

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

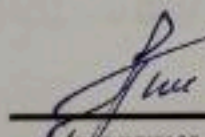
Специальность: 09.05.01 «Применение и эксплуатация
автоматизированных систем специального назначения»
Специализация: «Эксплуатация вычислительных машин, комплексов,
систем и сетей специального назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

**ПРОГРАММА ДЛЯ ТРЕКИНГА РАСПРЕДЕЛЕННОЙ
ГРУППЫ ЛЮДЕЙ**

Студент

 11.06.24
(подпись, дата)

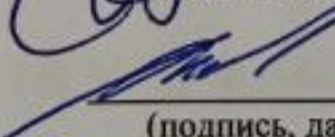
Пименова Алиса Андреевна
(ФИО полностью)

Руководитель

 11.06.24
(подпись, дата)

Федюнин Р.Н.
(фамилия, инициалы)

Нормоконтролёр

 11.06.24
(подпись, дата)

А.С. Бычков
(фамилия, инициалы)

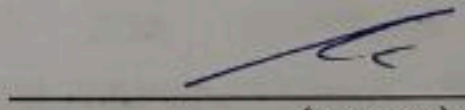
Рецензент

21.06.24
21.06.24
40, ИПП, Рубин
А.В. Васильев
(должность, место работы) (подпись, дата)

А.В. Васильев
(фамилия, инициалы)

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

Заведующий кафедрой


(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой отлично

(протокол заседания ГЭК от 19.06.2024г. № 4)

Секретарь ГЭК



Е.И. Гурин

Пенза 2024

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

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«19» апреля 2024 г.

**ЗАДАНИЕ
НА ДИПЛОМНЫЙ ПРОЕКТ**

1. Студент Пименова Алиса Андреевна гр. 19ВОЭ1 факультета вычислительной техники

2. Тема работы «Программа для трекинга распределенной группы людей»

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

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

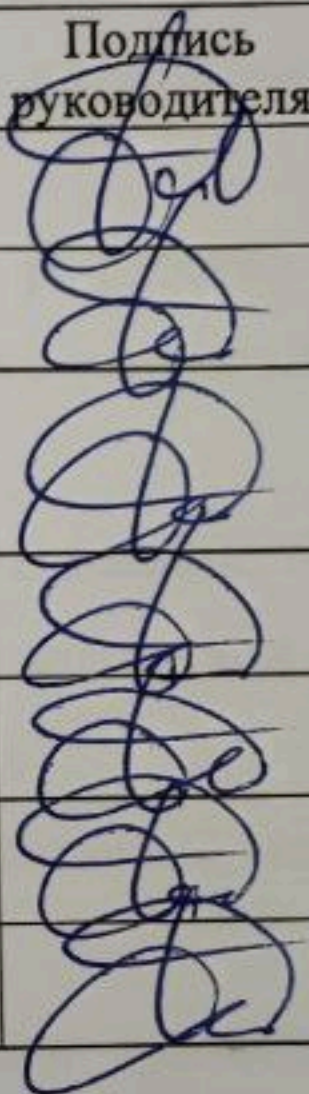
4. Задание на работу (назначение разработки, исходные данные и т.п.)
Разработать программа для трекинга распределенной группы людей.
Программа для трекинга распределенной группы людей должна выполнять следующие задачи: аутентификация пользователей, работа с картами, формирование групп, хранение информации об истории местоположений пользователя.

Программа для трекинга распределенной группы людей должна разрабатываться с помощью среды разработки Android Studio с помощью инструментов, имеющих лицензию свободного распространения.

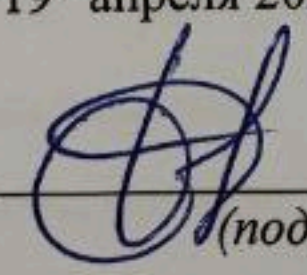
5. Перечень подлежащих разработке вопросов

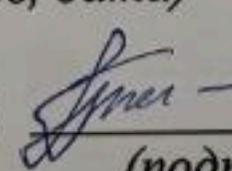
Требуется произвести анализ предметной области, определить функциональный состав программы, выбрать стек технологий для реализации программы, разработать алгоритм работы программы, разработать диаграммы: вариантов использования, развертывания, деятельности, данных, классов, с последующим написанием программы.

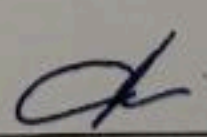
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10	19.04.2024-24.04.2024	
2.	Выбор инструментов для разработки программы	10	25.04.2024-30.04.2024	
3.	Проектирование и составление UML диаграмм для реализации программы	15	01.05.2024-06.05.2024	
4.	Реализация базы данных	15	07.05.2024-13.05.2024	
5.	Реализация программы	30	14.05.2024-28.05.2024	
6.	Развёртывание приложения и отладка	10	29.05.2024-01.06.2024	
7.	Оформление выпускной работы	10	02.06.2024-05.06.2024	

Дата выдачи задания "19" апреля 2024г.

Руководитель работы  19.04.2024 Федюнин Р.Н.
(подпись, дата)

Задание к исполнению принял студент  19.04.2024 Пименова А.А.
(подпись, дата)

Работу к защите допустить
Декан факультета  13.06.2024 Фионова Л.Р.
(подпись, дата)

Реферат

Работа содержит 133 листа, 29 рисунков, 8 таблиц, 9 источников, 3 приложения

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

Цель ВКР – разработка программы для трекинга распределенных групп людей. Отображение маршрутов, геолокации пользователей осуществляется с помощью MapKit SDK. Для аутентификации и хранения информации о пользователях и группах используется Firebase.

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

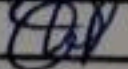
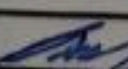
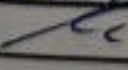
Разработаны макеты окон приложения, структура базы данных. Были разработаны функции составления маршрута, определения и внесения в базу данных текущего местоположения, аутентификация пользователя

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 трудоемкости и затрат на разработку программного обеспечения.

Графическая часть выпускной квалификационной работы содержит UML-диаграммы.

					ПГУ 1.090501.08.001 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Пименова А.А.		11.06.24			
Провер.		Федюнин Р.Н.		11.06.24			
Н. контр.		Бычков А.С.		11.06.24			
Утв.		Митрохин М.А.		11.06.24			
					Программа для трекинга распределенной группы людей Пояснительная записка		
					Лит.	Лист	Листов
						2	133
					ФВТ, каф. ВТ гр.19В031		

Содержание

Перечень сокращений и обозначений.....	5
Введение.....	6
1 Анализ предметной области.....	8
1.1 Анализ темы.....	8
1.2 Анализ аналогов.....	10
1.3 Постановка задачи.....	14
1.3.1 Требования к функциональным характеристикам ПО.....	14
1.3.2 Требования к надежности.....	15
1.3.3 Требования к составу и параметрам технических средств.....	15
1.3.4 Стадии и этапы разработки.....	15
2 Выбор инструментов разработки.....	16
2.1 Выбор языка программирования.....	16
2.2 Выбор сервиса для отображения карт.....	16
2.3 Выбор сервиса облачной СУБД.....	18
2.4 Выбор среды разработки.....	19
3 Архитектура приложения.....	21
3.1 Выбор архитектуры приложения.....	21
3.2 Разработка UML-диаграмм.....	22
4 Разработка приложения.....	28
4.1 Разработка интерфейса приложения.....	28
4.2 Аутентификации пользователей.....	41
4.3 Реализация БД для хранения информации.....	43
4.4 Работа с картами.....	46
4.5 Реализация функционала приложения.....	48
4.5.1 Главное окно приложения.....	48
4.5.2 Окно истории местоположения пользователя.....	50
4.5.3 Окно маршрута до точки сбора.....	50

4.5.4 Группа окон аутентификации: окна профиля, авторизации и регистрации.....	51
4.5.5 Группа окон взаимодействия с группой: окна группы, выбора способа вступления в группу, создания группы.....	52
5 Техничко-экономическое обоснование.....	55
6 Безопасность жизнедеятельности.....	61
6.1 Требования к рабочему месту инженера-разработчика.....	61
6.2 Требования к микроклимату в рабочем помещении.....	63
6.2.1 Требования к шуму.....	63
6.2.2 Требования к влажности.....	63
6.2.3 Требования к температуре.....	63
6.3 Окраска и коэффициенты отражения.....	64
Заключение.....	66
Список используемых источников.....	67
Приложение А – UML- диаграммы.....	68
Приложение А.1 – Диаграмма вариантов использования.....	69
Приложение А.2 – Диаграмма развертывания.....	70
Приложение А.3 – Диаграммы деятельности.....	71
Приложение А.4 – Диаграмма классов.....	72
Приложение А.5 – Диаграмма данных.....	73
Приложение Б – Листинг программы.....	74
Приложение В – Презентация.....	128

Перечень сокращений и обозначений

В данном проекте применяются следующие сокращения и обозначения

БД – база данных

ГК – группа компаний

НК РФ – налоговый кодекс Российской Федерации

ОМС – обязательное медицинское страхование

ОПС – обязательное пенсионное страхование

ОСС – обязательное социальное страхование

ОС – операционная система

ПО – программное обеспечение

ПФР – пенсионный фонд России

ПК – персональный компьютер

СанПин – санитарно-эпидемические нормы

СУБД – система управления базами данных

ФОМС – фонд обязательного медицинского страхования

ФСС – фонд социального страхования

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

GPS (Global Positioning System) – глобальная позиционирующая система

IDE (Integrated Development Environment) – интегрированная среда разработки

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

3D – трёхмерная графика

					<i>ПГУ 1.090501.08.001 ПЗ</i>	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

Мониторинг — это специальная форма наблюдения за текущим изменением тех или иных процессов или объектов в пространстве и во времени, осуществляемая на постоянной основе. Для наблюдения за изменениями местоположения используется GPS-мониторинг.

GPS-мониторинг — это технология, которая основывается на глобальных системах позиционирования и позволяет использовать данные со спутников для отслеживания в режиме реального времени местоположения объектов. Данная технология может использоваться в различных отраслях, например:

- логистические компании;
- таксопарки;
- сельскохозяйственные компании;
- строительные организации;
- авиация и мореходство.

