name = ((EditText)
findViewById(R.id.idName)).getText().toString();
            String sName = ((EditText)
findViewById(R.id.idSN)).getText().toString();
            if (!email.equals("") && !password.equals("") &&
!fName.equals("") && !name.equals("") && !sName.equals("")){
                createAccount(fName, name, sName, email,
password);
            }
        }
    });
}
}

```

```

    }

    private void createAccount(String fName, String name, String
sName, String email, String password) {
        Log.d("create", "createAccount:" + email);
        mAuth.createUserWithEmailAndPassword(email,
password).addOnCompleteListener(Profile.this, task -> {
            if (task.isSuccessful()) {
                Log.d("create", "createUserWithEmail:success");
                user = mAuth.getCurrentUser();
                database =
FirebaseDatabase.getInstance("https://diplom-854a9-default-
rttdb.europe-west1.firebaseio.com/app/");
                DatabaseReference myRef =
database.getReference("User/" + user.getId() + "/name");
                myRef.setValue(name);
                myRef = database.getReference("User/" +
user.getId() + "/fName");
                myRef.setValue(fName);
                myRef = database.getReference("User/" +
user.getId() + "/sName");
                myRef.setValue(sName);
                myRef = database.getReference("User/" +
user.getId() + "/eMail");
                myRef.setValue(email);
                myRef = database.getReference("User/" +
user.getId() + "/password");
                myRef.setValue(password);
                getUserInfo();
            } else {
                Log.d("create", "createUserWithEmail:abortively");
            }
        });
    }

    private void authActivity() {

        setContentView(R.layout.activity_autor);
        cnt = getApplicationContext();

        Button btnLog = findViewById(R.id.button2);
        btnLog.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String email = ((EditText)
findViewById(R.id.idLogin)).getText().toString();
                String password = ((EditText)
findViewById(R.id.idPassword)).getText().toString();
                if (!email.equals("") && !password.equals("")){
                    signIn(email, password);
                }
            }
        }
    }

```

```

    });
    Button btnReg = findViewById(R.id.button3);
    btnReg.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            regActivity();
        }
    });
}

private void signIn(String email, String password) {
    Log.d("sign", "signIn:" + email);
    mAuth.signInWithEmailAndPassword(email,
password).addOnCompleteListener(Profile.this, new
OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                getUserInfo();
                // Sign in success, update UI with the signed-in
user's information
                //Log.d(TAG, "signInWithEmail:success");
            } else {
                Toast.makeText(cnt, "Authentication failed.",
Toast.LENGTH_SHORT).show();
                getUserInfo();
                // If sign in fails, display a message to the
user.
                //Log.w(TAG, "signInWithEmail:failure",
task.getException());
            }
        }
    });
}

private void getUserInfo() {
    user = mAuth.getCurrentUser();
    FirebaseDatabase.getInstance("https://diplom-854a9-default-
rtdb.europe-
west1.firebaseio.com").getReference().child("User").child(use
r.getId()).addListenerForSingleValueEvent(new ValueEventListener()
{
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            // Get user information
            userName =
dataSnapshot.child("name").getValue(String.class);
            userEmail =
dataSnapshot.child("eMail").getValue(String.class);
            userFName =
dataSnapshot.child("fName").getValue(String.class);
            userSName =
dataSnapshot.child("sName").getValue(String.class);

```

```

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

private void profActivity() {
    setContentView(R.layout.activity_profile);
    cnt = getApplicationContext();
    EditText EName = (EditText) findViewById(R.id.idName);
    EditText EFN = (EditText) findViewById(R.id.idFN);
    EditText ESN = (EditText) findViewById(R.id.idSN);
    EditText Email =(EditText) findViewById(R.id.idMail);
    EName.setText(userName);
    EFN.setText(userFName);
    ESN.setText(userSName);
    Email.setText(userMail);

    Button bcar = findViewById(R.id.bcarp);
    bcar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent i = new Intent(cnt, Search.class);
            i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
            startActivity(i);
        }
    });
    Button bhome = findViewById(R.id.bhomep);
    bhome.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent i = new Intent(cnt, Home.class);
            i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
            startActivity(i);
        }
    });

    Button bedit = findViewById(R.id.buttonEdit);
    bedit.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            EditText EName = (EditText)
findViewById(R.id.idName);
            EName.setEnabled(true);

            EName.setBackgroundColor(Color.parseColor("#a8b5a6"));
            EditText EFN = (EditText) findViewById(R.id.idFN);
            EFN.setEnabled(true);

```

```

        EFN.setBackgroundColor(Color.parseColor("#a8b5a6"));
        EditText ESN = (EditText) findViewById(R.id.idSN);
        ESN.setEnabled(true);
        ESN.setBackgroundColor(Color.parseColor("#a8b5a6"));
        Button bedit2 = findViewById(R.id.buttonEdit);
        bedit2.setVisibility(View.INVISIBLE);
        Button bsave2 = findViewById(R.id.button4);
        bsave2.setVisibility(View.VISIBLE);
    }
});

Button bsave = findViewById(R.id.button4);
bsave.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        EditText EName = (EditText)
findViewById(R.id.idName);
        EName.setEnabled(false);

        EName.setBackgroundColor(Color.parseColor("#FFFFFF"));
        EditText EFN = (EditText) findViewById(R.id.idFN);
        EFN.setEnabled(false);
        EFN.setBackgroundColor(Color.parseColor("#FFFFFF"));
        EditText ESN = (EditText) findViewById(R.id.idSN);
        ESN.setEnabled(false);
        ESN.setBackgroundColor(Color.parseColor("#FFFFFF"));
        Button bedit2 = findViewById(R.id.buttonEdit);
        bedit2.setVisibility(View.VISIBLE);
        Button bsave2 = findViewById(R.id.button4);
        bsave2.setVisibility(View.INVISIBLE);
        userName = EName.getText().toString();
        userFName = EFN.getText().toString();
        userSName = ESN.getText().toString();
        database =
FirebaseDatabase.getInstance("https://diplom-854a9-default-
rttdb.europe-west1.firebaseio.com/");
        DatabaseReference myRef =
database.getReference("User/" + user.getId() + "/name");
        myRef.setValue(userName);
        myRef = database.getReference("User/" +
user.getId() + "/fName");        //set path to get the multiple
dropdown nodes
        myRef.setValue(userFName);
        myRef = database.getReference("User/" +
user.getId() + "/sName");        //set path to get the multiple
dropdown nodes
        myRef.setValue(userSName);
    }
});