Так же GPS используется для мониторинга местоположения людей. Например, многие службы доставки готовой еды и продуктов позволяют пользователям отслеживать, где на данный момент находится их заказ.

Вопрос мониторинга местоположения людей в современном мире очень важен. Большая вероятность потеряться самому или потерять в городе или на природе других людей. Данный вопрос актуален для многих сфер деятельности. Рассмотрим проблематику данного вопроса на примере туризма.

Туризм – выезды (путешествия) посетителей в другую страну или местность, отличную от места постоянного жительства, с любой целью, кроме трудоустройства. Человека, совершающего такое путешествие, называют «туристом», «путешественником», «посетителем».

Информационные технологии являются неотъемлемой частью жизни современного человека. Они используются практически во всех сферах жизни человека, в том числе и в туризме.

Мобильные приложения стали незаменимым инструментом для путешественников. Благодаря мобильным приложениям, путешественники могут легко ориентироваться в незнакомом городе и получать актуальную информацию в режиме реального времени. А также, они предоставляют доступ к информации о достопримечательностях, ресторанах, местных мероприятиях и общественном транспорте, а также предлагают персонализированные рекомендации и советы.

					ПГУ 1.090501.08.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

1 Анализ предметной области

1.1 Анализ темы

Туризм разделяется на активный и пассивный. Основной смысл такого разделения видов туризма — определение видов передвижения и интенсивности деятельности человека в процессе туризма. Активный туризм — это путешествие, основанное на энергичном передвижении по заданному маршруту с применением физических нагрузок. В настоящее время туризм - одно из самых интересных занятий, ведь оно затрагивает все сферы жизни и даже изменяет окружающую среду.

Современный активный туризм разделяется на несколько видов:

– пеший туризм;

– велосипедный туризм:

1) классический;

2) цивилизованный;

3) познавательный;

4) спортивный;

5) гонщицкий;

– водный туризм:

1) рафтинг;

2) каякинг;

3) сплавы на плотах;

4) яхтинг;

– спелеотуризм;

– конный туризм;

1) конно-верховой поход;

2) конно-верховой поход с сопровождением;

3) конно-упряжной туризм;

– лыжный туризм.

Рассмотрим пеший туризм более подробно.

Пеший туризм – это походы, которые совершает турист без использования средств передвижения. Эти походы осуществляются во всех климатических зонах и географических регионах. Пешеходный туризм самый известный вид туризма из всех возможных. Его положительными чертами являются доступность и полезность любому человеку, независимо от возраста и физического развития, а также возможность выбора нужного маршрута. Из всех видов туризма пеший туризм – самый естественный и самый легкий, так как он не требует подготовки. В пеших походах туристы укрепляют свой организм, улучшают здоровье, повышают выносливость, выдержку, стойкость, силу и другие качества, приобретают новые навыки, умения, а также учатся преодолевать сложные препятствия. Для пеших походов не нужны специальные средства передвижения, для них не нужна специальная экипировка. Пеший туризм также можно совмещать с культурно-историческим туризмом для изучения культуры различных стран и регионов [1].

По форме организации туризм выделяют следующие виды туризма:

- организованный (заранее разработанный профессиональной организацией-регулятором, туроператором, турфирмой и т. п.);
- неорганизованный (заранее разработанный лично путешественниками);
- самостоятельный туризм (на добровольной любительской основе, подготовленные туристскими клубами, союзами, туристско-спортивными организациями).

Неорганизованный туризм может выполняться как одним человеком, так и группой лиц. При этом разные члены группы могут передвигаться по разным маршрутам. Группы лиц, путешествующие совместно, но по частично различающимся маршрутам будем называть распределенной группой путешественников.

Для туризма важным инструментом являются карты. Электронные карты имеют ряд преимуществ над бумажными, а именно:

- доступность;

- компактность. В одном приложении помещена карта всего мира, что позволяет использовать его в любой точке мира;
- динамическое отображение текущего местоположения;
- возможность построения маршрута до точки назначения (пешком, на общественном транспорте или на автомобиле);
- возможность просматривать историю перемещений.

Для распределенных групп важно знать, где на данный момент находится каждый участник. Это необходимо для удобства определения точек сбора группы, а также для обеспечения безопасности, так как определение местоположения дает возможность прийти на помощь человеку, не выходящему на связь.

Было решено разработать мобильное приложение, с помощью которого распределенные группы людей могли бы более безопасно путешествовать по разным городам. Члены групп смогут отслеживать местоположение других участников, смотреть историю перемещений, а также устанавливать точку сбора и строить маршрут до нее.

1.2 Анализ аналогов

Перед разработкой программы был проведен сравнительный анализ аналогичных приложений в Google Play, которые могут помочь пользователям в определении местоположения других людей или планировании совместного времяпровождения. Был проведен анализ следующих приложений: «Blink», «iSharing», «Connected».

Приложение «Blink» обладает узким функционалом и направлено на общение с друзьями и отслеживание местоположения друзей (рисунок 1.1). Оно также позволяет «открывать» карту, то есть отображает места, где человек уже был. Достоинствами приложения является возможность изменить стиль карты (спутник или схема), простота интерфейса, отсутствие платных функций. Недостатком данного приложения является отсутствие истории перемещений. По отзывам пользователей, геолокация в приложении часто дает сбой.

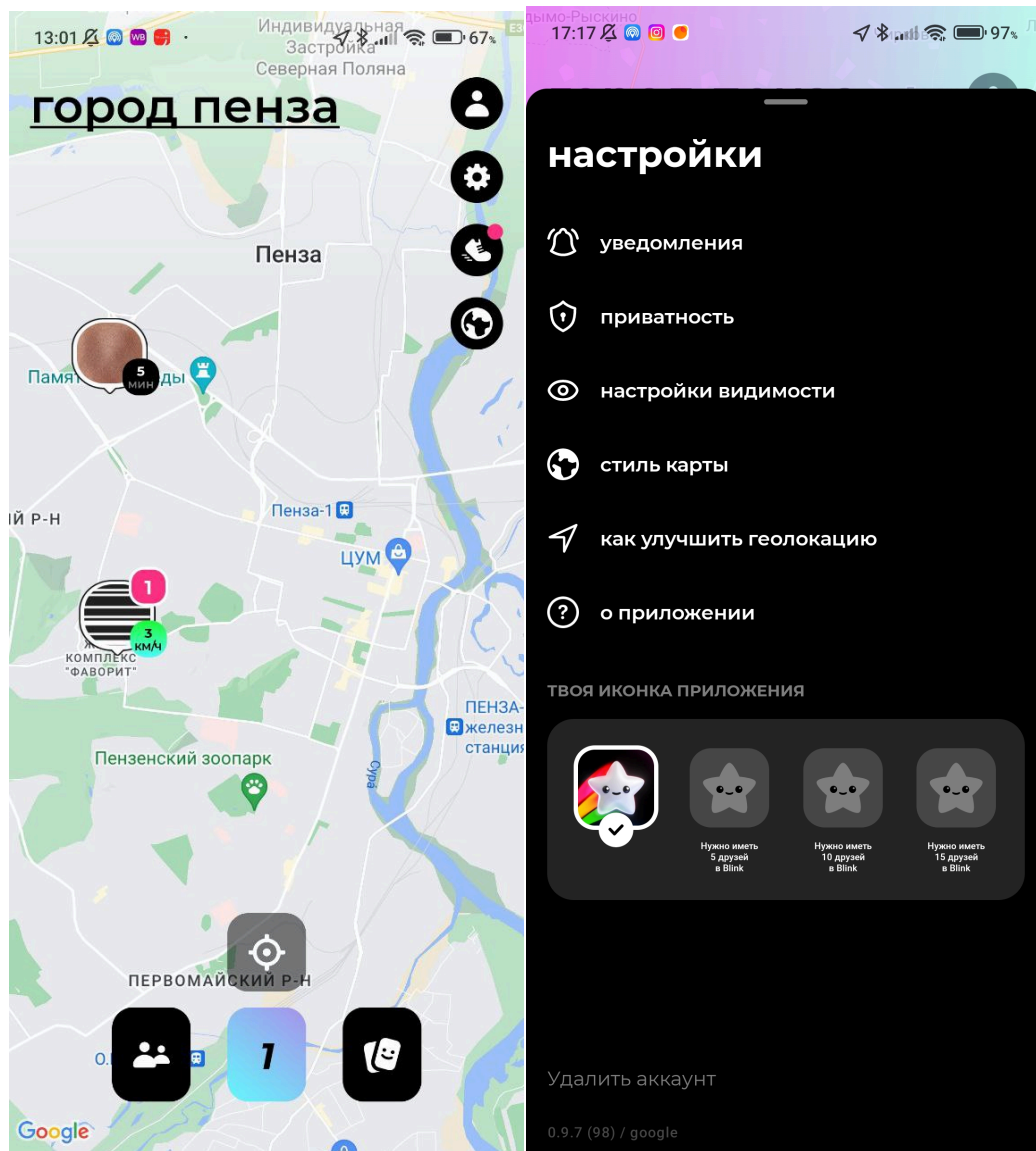


Рисунок 1.1 – Интерфейс приложения «Blink»

Приложение «iSharing» позволяет делиться своим местоположением в режиме реального времени, добавлять друзей, отображать их местоположение, отправлять сигнал SOS, а также просматривать историю перемещений (рисунок 1.2). Достоинствами данного приложения являются: возможность изменения единицы измерения (километр или миля), выбор источника карты (автоматический, Google или NAVER). Недостатком данного приложения является присутствие рекламы, недостаточная точность геолокации, невозможность добавить человека без отправки СМС-приглашения, а также несвоевременное обновление карты местности. В приложении есть платные функции, а именно: история перемещений за 90 дней, неограниченное кол-во сохраняемых мест,

просмотр 3D-окружения улиц, уведомление о низком заряде друзей, оповещение о поездках, отчет о вождении, отключение рекламы.

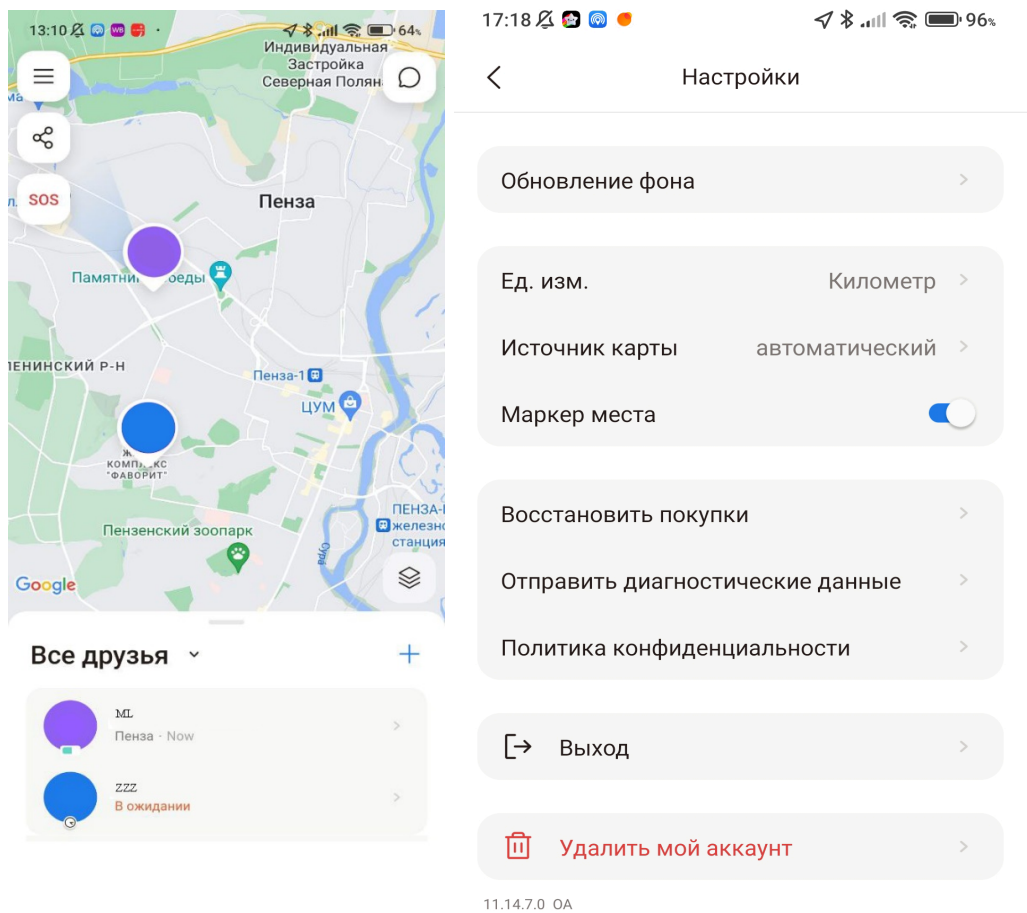


Рисунок 1.2 – Интерфейс приложения «iSharing»

Приложение «Connected» позволяет создавать группу (круг), отображать текущее местоположение и историю перемещений, добавлять места (рисунок 1.3). Оно обладает самым широким функционалом из аналогов, однако большая часть функций представлена в подписке «Premium», а именно: история путешествий за 30 дней, отключение рекламы, отчеты о движении, неограниченное место, неограниченная помощь (предупреждение членов круга, когда необходима помощь), уведомление о превышении скорости одного из членов круга, звонок на устройство при потере, история уведомлений за 30 дней. По отзывам пользователей, приложение имеет задержку и неточное отображение местоположения.

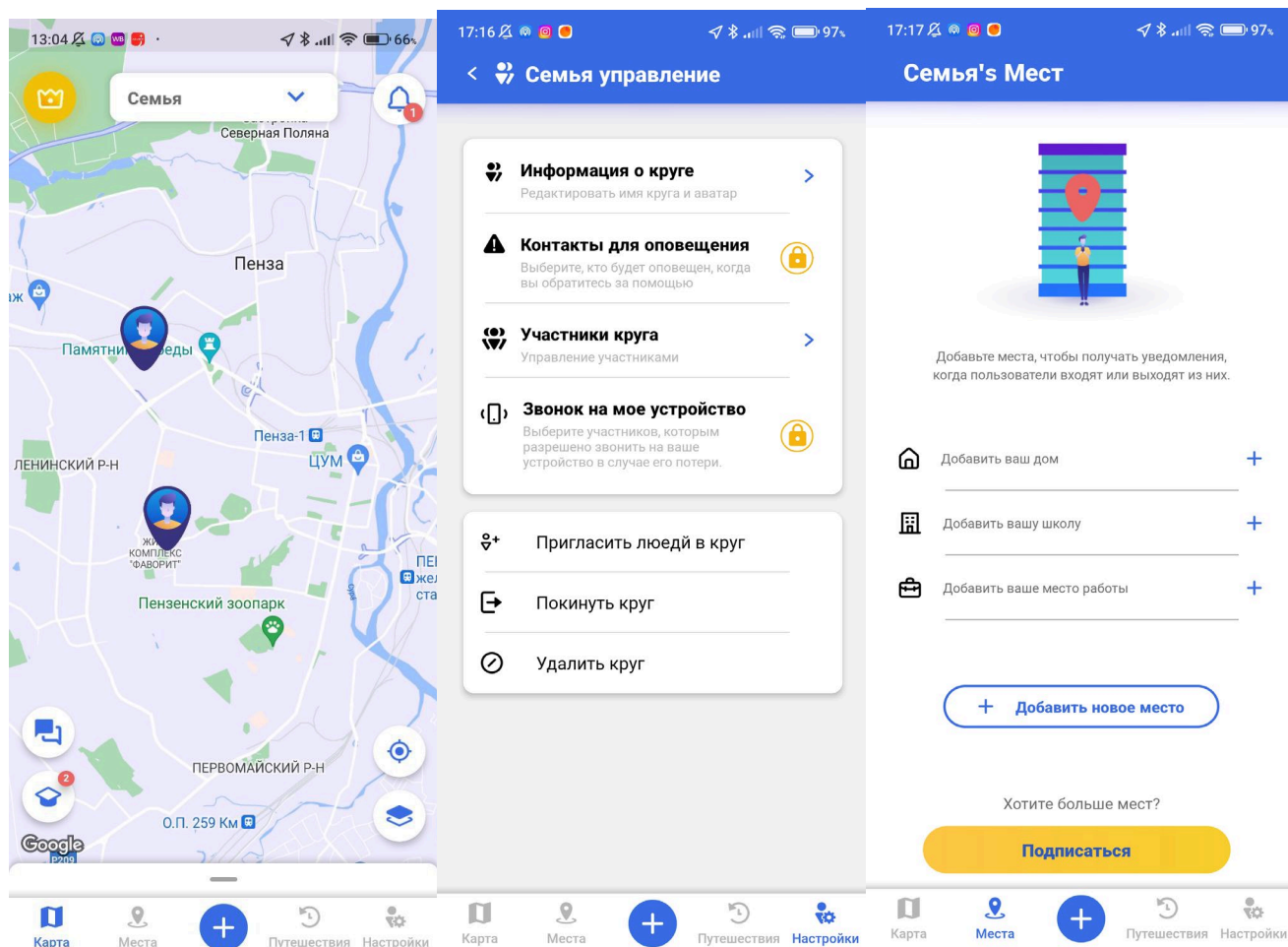


Рисунок 1.3 – Интерфейс приложения «Connected»

Составлена таблица с результатами сравнительного анализа приложений (таблица 1.1).

Таблица 1.1 – Результат сравнительного анализа приложений

Функция	Название приложения		
	Blink	Connected	iShared
Отображение текущего местоположения	+	+	+
Отображение истории перемещений	-	+	+
Создание группы	-	+	-
Создание точки сбора группы	-	-	-
Отображение маршрута до точки	-	-	-
Интуитивно понятный интерфейс	+	-	-

Функция	Название приложения		
	Blink	Connected	Blink
Количество скачиваний	1млн+	1млн+	10млн+
Средняя оценка	4.5	3.8	4.4

Исходя из этого анализа становится ясно, что в магазине приложений Google Play не существует полноценного приложения по трекингу распределенных групп путешественников, а существующие имеют ограниченный функционал. Именно поэтому появилась необходимость разработать приложение, имеющее набор всех необходимых функций, а именно: создание группы, отображение текущего местоположения и истории перемещений членов группы, добавление точки сбора группы, отображение маршрута до точки сбора.

1.3 Постановка задачи

1.3.1 Требования к функциональным характеристикам ПО

Программное обеспечение для трекинга распределенной группы людей должно обладать следующим функционалом:

- регистрация и авторизация пользователей;
- создание, редактирование и удаление групп;
- отображение местоположения членов группы на общей карте;
- отображение истории перемещений членов группы;
- установка и редактирование точки сбора, а также построение маршрута до нее от каждого участника группы.

Входными данными являются:

- информация о пользователе программы (имя, email, пароль, текущее местоположение);
- информация о группах (название, описание).

Выходными данными являются:

- хранимые данные в БД;
- маршрут до точки сбора;

– история перемещения пользователей на карте.

1.3.2 Требования к надежности

При функционировании ПО должен осуществляться контроль входной и выходной информации. В случае возникновения ошибки необходимо выдавать предупреждение и предоставляться возможности исправления ошибок.

1.3.3 Требования к составу и параметрам технических средств

Разрабатываемая программа предназначена для работы на мобильных телефонах под управлением ОС Android со следующими параметрами:

- минимальный SDK – 21;
- целевой SDK – 34;
- наличие GPS модуля;
- наличие выхода в интернет.

1.3.4 Стадии и этапы разработки

Стадии и этапы разработки представлены в таблице 1.2.