Button bexit = findViewById(R.id.buttonExit);

```

```

        bexit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                mAuth.signOut();
                user = null;
                finishAffinity();
            }
        });
    }
}

Home.java
package com.example.poputchik;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.annotation.SuppressLint;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.Objects;

public class Home extends AppCompatActivity {

    FirebaseAuth mAuth;
    FirebaseUser user;
    FirebaseDatabase database;
    DatabaseReference myRef;
    private MyAdapter adapter;
    private Context cnt;
    private ArrayList<Poezdki> poezdkis = new ArrayList<>();
    String NFS = "";
    int i = 0;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        cnt = getApplicationContext();

```

```

        mAuth = FirebaseAuth.getInstance();
        user = mAuth.getCurrentUser();
        mainHome();
    }

    private void mainHome() {
        setContentView(R.layout.activity_home);
        adapter = new MyAdapter(cnt, poezdkis);
        ListView listView = findViewById(R.id.ListP);
        listView.setAdapter(adapter);
        listView.setOnItemClickListener((parent, view, position, id)
-> {
            Poezdk p = poezdkis.get(position);
            String pID = p.ID;
            resultActivity(pID);
        });
        i++;
        if (i == 1) searchPoezdk();

        Button bcar = findViewById(R.id.bcar);
        bcar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(cnt, Search.class);
                i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
                startActivity(i);
            }
        });
        Button buser = findViewById(R.id.buser);
        buser.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(cnt, Profile.class);
                i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
                startActivity(i);
            }
        });
    }

    private void searchPoezdk() {
        FirebaseDatabase.getInstance("https://diplom-854a9-default-
rttdb.europe-
west1.firebaseio.com/").getReference().child("PU").addListener
ForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot)
{
                for(DataSnapshot poe: snapshot.getChildren()){

```

```

        String us =
poe.child("u").getValue(String.class);
        String po =
poe.child("p").getValue(String.class);
        if (Objects.equals(us, user.getUid()))
resPo(po);
    }
}

@Override
public void onCancelled(@NonNull DatabaseError error) {

}

});
}

private void resPo(String po) {
    FirebaseDatabase.getInstance("https://diplom-854a9-default-
rtddb.europe-
west1.firebaseio.app/").getReference().child("Poezdki").child(
po).addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot)
{
            String newID = snapshot.getKey();
            String newStart =
snapshot.child("start").getValue(String.class);
            String newEnd =
snapshot.child("end").getValue(String.class);
            String newPrice =
snapshot.child("price").getValue(String.class);
            String newPlace =
snapshot.child("place").getValue(String.class);
            String newSpace =
snapshot.child("space").getValue(String.class);
            String newDate =
snapshot.child("date").getValue(String.class);
            String newTime =
snapshot.child("time").getValue(String.class);
            Poezdki newP = new
Poezdki(newID,newStart,newEnd,newPrice,newPlace,newSpace,newDate,new
Time);

            setList(newP);
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }

    });
}

```

```

private void resultActivity(String pID) {
    setContentView(R.layout.activity_result);
    TextView nS = (TextView) findViewById(R.id.idStart3);
    TextView nE = (TextView) findViewById(R.id.idEnd3);
    TextView nDT = (TextView) findViewById(R.id.idDateTime);
    TextView nPL = (TextView) findViewById(R.id.idPlaceR);
    TextView nSP = (TextView) findViewById(R.id.idSpaceR);
    TextView nPR = (TextView) findViewById(R.id.idPriceR);
    TextView nA = (TextView) findViewById(R.id.idAutoNameR);
    TextView nAC = (TextView) findViewById(R.id.idAutoColorR);
    TextView nAN = (TextView) findViewById(R.id.idAutoNumR);
    TextView nC = (TextView) findViewById(R.id.idUserR2);
    Button exit = findViewById(R.id.button17);
    Button join = findViewById(R.id.button11);
    Button build = findViewById(R.id.button15);
    Button back = findViewById(R.id.button16);
    FirebaseDatabase.getInstance("https://diplom-854a9-default-
rtddb.europe-
west1.firebaseio.com/").getReference().child("Poezdki").child(
pID).addListenerForSingleValueEvent(new ValueEventListener() {
    @SuppressWarnings("SetTextI18n")
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot)
{
    nS.setText(snapshot.child("start").getValue(String.class));

    nE.setText(snapshot.child("end").getValue(String.class));

    nDT.setText(snapshot.child("date").getValue(String.class) + " " +
snapshot.child("time").getValue(String.class));
    nPL.setText("Кол-во свободных мест: " +
snapshot.child("place").getValue(String.class));
    nSP.setText("Свободный объем багажника: " +
snapshot.child("space").getValue(String.class));
    nPR.setText("Цена за пассажира: " +
snapshot.child("price").getValue(String.class));
    nA.setText("Карта: " +
snapshot.child("autoname").getValue(String.class));
    nAC.setText("Цвет: " +
snapshot.child("autocolor").getValue(String.class));
    nAN.setText("Номер: " +
snapshot.child("autonum").getValue(String.class));

    getUsername(snapshot.child("user").getValue(String.class));

    nC.setText(snapshot.child("com").getValue(String.class));
    String TU =
snapshot.child("user").getValue(String.class);
    String fPlace =
snapshot.child("place").getValue(String.class);
    buttonCheck(pID, TU, fPlace);
}
}

```