Таблица 1.2 – Стадии и этапы разработки

Наименование этапов работы	Объем работы	Срок выполнения
Анализ предметной области	10	19.04.2024- 24.04.2024
Выбор инструментов для разработки программы	10	25.04.2024- 30.04.2024
Проектирование и составление UML-диаграмм для реализации программы	15	01.05.2024- 06.05.2024
Реализация базы данных	15	07.05.2024- 13.05.2024
Реализация программы	30	14.05.2024- 28.05.2024
Развёртывание приложения и отладка	10	29.05.2024- 01.06.2024
Оформление выпускной работы	10	02.06.2024- 05.06.2024

2 Выбор инструментов разработки

2.1 Выбор языка программирования

Для программирования под ОС Android используются языки Java и Kotlin. Данные языки обладают рядом преимуществ и недостатков:

- Kotlin более прост для начинающих разработчиков, программный код получается в среднем на 40% короче;
- язык Java обладает большим сообществом, с большим количеством форумов, тематических сайтов и методических ресурсов, 7.8% разработчиков предпочитают Kotlin, более чем 40% — Java;
- низкая скорость компиляции приложений на Kotlin, по сравнению с Java [2].

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

2.2 Выбор сервиса для отображения карт

Для разработки приложений с использованием карт необходимо выбрать SDK. В настоящее время самыми популярными SDK для работы с картами являются Google Maps SDK, Yandex MapKit, 2GIS. Сравнения SDK для работы с картами представлено в таблице 2.1 [3].

Таблица 2.1 – Сравнение SDK для работы с картами

Критерий	Yandex MapKit	2GIS	Google Maps
Покрытие	Лучшее покрытие России, уступает Google в покрытии мира	Уступает конкурентам в покрытии как в России, так и в других странах	Лучшее покрытие всего мира
Детализация	Хорошая детализация России,	Одна из лучших детализаций в городах присутствия	Хорошая детализация по всему миру. На

Критерий	Yandex MapKit	2GIS	Google Maps
	достаточная в мире		отсутствовать некоторые города.
Отображение пробок в крупных городах	Да. Отображение доп. информации о дорожной обстановке	Не все города	Не все города. Интеграция с сервисом Waze
Вариант выбора отображения ландшафта	Карта, спутник, народная карта	Карта	Карта, спутник, Велокарта, общественный транспорт
Справочная информация	Подробная информация об организациях	Подробная информация об организациях. Ежемесячные обновления	Хуже других знает российские организации
Построение маршрута	Автомобиль, общественный транспорт. Строит с учетом пробок. Требуется интернет для построения	Автомобиль, общественный транспорт. Возможность отдельно выбрать вариант «Метро». Не требует интернета для построения маршрута	Автомобиль, общественный транспорт, пешеходный маршрут. Возможность выбрать только один из видов транспорта или вариант пешком. Строит с учетом пробок и расписания общественного транспорта. Требуется интернет для построения
Получение API-ключа	Бесплатная версия (практически аналогичный с платной функционал)	Бесплатная версия (ограниченный функционал)	Недоступно для разработчиков РФ

Исходя из вышеперечисленного было решено использовать Yandex MapKit.

2.3 Выбор сервиса облачной СУБД

Самыми распространенными сервисами облачной СУБД являются FireBase и MySQL.

Проанализируем различия между Firebase и MySQL:

– архитектура. Firebase — это база данных NoSQL, которая хранит и синхронизирует данные в режиме реального времени, иногда называемая хранилищем документов в режиме реального времени. MySQL — это система управления реляционными базами данных с открытым исходным кодом, основанная на специфичном для предметной области языке SQL;

– обработка данных: Firebase эффективно обрабатывает большие наборы данных. MySQL - хороший выбор для обработки сложных данных;

– языковая поддержка: MySQL поддерживает больше языков программирования, чем Firebase, включая Ada, C++, Python и другие. Firebase специализируется на серверных языках для поддержки разработчиков приложений, таких как JSON и Node.js;

– цена: Firebase имеет бесплатную и платную версии. MySQL имеет открытый исходный код и бесплатен;

– отзывы: Firebase имеет оценку отзывов клиентов 4,5/5 на веб-сайте G2. У MySQL оценка отзывов клиентов немного ниже - 4,4/5.

Исходя из вышеперечисленных различий, было решено использовать FireBase, так как для данного приложения необходимо обрабатывать большие простые наборы данных.

2.4 Выбор среды разработки

Самыми популярными IDE для разработки Android-приложений являются:

- Android Studio;
- Visual Studio (с Xamarin);
- Eclipse;
- IntelliJ IDEA [4].

Рассмотрим подробнее каждую из них.

Android Studio — официальная интегрированная среда разработки (IDE) для разработки приложений Android. Разработанный Google, он предоставляет полный набор инструментов для создания высококачественных приложений для различных устройств Android.

Преимущества:

- обеспечение встроенной поддержки Google Cloud;
- предоставление эмулятора для версий и устройств Android;
- интеграция с GitHub и готовые шаблоны;
- предоставление поддержки диаграмм UML.

Недостатки:

- ограниченная языковая поддержка (Java, Kotlin);
- единственная, поддерживаемая ОС – Android.

Инструменты Visual Studio для Xamarin IDE позволяют разработчикам создавать приложения для Android, а также iOS и Windows с помощью единой общей базы кода .NET. Xamarin - платформа с открытым исходным кодом для создания современных и производительных приложений для .NET и iOS, Android и Windows. Используя этот инструмент, программисты мгновенно видят изменения на подключенных мобильных устройствах при разработке адаптивных приложений.

Преимущества:

- позволяет .Net-программистам писать приложения для Android;
- возможность создавать кроссплатформенные мобильные приложения.

Недостаток – ограниченная языковая поддержка (C#).

Eclipse — это бесплатная официальная IDE, которая используется для разработки приложений на нескольких языках. Изначально она предназначалась только для программирования на Java, но со временем добавила поддержку C/C++, Cobol, Python, Perl, JEEclipse aims language и PHP.

Преимущества:

- встроенная консоль;
- удобство управления крупными проектами;
- широкий выбор плагинов.

Недостатки:

- отсутствие функции замены кода;
- тяжелая настройка IDE.

IntelliJ IDEA — это JetBrains IDE, содержащая изысканный интерфейс и позволяющая создавать корпоративные решения и мобильные приложения. Этот интеллектуальный редактор кода поддерживает фреймворки Kotlin, Java, Groovy и Scala для создания приложений для Android.

Преимущества:

- мощный отладчик для кода Java;
- имеет четкую и подробную документацию;
- поддержка нескольких языков;
- повреждение файлов незначительно;
- поддержка GitHub.

Недостатки:

- когда IDE теряет токен доступа (соединение) к GitHub, она случайным образом запрашивает пароль без указания конкретного контекста;
- может не подойти для интерфейсных разработчиков.

Проанализировав преимущества и недостатки представленных IDE, было принято выбрать Android Studio для разработки мобильного приложения.

3 Архитектура приложения

3.1 Выбор архитектуры приложения

Реализуемое приложение состоит из серверной части, СУБД, и пользовательской части. Для реализации проекта была выбрана монолитная архитектура приложения. Монолитная архитектура означает, что пользовательский интерфейс и код для доступа к данным объединены в одну автономную программу.

Плюсами монолитной архитектуры являются:

- легкое развитие на первоначальном этапе. В связи с этим относительно легко добавлять новые функциональные возможности в монолит;
- меньшая задержка. Поскольку все вызовы в монолите являются локальными, пользователи испытывают меньшую задержку в выполнении операций, чем в среде на основе микросервисов;
- удобство отладки. Так как весь код располагается в одном месте, проще выполнять отладку и выявлять ошибки.

Недостатками монолитной архитектуры являются:

- длительное время развертывания приложения. Вся кодовая база приложения загружается полностью за один раз. Восстановление всего приложения занимает много времени и замедляет скорость, с которой могут быть предоставлены новые функции;
- сложность. Большие приложения сложнее для понимания. Могут появиться побочные эффекты изменений, которые легко пропустить из-за легко пропускаемых зависимостей;
- меньшая масштабируемость. Масштабирование становится вопросом добавления новых экземпляров для запуска всей кодовой базы, а не добавления экземпляров для масштабирования отдельных частей приложения, которые могут использоваться чаще других. Неравное использование поддерживаемых модулей может привести к потере некоторых ресурсов и нагрузке на сервер.

					<i>ПГУ 1.090501.08.001 ПЗ</i>	Лист
						21
Изм.	Лист	№ докум.	Подпись	Дата		

3.2 Разработка UML-диаграмм

Разрабатываемое приложение состоит из мобильного приложения клиента и сервера баз данных. Клиент будет взаимодействовать с сервером баз данных для того, чтобы хранить и использовать информацию о участниках и группах. UML-диаграмма развертывания (рисунок 3.1) наглядно показывает взаимодействие клиента и сервера через интернет.

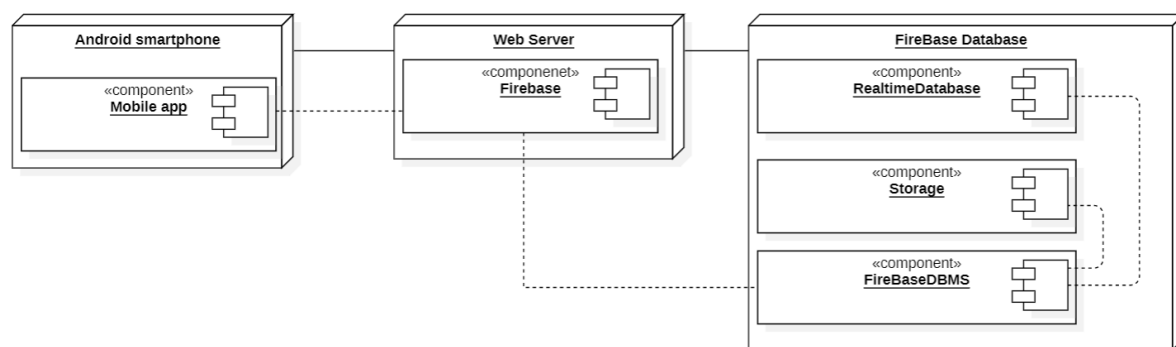


Рисунок 3.1 – Диаграмма развертывания

На рисунке 3.2 представлена диаграмма вариантов использования, отображающая функционал приложения.

Разработанная диаграмма вариантов использования включает в себя: один актер (Пользователь) и 18 вариантов использования (Регистрация, Авторизация, Создание группы, Управление группой, Удаление группы, Редактирование группы, Добавление точки сбора, Удаление точки сбора, Вступление в группу, Выход из группы, Использование интерактивной карты, Просмотр информации о группе, Просмотр текущего местоположения членов группы, Просмотр истории перемещений члена группы, Отображение маршрута до точки сбора, Просмотр профиля, Изменение имени, Выход из аккаунта).

При запуске программы пользователь попадает на страницу с картой. На ней отмечено текущее местоположение пользователя. При нажатии на иконку профиля, пользователь попадает на страницу авторизации (если пользователь не был авторизован ранее). На этой странице необходимо авторизоваться или, если нет учетной записи, нажать на кнопку регистрации и зарегистрироваться. После

авторизации (регистрации) пользователь попадает на страницу профиля, где можно просмотреть информацию о пользователе и, при необходимости, изменить имя или выйти из аккаунта. Со страницы профиля можно перейти обратно на главный экран. Если пользователь авторизован, то он может перейти на страницу группы. Если пользователь не является членом группы, то ему предоставляется выбор: создать группу или выступить в уже существующую группу. При выборе пункта «Создание группы» пользователь вводит информацию о группе, подтверждает создание, после чего пользователь попадет в окно с информацией о группе. При выборе пункта «Вступить в группу» выполняется переход на окно со списком доступных групп. После выбора группы выполняется переход на окно с информацией о группе. Если пользователь не является владельцем группы, то он может выйти из группы. Если пользователь является владельцем группы, то он может удалить группу, создать точку сбора (если точка сбора не была установлена), удалить точку сбора (если точка сбора была установлена ранее), изменить название или описание группы. Если пользователь является членом группы, то при переходе на главную страницу на карте будут отображены метки членов группы и точки сбора. При нажатии на метку точки сбора, открывается окно с маршрутом от текущего местоположения пользователя до точки сбора. При нажатии на метку другого члена группы, открывается окно с историей перемещений данного члена группы.

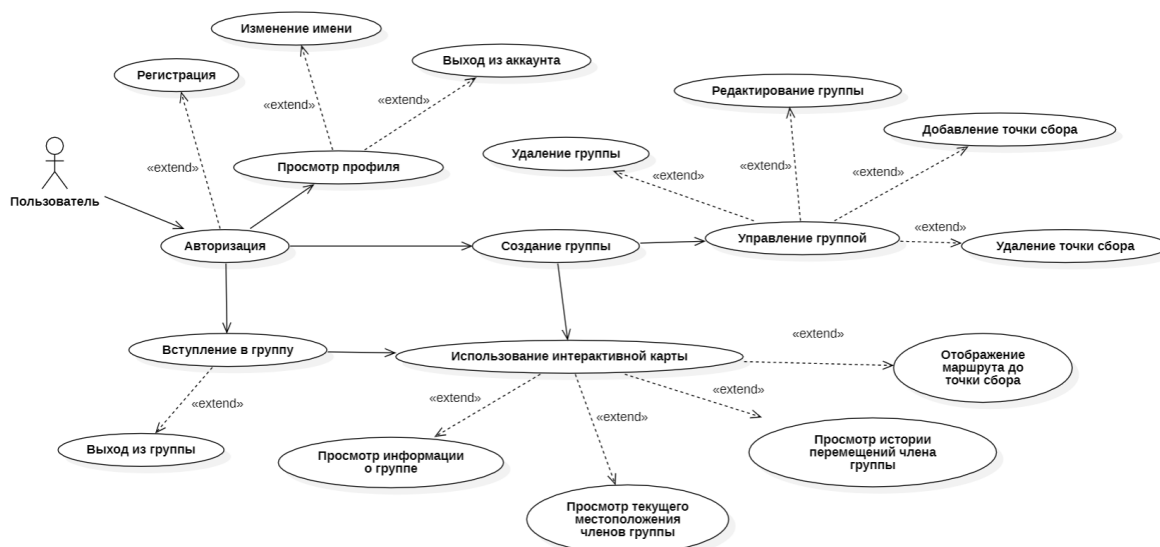


Рисунок 3.2 - Диаграмма вариантов использования

Разработаны диаграммы деятельности (рисунки 3.3-3.5), которые отображают последовательность действий для некоторых функций приложений.

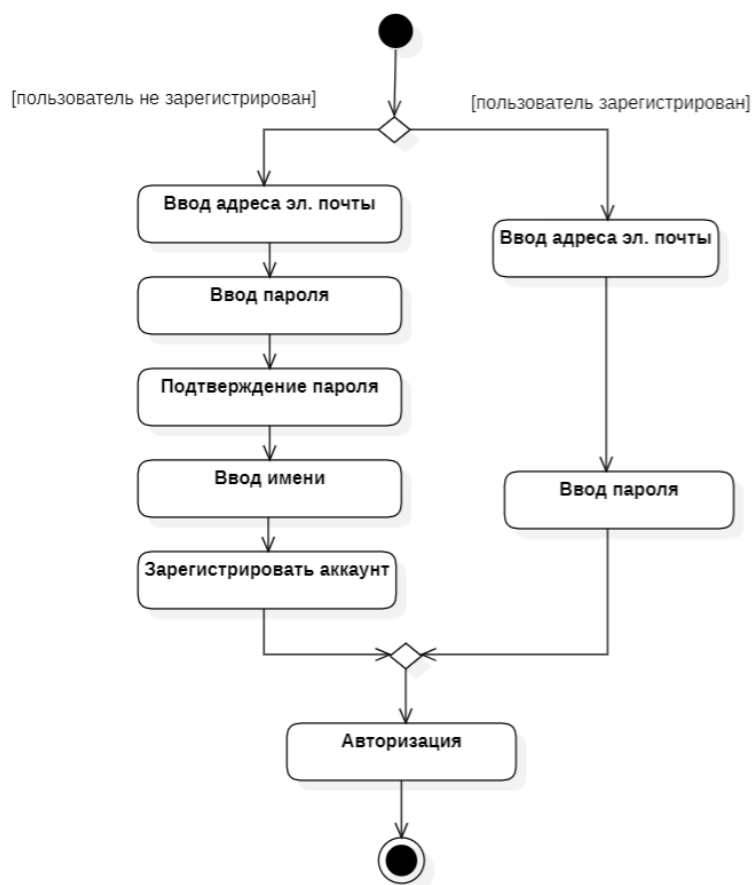


Рисунок 3.3 – Диаграмма деятельности «Авторизация»

При переходе в профиль не авторизованный пользователь попадает в окно авторизации. Если пользователь зарегистрирован, то ему необходимо ввести адрес электронной почты и пароль, после чего подтвердить авторизацию. Если пользователь не зарегистрирован, то он должен нажать на кнопку «Регистрация», после чего ввести адрес электронной почты, пароль, подтверждение пароля и имя, после чего подтвердить регистрацию. После авторизации пользователь попадет в окно профиля.

При выборе пункта «Создание группы» пользователь попадает в окно создания группы, где необходимо ввести название и описание группы и подтвердить создание или отменить. При подтверждении создания пользователь переходит на страницу группы. При отмене создания пользователь переходит на страницу выбора вступить в группу или создать группу.

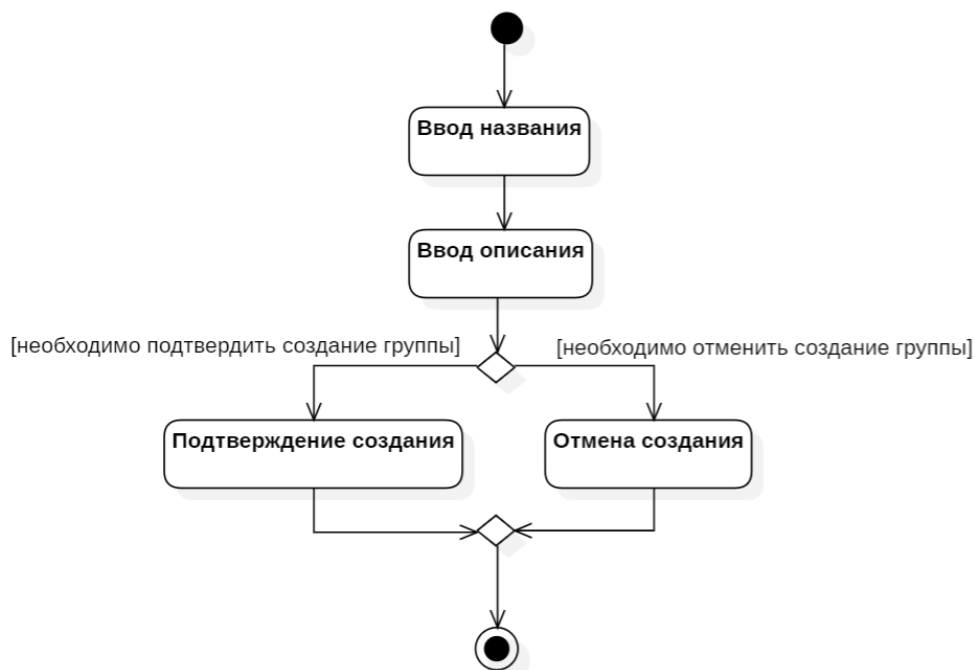


Рисунок 3.4 – Диаграмма деятельности «Создание группы»