```

    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }

});
join.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        FirebaseDatabase.getInstance("https://diplom-854a9-
default-rtdb.europe-
west1.firebaseio.com/").getReference().child("Poezdki").child(
pID).addListenerForSingleValueEvent(new ValueEventListener() {
        @SuppressWarnings("SetTextI18n")
        @Override
        public void onDataChange(@NonNull DataSnapshot
snapshot) {
            String newPl =
snapshot.child("place").getValue(String.class);
            String newSp =
snapshot.child("space").getValue(String.class);
            joinActivity(pID,newPl, newSp);
        }

        @Override
        public void onCancelled(@NonNull DatabaseError
error) {

        }

    });
}

});
exit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        FirebaseDatabase.getInstance("https://diplom-854a9-
default-rtdb.europe-
west1.firebaseio.com/").getReference().child("PU").addListener
ForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot
snapshot) {
            for (DataSnapshot puTemp:
snapshot.getChildren()) {
                String nP =
puTemp.child("p").getValue(String.class);
                String nU =
puTemp.child("u").getValue(String.class);
                String nW =
puTemp.child("w").getValue(String.class);

```

```

                                if (Objects.equals(nP, pID) &&
Objects.equals(nU, user.getUid())){
                                    changeSP(nP,nW,puTemp.getKey());
                                }
                            }
                        }

@Override
public void onCancelled(@NonNull DatabaseError
error) {

    }

});

    }

});
build.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent i = new Intent(Home.this, build.class);
        i.putExtra("ID",pID);
        i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
        startActivity(i);
    }
});
back.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mainHome();
    }
});

}

private void changeSP(String nP, String nW,String cpID) {
    FirebaseDatabase.getInstance("https://diplom-854a9-default-
rtadb.europe-
west1.firebaseio.com/").getReference().child("Poezdki").child(
nP).addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot)
{
        String changeP =
snapshot.child("place").getValue(String.class);
        String changeS =
snapshot.child("space").getValue(String.class);

        Integer newS = Integer.valueOf(changeS) +
Integer.valueOf(nW);
        Integer newP = Integer.valueOf(changeP) +1;

```

```

        String strP = String.valueOf(newP);
        String strS = String.valueOf(newS);
        lastChange(strP, strS, nP ,cpID);

    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }

    });
}

private void lastChange(String strP, String strS, String nP,
String cpID) {
    database = FirebaseDatabase.getInstance("https://diplom-
854a9-default-rtdb.europe-west1.firebaseio.com/");
    DatabaseReference myRef = database.getReference("Poezdki/" +
nP + "/place");
    myRef.setValue(strP);
    myRef = database.getReference("Poezdki/" + nP + "/space");
    myRef.setValue(strS);

    DatabaseReference myRef2 = database.getReference("PU/" +
cpID + "/p");
    myRef2.setValue(" ");
    myRef2 = database.getReference("PU/" + cpID + "/u");
    myRef2.setValue(" ");
    finish();
}

private void joinActivity(String pID,String pl, String sp) {
    setContentView(R.layout.activity_join);
    TextView nW = (TextView) findViewById(R.id.editTextNumber3);
    Button back = findViewById(R.id.b5);
    back.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            resultActivity(pID);
        }
    });
    Button join = findViewById(R.id.button18);
    join.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String weight = nW.getText().toString();
            if (weight.equals("")) weight = "0";
            if (Integer.valueOf(sp)>Integer.valueOf(weight)) {
                database =
FirebaseDatabase.getInstance("https://diplom-854a9-default-
rtdb.europe-west1.firebaseio.com/");

```

```

        DatabaseReference myRef =
database.getReference("PU/");
        String puID = myRef.push().getKey();
        myRef = database.getReference("PU/" + puID +
"/u");
        myRef.setValue(user.getUid());
        myRef = database.getReference("PU/" + puID +
"/p");
        myRef.setValue(pID);
        myRef = database.getReference("PU/" + puID +
"/w");
        myRef.setValue(weight);
        Integer i = Integer.valueOf(pl);
        String npl = String.valueOf(i - 1);
        i = Integer.valueOf(sp);
        Integer j = Integer.valueOf(weight);
        String nsp = String.valueOf(i - j);
        database =
FirebaseDatabase.getInstance("https://diplom-854a9-default-
rtdb.europe-west1.firebaseio.app/");
        DatabaseReference myRef3 =
database.getReference("Poezdki/" + pID + "/place");
        myRef3.setValue(npl);
        myRef3 = database.getReference("Poezdki/" + pID
+ "/space");
        myRef3.setValue(nsp);
        finish();
    }
}
});
}

private void buttonCheck(String pID, String userTemp, String fp)
{
    Button exit = findViewById(R.id.button17);
    Button join = findViewById(R.id.button11);
    exit.setVisibility(View.INVISIBLE);
    FirebaseDatabase.getInstance("https://diplom-854a9-default-
rtdb.europe-
west1.firebaseio.app/").getReference().child("PU").addListener
ForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot)
        {
            for (DataSnapshot puTemp: snapshot.getChildren()){
                String nP =
puTemp.child("p").getValue(String.class);
                String nU =
puTemp.child("u").getValue(String.class);
                boolean flag = false;
                if (Objects.equals(nU, user.getUid()) &&
(Objects.equals(nP, pID))) {

```

```

        flag = true;
    }
    if (flag || (Objects.equals(user.getId(),
userTemp))) {
        join.setVisibility(View.INVISIBLE);
        exit.setVisibility(View.VISIBLE);
    }
    if (Objects.equals(user.getId(), userTemp) ||
!flag) {
        exit.setVisibility(View.INVISIBLE);
    }
    if (Objects.equals(fp, "0")) {
        join.setVisibility(View.INVISIBLE);
    }
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {
    }

    });
}

private void getUsername(String upoe) {
    FirebaseDatabase.getInstance("https://diplom-854a9-default-
rtadb.europe-
west1.firebaseio.com/").getReference().child("User").child(upoe)
.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // Get user information

        String userName =
dataSnapshot.child("name").getValue(String.class);
        String userFName =
dataSnapshot.child("fName").getValue(String.class);
        String userSName =
dataSnapshot.child("sName").getValue(String.class);
        NFS = userFName + " " + userName.charAt(0) + ". " +
userSName.charAt(0) + ".";
        TextView nU = (TextView) findViewById(R.id.idUserR);
        nU.setText("Водитель: " + NFS);
    }
    @Override
    public void onCancelled(DatabaseError databaseError) { }
    });
}

public void setList(Poezdki p) {
    adapter.AddPoezd(p);
}

```

```

        adapter.notifyDataSetChanged();
    }
}
Search.java
package com.example.poputchik;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.annotation.SuppressLint;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.Objects;

public class Search extends AppCompatActivity {

    FirebaseAuth mAuth;
    FirebaseUser user;
    FirebaseDatabase database;
    DatabaseReference myRef;
    private MyAdapter adapter;
    private Context cnt;
    private ArrayList<Poezdki> poezdkis = new ArrayList<>();
    AutoCompleteTextView autoTextView;
    AutoCompleteTextView autoTextView2;
    String start = "";
    String end = "";
    String NFS = "";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        cnt = getApplicationContext();
        mAuth = FirebaseAuth.getInstance();

```

```

        user = mAuth.getCurrentUser();
        mainSearch();
    }

    private void mainSearch() {
        setContentView(R.layout.activity_search);
        autoTextView = (AutoCompleteTextView)
        findViewById(R.id.editTextTextPersonName);
        autoTextView2 = (AutoCompleteTextView)
        findViewById(R.id.editTextTextPersonName2);
        ArrayAdapter<String> arrayAdapter = new
        ArrayAdapter<String>(this, android.R.layout.select_dialog_item,
        cityName);
        autoTextView.setThreshold(1);
        autoTextView2.setThreshold(1);
        autoTextView.setAdapter(arrayAdapter);
        autoTextView2.setAdapter(arrayAdapter);
        autoTextView.setText(start);
        autoTextView2.setText(end);

        adapter = new MyAdapter(cnt, poezdkis);
        ListView listView = findViewById(R.id.listView);
        listView.setAdapter(adapter);
        listView.setOnItemClickListener((parent, view, position, id)
-> {
            Poezdki p = poezdkis.get(position);
            String pID = p.ID;
            resultActivity(pID);
        });

        Button btn = findViewById(R.id.btn);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                start = autoTextView.getText().toString();
                end = autoTextView2.getText().toString();
                if (((!Objects.equals(start, "")) &&
(!end.equals(""))))) {
                    searchPoezdki();
                }
            }
        });

        Button buser = findViewById(R.id.busers);
        buser.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(cnt, Profile.class);
                i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
                startActivity(i);
            }
        });
    }

```

```

    }
});
Button bhome = findViewById(R.id.bhomes);
bhome.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent i = new Intent(cnt, Home.class);
        i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
        startActivity(i);
    }
});

Button bcreate = findViewById(R.id.button);
bcreate.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent i = new Intent(cnt, create1.class);
        i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
        startActivity(i);
    }
});
}

private void resultActivity(String pID) {
    setContentView(R.layout.activity_result);
    TextView nS = (TextView) findViewById(R.id.idStart3);
    TextView nE = (TextView) findViewById(R.id.idEnd3);
    TextView nDT = (TextView) findViewById(R.id.idDateTime);
    TextView nPL = (TextView) findViewById(R.id.idPlaceR);
    TextView nSP = (TextView) findViewById(R.id.idSpaceR);
    TextView nPR = (TextView) findViewById(R.id.idPriceR);
    TextView nA = (TextView) findViewById(R.id.idAutoNameR);
    TextView nAC = (TextView) findViewById(R.id.idAutoColorR);
    TextView nAN = (TextView) findViewById(R.id.idAutoNumR);
    TextView nC = (TextView) findViewById(R.id.idUserR2);
    Button exit = findViewById(R.id.button17);
    Button join = findViewById(R.id.button11);
    Button build = findViewById(R.id.button15);
    Button back = findViewById(R.id.button16);
    FirebaseDatabase.getInstance("https://diplom-854a9-default-
rtddb.europe-
west1.firebaseio.com/").getReference().child("Poezdki").child(
pID).addListenerForSingleValueEvent(new ValueEventListener() {
        @SuppressWarnings("SetTextI18n")
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot)
{
nS.setText(snapshot.child("start").getValue(String.class));

```

```

nE.setText(snapshot.child("end").getValue(String.class));

nDT.setText(snapshot.child("date").getValue(String.class) + " " +
snapshot.child("time").getValue(String.class));
    nPL.setText("Кол-во свободных мест: " +
snapshot.child("place").getValue(String.class));
    nSP.setText("Свободный объем багажника: " +
snapshot.child("space").getValue(String.class));
    nPR.setText("Цена за пассажира: " +
snapshot.child("price").getValue(String.class));
    nA.setText("Карта: " +
snapshot.child("autoname").getValue(String.class));
    nAC.setText("Цвет: " +
snapshot.child("autocolor").getValue(String.class));
    nAN.setText("Homep: " +
snapshot.child("autonum").getValue(String.class));

getUserName(snapshot.child("user").getValue(String.class));

nC.setText(snapshot.child("com").getValue(String.class));
    String TU =
snapshot.child("user").getValue(String.class);
    String fPlace =
snapshot.child("place").getValue(String.class);
    buttonCheck(pID, TU, fPlace);
}

@Override
public void onCancelled(@NonNull DatabaseError error) {

}

});
join.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        FirebaseDatabase.getInstance("https://diplom-854a9-
default-rtdb.europe-
west1.firebaseio.com").getReference().child("Poezdki").child(
pID).addListenerForSingleValueEvent(new ValueEventListener() {
            @SuppressWarnings("SetTextI18n")
            @Override
            public void onDataChange(@NonNull DataSnapshot
snapshot) {
                String newPl =
snapshot.child("place").getValue(String.class);
                String newSp =
snapshot.child("space").getValue(String.class);
                joinActivity(pID, newPl, newSp);
            }