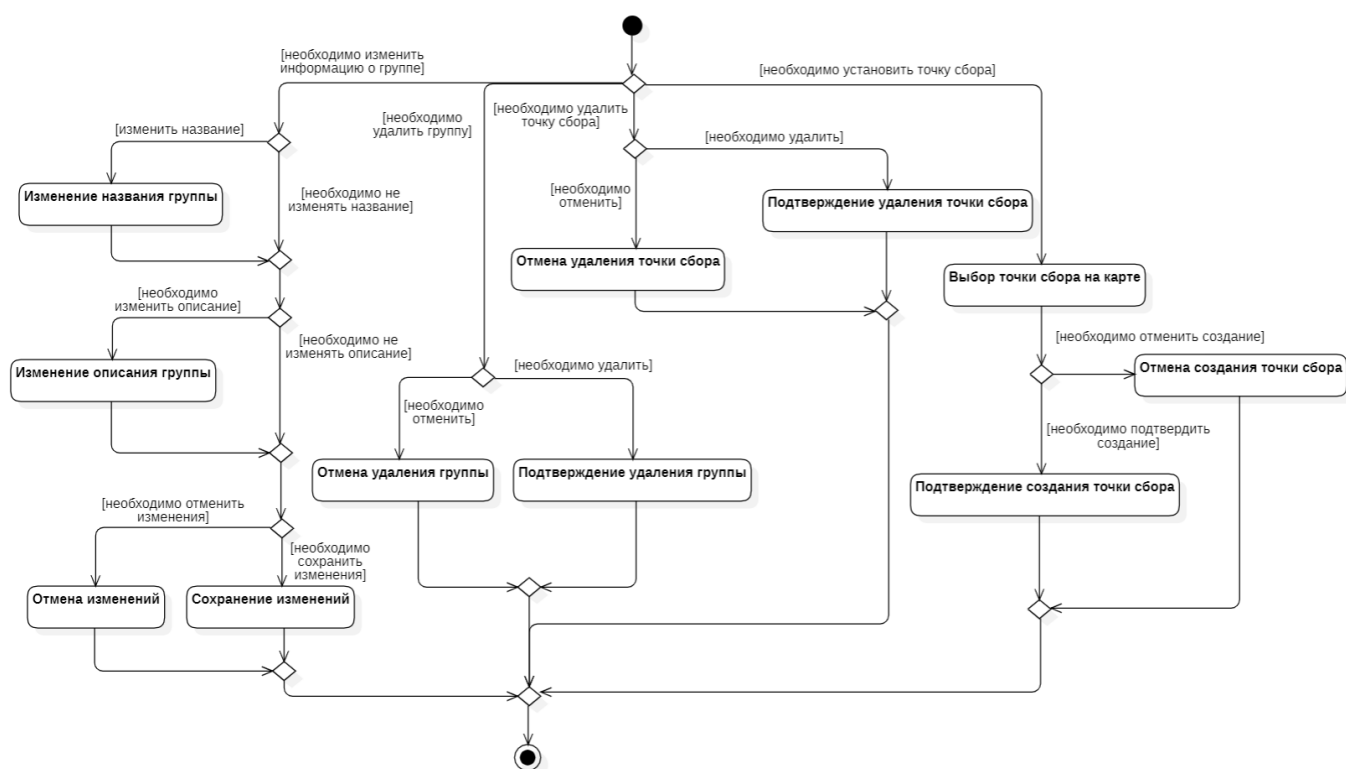


Рисунок 3.5 – Диаграмма деятельности «Управление группой»

При переходе в окно группы владельца, ему предоставляется выбор действий: изменить информацию о группе, удалить группу, создать точку сбора (если точка сбора не установлена), удалить точку сбора (если точка сбора

установлена). При выборе пункта изменения информации о группе пользователь попадает в окно изменения группы, где может изменить название группы и/или описание группы, после чего сохранить или отменить изменения. При выборе пункта удаления группы пользователь может подтвердить или отменить удаление в диалоговом окне. При выборе пункта создания точки сбора пользователь попадает в окно с картой, где перемещением метки выбирает местоположение точки сбора, после чего может подтвердить или отменить создание. При выборе пункта удаления точки сбора пользователь может подтвердить или отменить изменения в диалоговом окне.

Для создания базы данных была разработана диаграмма данных (рисунок 3.6). На ней отображены сущности разрабатываемой базы данных.

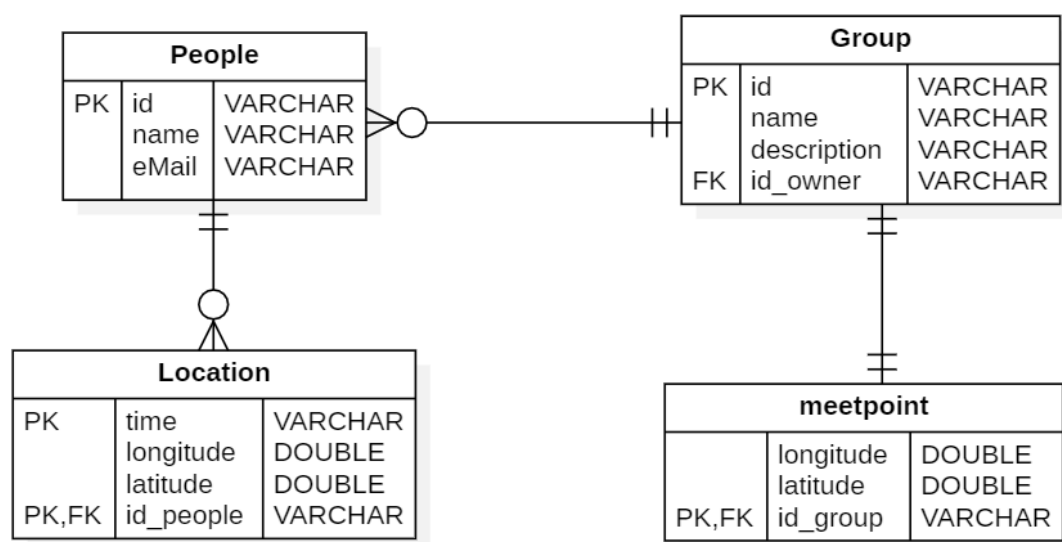


Рисунок 3.6 – Диаграмма данных

Модель базы данных содержит четыре сущности. Таблица People содержит информацию о пользователях, а именно: name - имя пользователя строкового типа, id – UID пользователя строкового типа, email – электронная почта строкового типа. Таблица Group содержит информацию о группах, а именно: id –идентификатор группы строкового типа, name – название группы строкового типа, description – описание группы строкового типа, id_owner – UID пользователя-владельца группы. Таблица Location содержит информацию о местоположении пользователя, а

именно: time – время, в которое пользователь находился по координатам, строкового типа, longitude – долгота местоположения вещественного типа, latitude – широта местоположения строкового типа, id_people – UID пользователя. Таблица meetpoint содержит информацию о точке сбора группы, а именно: longitude – долгота точки сбора вещественного типа, latitude – широта точки сбора строкового типа, id_group – идентификатор группы строкового типа.

					ПГУ 1.090501.08.001 ПЗ	Лист
						27
Изм.	Лист	№ докум.	Подпись	Дата		

4 Разработка приложения

4.1 Разработка интерфейса приложения

Интерфейс – это часть программы, позволяющая людям управлять программой с помощью голоса, нажатий, жестов или с помощью командной строки. В качестве инструментов взаимодействия с системами могут выступать:

- текстовые поля;
- кнопки и галочки;
- всплывающие подсказки;
- выпадающие списки;
- элементы меню программы;
- переключатели.

При разработке программы были созданы интерфейсы следующих окон:

- главное окно;
- окно группы;
- окно выбора способа вступления в группу;
- окно выбора группы;
- окно создания группы;
- окно изменения группы;
- окно создания точки сбора;
- окно профиля;
- окно регистрации;
- окно авторизации;
- окно отображения маршрута до точки сбора;
- окно отображения истории перемещений пользователя;
- диалоговое окно изменения имени пользователя;
- диалоговое окно удаления группы;
- диалоговое окно выхода из группы.

Главное окно приложения отображает местоположение пользователя, членов групп и точки сбора на карте, а также содержит кнопки, с помощью которых можно выполнить переход в окна профиля и группы.

Макет главного окна приложения представлен на рисунке 4.1.

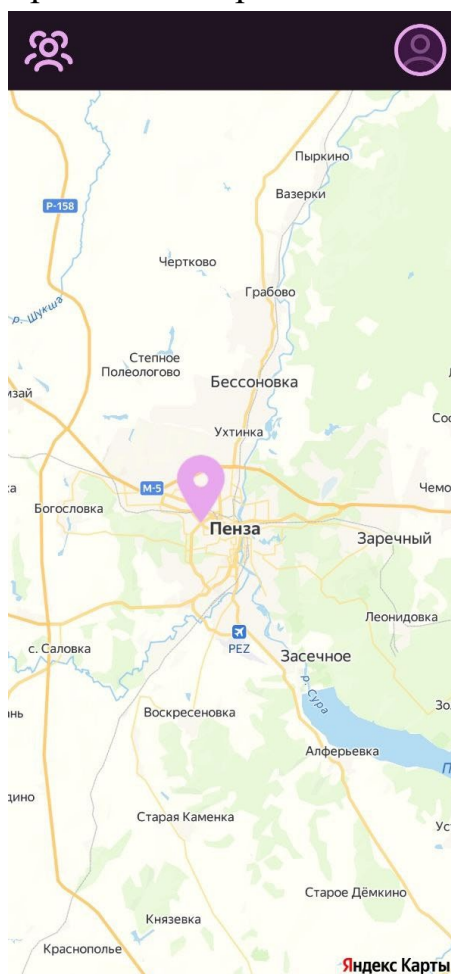


Рисунок 4.1 – Интерфейс главного окна приложения

Элементами главного окна являются:

- карта, типа MapView;
- кнопка профиля, типа Button;
- кнопка группы, типа Button [5].

Окно группы содержит информацию о группе (название, описание), о членах группы (имя и email), а также содержит кнопки, с помощью которых можно выполнить переход на главную страницу или выйти из группы.

Макет окна группы представлен на рисунке 4.2.

Элементами окна группы являются:

- кнопка карты, типа Button;
- кнопка выхода из группы, типа Button;
- информационное поле с названием группы, типа TextView;
- информационное поле с описанием группы, типа TextView;
- список членов группы типа ListView, элементы которого состоят из следующих элементов:

- 1) имя члена группы типа TextView;
- 2) email члена группы типа TextView.