            @Override

```

```

        public void onCancelled(@NonNull DatabaseError
error) {

        }

    });
}

});
exit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        FirebaseDatabase.getInstance("https://diplom-854a9-
default-rtdb.europe-
west1.firebaseio.app/").getReference().child("PU").addListener
ForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot
snapshot) {
            for (DataSnapshot puTemp:
snapshot.getChildren()) {
                String nP =
puTemp.child("p").getValue(String.class);
                String nU =
puTemp.child("u").getValue(String.class);
                String nW =
puTemp.child("w").getValue(String.class);
                if (Objects.equals(nP, pID) &&
Objects.equals(nU, user.getId())) {
                    changeSP(nP, nW, puTemp.getKey());
                }
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError
error) {

        }

    });

}

});
build.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent i = new Intent(Search.this, build.class);
        i.putExtra("ID", pID);
        i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
        startActivity(i);
    }
});

```

```

        back.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mainSearch();
            }
        });

    }

    private void changeSP(String nP, String nW,String cpID) {
        FirebaseDatabase.getInstance("https://diplom-854a9-default-
rtdb.europe-
west1.firebaseio.com/").getReference().child("Poezdki").child(
nP).addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot)
{
                String changeP =
snapshot.child("place").getValue(String.class);
                String changeS =
snapshot.child("space").getValue(String.class);

                Integer newS = Integer.valueOf(changeS) +
Integer.valueOf(nW);
                Integer newP = Integer.valueOf(changeP) +1;
                String strP = String.valueOf(newP);
                String strS = String.valueOf(newS);
                lastChange(strP, strS, nP ,cpID);

            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {

            }

        });
    }

    private void lastChange(String strP, String strS, String nP,
String cpID) {
        database = FirebaseDatabase.getInstance("https://diplom-
854a9-default-rtdb.europe-west1.firebaseio.com/");
        DatabaseReference myRef = database.getReference("Poezdki/" +
nP + "/place");
        myRef.setValue(strP);
        myRef = database.getReference("Poezdki/" + nP + "/space");
        myRef.setValue(strS);

        DatabaseReference myRef2 = database.getReference("PU/" +
cpID + "/p");
        myRef2.setValue(" ");
        myRef2 = database.getReference("PU/" + cpID + "/u");
    }

```

```

        myRef2.setValue(" ");
        finish();
    }

    private void joinActivity(String pID,String pl, String sp) {
        setContentView(R.layout.activity_join);
        TextView nW = (TextView) findViewById(R.id.editTextNumber3);
        Button back = findViewById(R.id.b5);
        back.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                resultActivity(pID);
            }
        });
        Button join = findViewById(R.id.button18);
        join.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String weight = nW.getText().toString();
                if (weight.equals("")) weight = "0";
                if (Integer.valueOf(sp)>Integer.valueOf(weight)) {
                    database =
FirebaseDatabase.getInstance("https://diplom-854a9-default-
rtdb.europe-west1.firebaseio.com/");
                    DatabaseReference myRef =
database.getReference("PU/");
                    String puID = myRef.push().getKey();
                    myRef = database.getReference("PU/" + puID +
"/u");
                    myRef.setValue(user.getUid());
                    myRef = database.getReference("PU/" + puID +
"/p");
                    myRef.setValue(pID);
                    myRef = database.getReference("PU/" + puID +
"/w");
                    myRef.setValue(weight);
                    Integer i = Integer.valueOf(pl);
                    String npl = String.valueOf(i - 1);
                    i = Integer.valueOf(sp);
                    Integer j = Integer.valueOf(weight);
                    String nsp = String.valueOf(i - j);
                    database =
FirebaseDatabase.getInstance("https://diplom-854a9-default-
rtdb.europe-west1.firebaseio.com/");
                    DatabaseReference myRef3 =
database.getReference("Poezdki/" + pID + "/place");
                    myRef3.setValue(npl);
                    myRef3 = database.getReference("Poezdki/" + pID
+ "/space");
                    myRef3.setValue(nsp);
                    start = "";
                    end = "";
                }
            }
        });
    }

```

```

        finish();
    }
}

private void buttonCheck(String pID, String userTemp, String fp)
{
    Button exit = findViewById(R.id.button17);
    Button join = findViewById(R.id.button11);
    exit.setVisibility(View.INVISIBLE);
    FirebaseDatabase.getInstance("https://diplom-854a9-default-
rtddb.europe-
west1.firebaseio.com/").getReference().child("PU").addListener
ForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot)
    {
        for (DataSnapshot puTemp: snapshot.getChildren()){
            String nP =
puTemp.child("p").getValue(String.class);
            String nU =
puTemp.child("u").getValue(String.class);
            boolean flag = false;
            if (Objects.equals(nU, user.getUid()) &&
(Objects.equals(nP, pID))){
                flag = true;
            }
            if (flag || (Objects.equals(user.getUid(),
userTemp))){
                join.setVisibility(View.INVISIBLE);
                exit.setVisibility(View.VISIBLE);
            }
            if (Objects.equals(user.getUid(), userTemp) ||
!flag){
                exit.setVisibility(View.INVISIBLE);
            }
            if (Objects.equals(fp, "0")){
                join.setVisibility(View.INVISIBLE);
            }
        }
    }

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

private void getUsername(String upoe) {

```

```

        FirebaseDatabase.getInstance("https://diplom-854a9-default-
rtdb.europe-
west1.firebaseio.com/").getReference().child("User").child(upo
e).addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // Get user information

        String userName =
dataSnapshot.child("name").getValue(String.class);
        String userFName =
dataSnapshot.child("fName").getValue(String.class);
        String userSName =
dataSnapshot.child("sName").getValue(String.class);
        NFS = userFName + " " + userName.charAt(0) + ". " +
userSName.charAt(0) + ".";
        TextView nU = (TextView) findViewById(R.id.idUserR);
        nU.setText("Водитель: " + NFS);
    }
    @Override
    public void onCancelled(DatabaseError databaseError) { }
});
}

private void searchPoezdki() {
    FirebaseDatabase.getInstance("https://diplom-854a9-default-
rtdb.europe-
west1.firebaseio.com/").getReference().child("Poezdki").addLis
tenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot)
{
        for(DataSnapshot poe: snapshot.getChildren()){
            String newID = poe.getKey();
            String newStart =
poe.child("start").getValue(String.class);
            String newEnd =
poe.child("end").getValue(String.class);
            String newPrice =
poe.child("price").getValue(String.class);
            String newPlace =
poe.child("place").getValue(String.class);
            String newSpace =
poe.child("space").getValue(String.class);
            String newDate =
poe.child("date").getValue(String.class);
            String newTime =
poe.child("time").getValue(String.class);
            if (((Objects.equals(start, newStart)) &&
(Objects.equals(end, newEnd)))){

```

```

        Poezdki newP = new
Poezdki (newID,newStart,newEnd,newPrice,newPlace,newSpace,newDate,new
Time);

        setList(newP);
    }

}

@Override
public void onCancelled(@NonNull DatabaseError error) {

}

});
}

public void setList(Poezdki p){
    adapter.AddPoezd(p);
    adapter.notifyDataSetChanged();
}
}

```

Create.java

```

package com.example.poputchik;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.yandex.mapkit.MapKitFactory;
import com.yandex.mapkit.RequestPoint;
import com.yandex.mapkit.RequestPointType;
import com.yandex.mapkit.directions.DirectionsFactory;
import com.yandex.mapkit.directions.driving.DrivingOptions;
import com.yandex.mapkit.directions.driving.DrivingRoute;
import com.yandex.mapkit.directions.driving.DrivingRouter;
import com.yandex.mapkit.directions.driving.DrivingRouterType;
import com.yandex.mapkit.directions.driving.DrivingSession;
import com.yandex.mapkit.directions.driving.VehicleOptions;

```

```

import com.yandex.mapkit.geometry.Point;
import com.yandex.mapkit.map.CameraPosition;
import com.yandex.mapkit.map.MapObjectCollection;
import com.yandex.mapkit.map.PlacemarkMapObject;
import com.yandex.mapkit.mapview.MapView;
import com.yandex.mapkit.search.Session;
import com.yandex.runtime.Error;
import com.yandex.runtime.image.ImageProvider;

import java.util.ArrayList;
import java.util.List;
import java.util.Objects;

public class create1 extends AppCompatActivity{

    private Context cnt;
    AutoCompleteTextView autoTextView;
    AutoCompleteTextView autoTextView2;

    FirebaseAuth mAuth;
    FirebaseUser user;
    FirebaseDatabase database;
    DatabaseReference myRef;

    Poezdki newCreate;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        newCreate = new Poezdki();
        mAuth = FirebaseAuth.getInstance();
        user = mAuth.getCurrentUser();

        startEndActivity();
    }

    private void startEndActivity() {
        setContentView(R.layout.activity_create1);
        cnt = getApplicationContext();

        autoTextView = (AutoCompleteTextView)
        findViewById(R.id.editTextTextPersonName3);
        autoTextView2 = (AutoCompleteTextView)
        findViewById(R.id.editTextTextPersonName4);
        ArrayAdapter<String> arrayAdapter = new
        ArrayAdapter<String>(this, android.R.layout.select_dialog_item,
        cityName);
        autoTextView.setThreshold(1);
        autoTextView2.setThreshold(1);
        autoTextView.setAdapter(arrayAdapter);
    }

```

```

autoTextView2.setAdapter(arrayAdapter);
autoTextView.setText(newCreate.Start);
autoTextView2.setText(newCreate.End);

Button bcreate1 = findViewById(R.id.button6);
bcreate1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        newCreate.Start = ((EditText)
findViewById(R.id.editTextTextPersonName3)).getText().toString();
        newCreate.End = ((EditText)
findViewById(R.id.editTextTextPersonName4)).getText().toString();

        commentActivity();
    }
});
Button b1 = findViewById(R.id.b4);

b1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        finish();
    }
});

}

public void commentActivity(){
    setContentView(R.layout.activity_create2);
    cnt = getApplicationContext();
    ((EditText)
findViewById(R.id.editTextTextMultiLine)).setText(newCreate.Comment)
;

    Button bcreate2 = findViewById(R.id.button8);
    bcreate2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            newCreate.Comment = ((EditText)
findViewById(R.id.editTextTextMultiLine)).getText().toString();
            dateAndTimeActivity();
        }
    });
    Button b2 = findViewById(R.id.b3);
    b2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            startEndActivity();
        }
    });

}

public void dateAndTimeActivity(){

```

```

        setContentView(R.layout.activity_create3);
        cnt = getApplicationContext();
        ((EditText)
findViewById(R.id.editTextDate)).setText(newCreate.Date);
        ((EditText)
findViewById(R.id.editTextTime)).setText(newCreate.Time);
        Button bcreate3 = findViewById(R.id.button7);
        bcreate3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                newCreate.Date = ((EditText)
findViewById(R.id.editTextDate)).getText().toString();
                newCreate.Time = ((EditText)
findViewById(R.id.editTextTime)).getText().toString();
                carActivity();
            }
        });
        Button b3 = findViewById(R.id.b);
        b3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                commentActivity();
            }
        });
    }
    public void carActivity(){
        setContentView(R.layout.activity_create4);
        cnt = getApplicationContext();
        ((EditText)
findViewById(R.id.editTextTextPersonName6)).setText(newCreate.AutoNa
me);
        ((EditText)
findViewById(R.id.editTextTextPersonName7)).setText(newCreate.AutoNu
mber);
        ((EditText)
findViewById(R.id.editTextTextPersonName8)).setText(newCreate.AutoCo
lor);
        ((EditText)
findViewById(R.id.editTextNumber)).setText(newCreate.Place);
        ((EditText)
findViewById(R.id.editTextNumber2)).setText(newCreate.Space);
        Button bcreate4 = findViewById(R.id.button9);
        bcreate4.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                newCreate.AutoName = ((EditText)
findViewById(R.id.editTextTextPersonName6)).getText().toString();
                newCreate.AutoNumber = ((EditText)
findViewById(R.id.editTextTextPersonName7)).getText().toString();
                newCreate.AutoColor = ((EditText)
findViewById(R.id.editTextTextPersonName8)).getText().toString();
            }
        });
    }
}

```