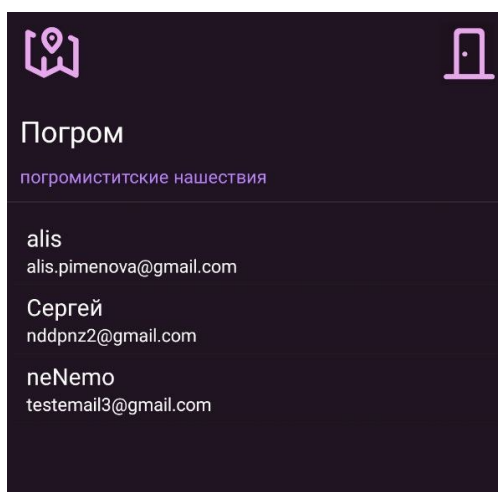


Рисунок 4.2 – Интерфейс окна группы

Окно выбора способа вступления в группу содержит кнопки, с помощью которых можно выбрать создать группу или вступить в существующую.

Макет окна выбора способа вступления в группу представлен на рисунке 4.3.

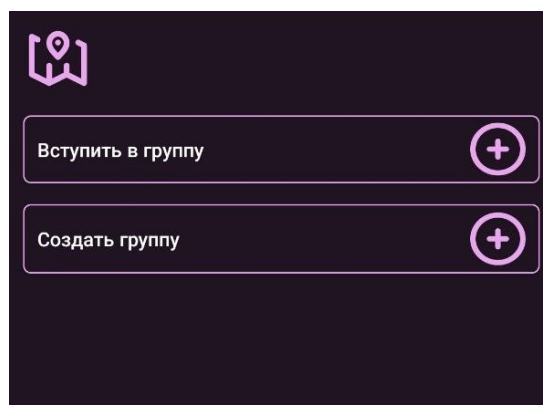


Рисунок 4.3 – Интерфейс окна выбора способа вступления в группу

Элементами окна выбора способа вступления в группу являются:

- кнопка карты, типа Button;
- кнопка вступления в группу, типа Button;
- кнопка создания группы, типа Button.

Окно выбора группы содержит список групп (название и описание групп), кнопку, с помощью которых можно выполнить переход на главную страницу.

Макет окна выбора группы представлен на рисунке 4.4.

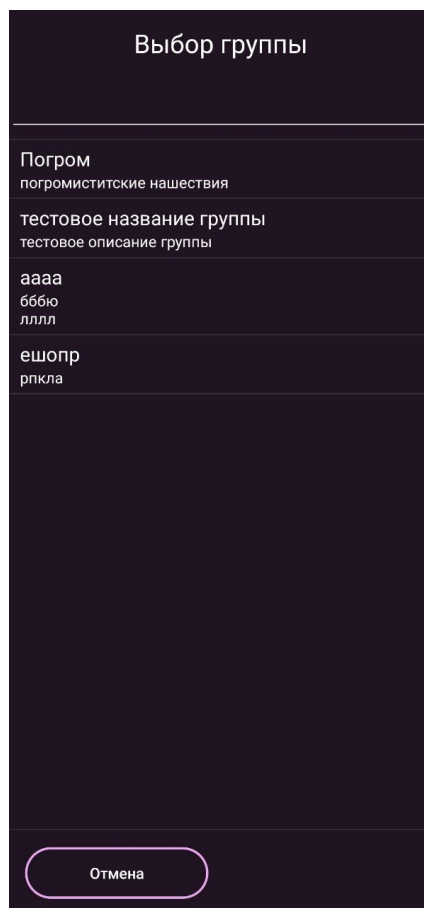


Рисунок 4.4 – Интерфейс окна выбора группы

Элементами окна выбора группы являются:

- кнопка отмены, типа Button;
- список групп типа ListView, элементы которого состоят из следующих элементов:

- 1) название группы типа TextView;
- 2) описание группы типа TextView.

Окно создания группы содержит два поля для ввода названия и описания создаваемой группы, а также две кнопки, с помощью которых можно отменить или подтвердить создание группы.

Макет окна группы представлен на рисунке 4.5.

Рисунок 4.5 – Интерфейс окна создания группы

Элементами окна создания группы являются:

- кнопка отмены, типа Button;
- кнопка создания, типа Button;
- информационное поля, типа TextView;
- поле ввода названия группы, типа EditText;
- поле ввода описания группы, типа EditText.

Окно изменения группы содержит два поля для ввода названия и описания создаваемой группы, а также две кнопки, с помощью которых можно отменить или подтвердить изменение группы.

Макет окна изменения группы представлен на рисунке 4.6.

Элементами окна изменения группы являются:

- кнопка отмены, типа Button;
- кнопка создания, типа Button;
- информационное поля, типа TextView;
- поле ввода названия группы, типа EditText;
- поле ввода описания группы, типа EditText.

Рисунок 4.6 – Интерфейс окна изменения группы

Окно создания точки сбора содержит карту с перемещаемой меткой для выбора местоположения, две кнопки, с помощью которых можно отменить или подтвердить создание.

Макет окна создания точки сбора представлен на рисунке 4.7.

Элементами окна создания точки сбора являются:

- кнопка отмены, типа Button;
- кнопка создания, типа Button;
- карта, типа MapView.

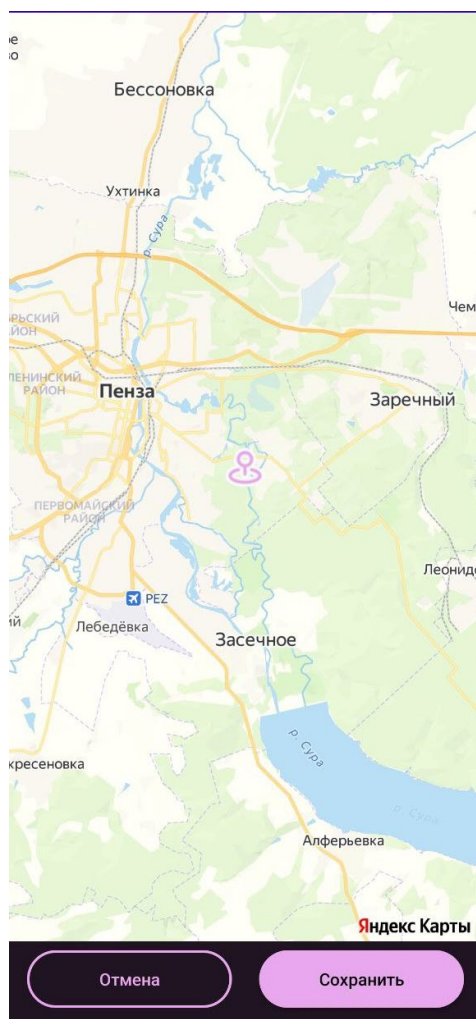


Рисунок 4.7 – Интерфейс окна создания точки сбора

Окно профиля содержит имя и email пользователя, а также три кнопки, с помощью которых можно вернуться на главное окно, изменить имя пользователя или выйти из аккаунта.

Макет окна профиля представлен на рисунке 4.8.

Элементами окна профиля являются:

- кнопка возврата на карту, типа Button;
- кнопка изменения имени, типа Button;
- кнопка выхода из аккаунта, типа Button;

- информационное поле с именем пользователя, типа TextView;
- информационное поле с email-ом пользователя, типа TextView.

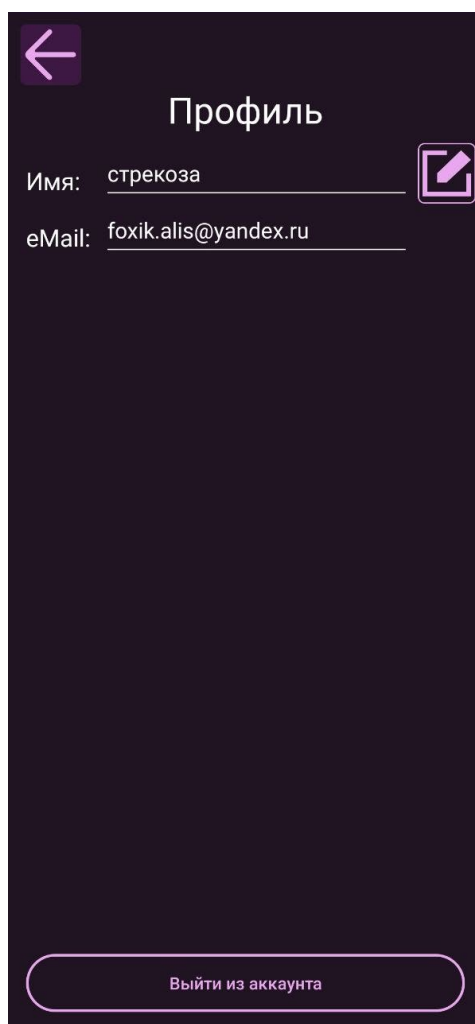


Рисунок 4.8 – Интерфейс окна профиля

Окно регистрации содержит четыре поля для ввода имени, email-а, пароля и подтверждения пароля, а также две кнопки, с помощью которых можно отменить или подтвердить регистрацию.

Макет окна регистрации представлен на рисунке 4.9.

Элементами главного окна являются:

- информационное поле, типа TextView;
- кнопка отмены, типа Button;
- кнопка регистрации, типа Button;
- поле ввода email-а, типа EditText;
- поле ввода имени, типа EditText;

- поле ввода пароля, типа EditText;
- поле ввода подтверждения пароля, типа EditText.

Рисунок 4.9 – Интерфейс окна регистрации

Окно авторизации содержит два поля для ввода email-а и пароля, а также две кнопки, с помощью которых можно отменить или подтвердить авторизацию.

Макет окна авторизации представлен на рисунке 4.10.

Элементами окна авторизации являются:

- информационное поле, типа TextView;
- кнопка отмены, типа Button;
- кнопка регистрации, типа Button;
- кнопка входа, типа Button;
- поле ввода eMail-а, типа EditText;
- поле ввода пароля, типа EditText.

Рисунок 4.10 – Интерфейс окна авторизации

Окно истории перемещений пользователя имя выбранного пользователя, карту с маршрутом перемещений, а также кнопку, с помощью которой можно вернуться на главное окно.

Макет окна истории перемещений пользователя представлен на рисунке 4.11.

Элементами окна истории перемещений пользователя являются:

- информационное поле для вывода имени пользователя, типа TextView;
- кнопка возврата в главное окно, типа Button;
- карта с маршрутом, типа MapView.