```

        newCreate.Place = ((EditText)
findViewById(R.id.editTextNumber)).getText().toString();
        newCreate.Space = ((EditText)
findViewById(R.id.editTextNumber2)).getText().toString();
        priceActivity();
    }
});
Button b4 = findViewById(R.id.b2);
b4.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        dateAndTimeActivity();
    }
});
}

private void priceActivity() {
    setContentView(R.layout.activity_create6);
    cnt = getApplicationContext();
    ((EditText)
findViewById(R.id.editTextNumber4)).setText(newCreate.Price);
    Button bcreate6 = findViewById(R.id.button13);
    bcreate6.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            newCreate.Price = ((EditText)
findViewById(R.id.editTextNumber4)).getText().toString();
            confirmActivity();
        }
    });
    Button b6 = findViewById(R.id.b15);
    b6.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            carActivity();
        }
    });
}

private void confirmActivity() {
    setContentView(R.layout.activity_create7);
    cnt = getApplicationContext();
    TextView startT = (TextView) findViewById(R.id.textView25);
    startT.setText("Город выезда: " + newCreate.Start);
    TextView endT = (TextView) findViewById(R.id.textView26);
    endT.setText("Город приезда: " + newCreate.End);
    TextView dateT = (TextView) findViewById(R.id.textView27);
    dateT.setText("Дата выезда: " + newCreate.Date);
    TextView timeT = (TextView) findViewById(R.id.textView28);
    timeT.setText("Время выезда: " + newCreate.Time);
    TextView anT = (TextView) findViewById(R.id.textView29);
    anT.setText("Карта: " + newCreate.AutoName);
}

```

```

        TextView acT = (TextView) findViewById(R.id.textView30);
        acT.setText("Цвет: " + newCreate.AutoColor);
        TextView anumT = (TextView) findViewById(R.id.textView31);
        anumT.setText("Homep: " + newCreate.AutoNumber);
        TextView placeT = (TextView) findViewById(R.id.textView32);
        placeT.setText("Кол-во мест: " + newCreate.Place);
        TextView spaceT = (TextView) findViewById(R.id.textView33);
        spaceT.setText("Объём баража: " + newCreate.Space + "л");
        TextView priceT = (TextView) findViewById(R.id.textView34);
        priceT.setText("Цена: " + newCreate.Price + "p");
        TextView comT = (TextView) findViewById(R.id.textView35);
        comT.setText(newCreate.Comment);

        Button bcreate7 = findViewById(R.id.button12);
        bcreate7.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                database =
                FirebaseDatabase.getInstance("https://diplom-854a9-default-
                rtdb.europe-west1.firebaseio.com/");
                DatabaseReference myRef =
                database.getReference("Poezdki/");
                String poezdkiID = myRef.push().getKey();
                myRef = database.getReference("Poezdki/" + poezdkiID
+ "/user");
                myRef.setValue(user.getId());
                myRef = database.getReference("Poezdki/" + poezdkiID
+ "/start");
                myRef.setValue(newCreate.Start);
                myRef = database.getReference("Poezdki/" + poezdkiID
+ "/end");
                myRef.setValue(newCreate.End);
                myRef = database.getReference("Poezdki/" + poezdkiID
+ "/date");
                myRef.setValue(newCreate.Date);
                myRef = database.getReference("Poezdki/" + poezdkiID
+ "/time");
                myRef.setValue(newCreate.Time);
                myRef = database.getReference("Poezdki/" + poezdkiID
+ "/autoname");
                myRef.setValue(newCreate.AutoName);
                myRef = database.getReference("Poezdki/" + poezdkiID
+ "/autocolor");
                myRef.setValue(newCreate.AutoColor);
                myRef = database.getReference("Poezdki/" + poezdkiID
+ "/autonum");
                myRef.setValue(newCreate.AutoNumber);
                myRef = database.getReference("Poezdki/" + poezdkiID
+ "/place");
                myRef.setValue(newCreate.Place);
                myRef = database.getReference("Poezdki/" + poezdkiID
+ "/space");
            }
        });
    }
}

```

```

        myRef.setValue(newCreate.Space);
        myRef = database.getReference("Poezdki/" + poezdkiID
+ "/price");
        myRef.setValue(newCreate.Price);
        myRef = database.getReference("Poezdki/" + poezdkiID
+ "/com");
        myRef.setValue(newCreate.Comment);

        DatabaseReference myRef2 =
database.getReference("PU/");
        String puID = myRef2.push().getKey();
        myRef2 = database.getReference("PU/" + puID + "/u");
        myRef2.setValue(user.getUid());
        myRef2 = database.getReference("PU/" + puID + "/p");
        myRef2.setValue(poezdkiID);

        Intent i = new Intent(cnt, create5.class);

        i.putExtra("ID", poezdkiID);

        i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
        startActivity(i);
    }
});
Button b7 = findViewById(R.id.button14);
b7.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        priceActivity();
    }
});
}
}

```

}

Route.java

package com.example.poputchik;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

					ПГУ 1.090501.06.001 ПЗ	Лист
						101
Изм.	Лист	№ докум.	Подпись	Дата		

```

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.yandex.mapkit.MapKitFactory;
import com.yandex.mapkit.RequestPoint;
import com.yandex.mapkit.RequestPointType;
import com.yandex.mapkit.ScreenPoint;
import com.yandex.mapkit.ScreenRect;
import com.yandex.mapkit.directions.DirectionsFactory;
import com.yandex.mapkit.directions.driving.DrivingOptions;
import com.yandex.mapkit.directions.driving.DrivingRoute;
import com.yandex.mapkit.directions.driving.DrivingRouter;
import com.yandex.mapkit.directions.driving.DrivingRouterType;
import com.yandex.mapkit.directions.driving.DrivingSession;
import com.yandex.mapkit.directions.driving.VehicleOptions;
import com.yandex.mapkit.geometry.BoundingBox;
import com.yandex.mapkit.geometry.BoundingBoxHelper;
import com.yandex.mapkit.geometry.Geometry;
import com.yandex.mapkit.geometry.Point;
import com.yandex.mapkit.map.CameraPosition;
import com.yandex.mapkit.map.MapObjectCollection;
import com.yandex.mapkit.map.PlacemarkMapObject;
import com.yandex.mapkit.mapview.MapView;
import com.yandex.mapkit.search.Session;
import com.yandex.runtime.Error;
import com.yandex.runtime.image.ImageProvider;

import java.util.ArrayList;
import java.util.List;
import java.util.Objects;

```

```

public class create5 extends AppCompatActivity implements
DrivingSession.DrivingRouteListener {

```

```

    class City{
        String Name;
        double lat;
        double lon;

        City(String NewName, double Newlat, double Newlon){
            this.Name = NewName;
            this.lat = Newlat;
            this.lon = Newlon;
        }
    }
    City[] CityArray = new City[82];

    private final String MAPKIT_API_KEY = "68093b0a-6ee9-45c0-a701-
d2eb2bfa6dea";

```

					ПГУ 1.090501.06.001 ПЗ	Лист
						102
Изм.	Лист	№ докум.	Подпись	Дата		

```

private MapView mapView;
private Session session;
private String StartCity = "";
private String EndCity = "";
private Point start;
private Point end;
private Point center;

private MapObjectCollection mapOb = null;
private DrivingRouter drivingRouter = null;
private DrivingSession drivingSession = null;
private Context cnt;

CameraPosition boundingBoxPosition;
PlacemarkMapObject slock;
PlacemarkMapObject elock;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    String poezdID = (String)getIntent().getStringExtra("ID");
    FirebaseDatabase.getInstance("https://diplom-854a9-default-
rtdb.europe-
west1.firebaseio.com/").getReference().child("Poezdki").child(
poezdID).addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot)
    {
        StartCity =
snapshot.child("start").getValue(String.class);
        EndCity =
snapshot.child("end").getValue(String.class);
        pointCreate();
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }
});
CreateMass();

MapKitFactory.setApiKey(MAPKIT_API_KEY);
MapKitFactory.initialize(this);
setContentView(R.layout.activity_create5);
mapView = findViewById(R.id.mapview);

```

```

        drivingRouter =
DirectionsFactory.getInstance().createDrivingRouter(DrivingRouterTyp
e.COMBINED);
        mapOb = mapView.getMap().getMapObjects().addCollection();

        Button bcreate5 = findViewById(R.id.bexit);
        bcreate5.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(create5.this, Search.class);
                i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
                startActivity(i);
            }
        });

    }

    private void pointCreate() {
        start = new
Point(getMassLat(StartCity),getMassLon(StartCity));
        end = new Point(getMassLat(EndCity),getMassLon(EndCity));
        center = new Point((start.getLatitude() +
end.getLatitude())/2, (start.getLongitude() +
end.getLongitude())/2);
        submitRequest();
    }

    @Override
    protected void onStop() {
        mapView.onStop();
        MapKitFactory.getInstance().onStop();
        super.onStop();
    }

    @Override
    protected void onStart() {
        super.onStart();
        MapKitFactory.getInstance().onStart();
        mapView.onStart();
    }

    @Override
    public void onDrivingRoutes(@NonNull List<DrivingRoute> routes)
{

        if (routes != null && !routes.isEmpty()) {
            DrivingRoute route = routes.get(0);

```

```

        mapOb.addPolyline(route.getGeometry());
        mapView.getMapWindow().setFocusRect(new ScreenRect(new
ScreenPoint(100f, 100f), new
ScreenPoint(mapView.getMapWindow().width() - 100f,
mapView.getMapWindow().height() - 100f)));
        BoundingBox box =
BoundingBoxHelper.getBounds(route.getGeometry());
        boundingBoxPosition =
mapView.getMap().cameraPosition(Geometry.fromBoundingBox(box));
        mapView.getMap().move(boundingBoxPosition);
    }

}

@Override
public void onDrivingRoutesError(@NonNull Error error) {

}

private void submitRequest() {
    DrivingOptions drivingOptions = new DrivingOptions();
    VehicleOptions vehicleOptions = new VehicleOptions();
    ArrayList<RequestPoint> requestPoints = new ArrayList<>();

    requestPoints.add(new RequestPoint(start,
RequestPointType.WAYPOINT, null, null));

    requestPoints.add(new RequestPoint(end,
RequestPointType.WAYPOINT, null, null));
    drivingSession = drivingRouter.requestRoutes(requestPoints,
drivingOptions, vehicleOptions, this);
    slock = mapView.getMap().getMapObjects().addPlacemark(start,
ImageProvider.fromResource(create5.this, R.drawable.start));
    elock = mapView.getMap().getMapObjects().addPlacemark(end,
ImageProvider.fromResource(create5.this, R.drawable.end));

}

private double getMassLon(String find) {
    double num = 0;
    for (int i = 0; i<=81; i++){
        if (Objects.equals(find, CityArray[i].Name)) {
            num = CityArray[i].lon;
            break;
        }
    }
    return num;
}

private double getMassLat(String find) {
    double num = 0;

```

```

        for (int i = 0; i<=81; i++){
            if (Objects.equals(find, CityArray[i].Name)) {
                num = CityArray[i].lat;
                break;
            }
        }
        return num;
    }
}

```

Приложение В

(обязательное)

Программа оптимизации логистики передвижения

Презентация

					<i>ПГУ 1.090501.06.001 ПЗ</i>	Лист
						107
Изм.	Лист	№ докум.	Подпись	Дата		