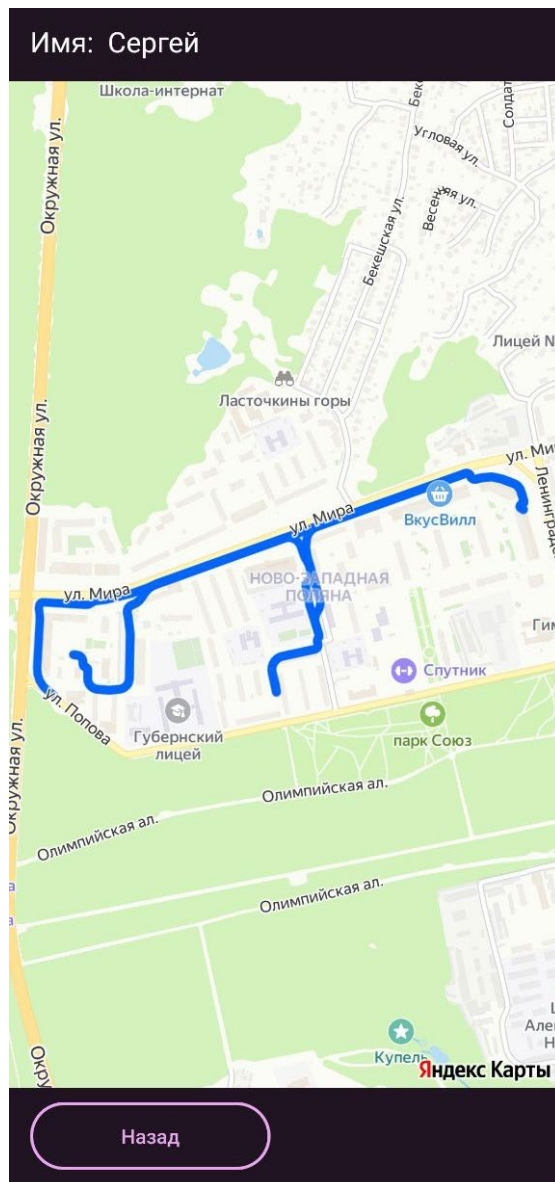


Рисунок 4.11 – Интерфейс окна истории перемещений пользователя

Окно маршрута до точки сбора содержит карту с маршрутом до точки, а также кнопку, с помощью которой можно вернуться на главное окно.

Макет окна маршрута до точки сбора представлен на рисунке 4.12.

Элементами окна маршрута до точки сбора являются:

- информационное поле, типа TextView;
- кнопка возврата в главное окно, типа Button;
- карта с маршрутом, типа MapView.

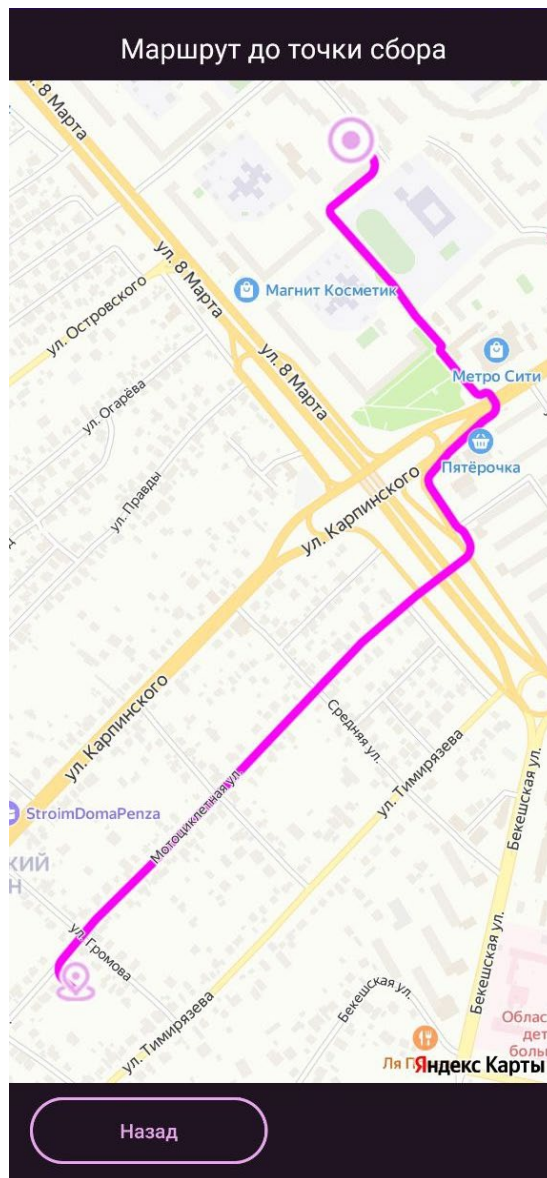


Рисунок 4.12 – Интерфейс окна маршрута до точки сбора

Диалоговое окно изменения имени пользователя содержит текстовое поле для ввода нового имени, а также две кнопки для отмены или подтверждения изменения имени.

Макет окна изменения имени представлен на рисунке 4.13.

Элементами диалогового окна изменения имени являются:

- поле ввода нового имени, типа EditText;
- кнопка отмены, типа Button;
- кнопка подтверждения, типа Button.

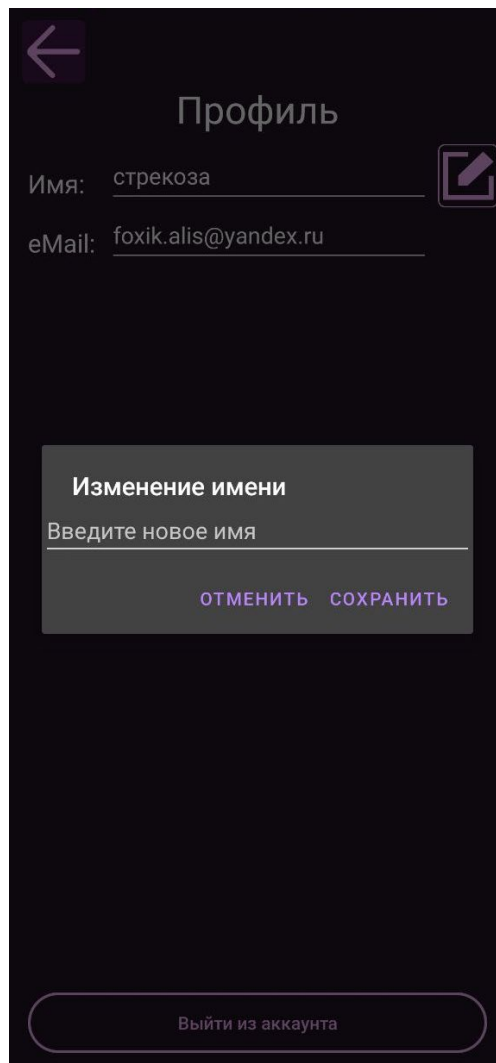


Рисунок 4.13 – Интерфейс диалогового окна изменения имени

Диалоговое окно удаления группы содержит две кнопки для отмены или подтверждения удаления группы.

Макет диалогового окна удаления группы представлен на рисунке 4.14.

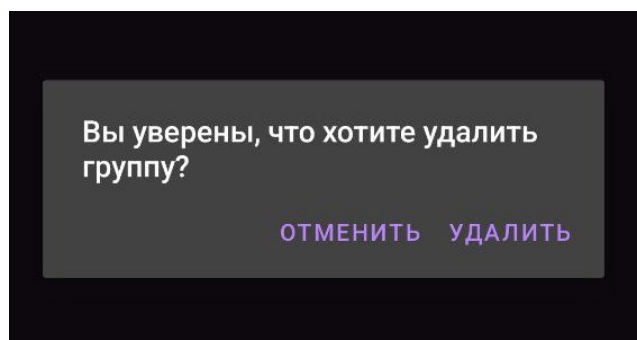


Рисунок 4.14 – Интерфейс диалогового окна удаления группы

Элементами диалогового окна удаления группы являются:

- кнопка отмены, типа Button;
- кнопка подтверждения, типа Button.

Диалоговое окно выхода из группы содержит две кнопки для отмены или подтверждения выхода из группы.

Макет диалогового окна выхода из группы представлен на рисунке 4.15.

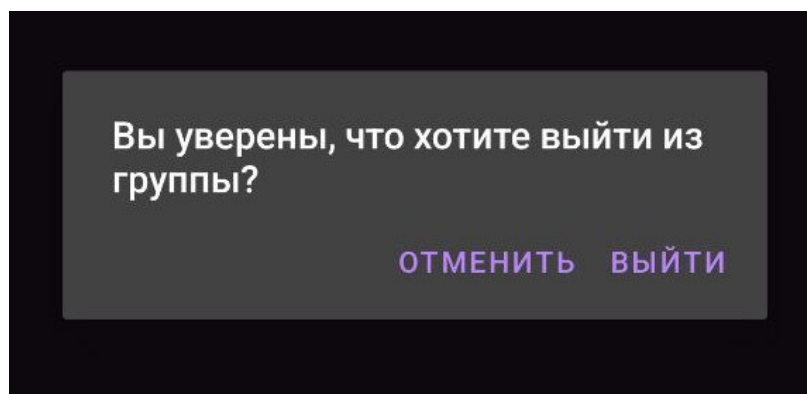


Рисунок 4.15 – Интерфейс диалогового окна выхода из группы

Элементами диалогового окна выхода из группы являются:

- кнопка отмены, типа Button;
- кнопка подтверждения, типа Button.

4.2 Аутентификации пользователей

Для аутентификации пользователей была использована платформа Firebase.

Аутентификация Firebase — это функция аутентификации пользователей, созданная на основе токенов, что обеспечивает легкую интеграцию с большинством платформ. Еще одна важная особенность заключается в том, что эта аутентификация легко интегрируется с API от Firebase, что делает возможным запустить эту функцию очень быстро и легко.

Для использования аутентификации Firebase необходимо зарегистрироваться или войти в профиль Firebase, после чего добавить проект. Далее в окне проекта необходимо добавить Authentication. Firebase автоматически создаст файл с расширением json с настройками подключения к сервису [6].

Для подключения сервиса аутентификации в проект необходимо в файле зависимостей вписать следующие строки:

```
implementation(platform("com.google.firebase:firebase-bom:32.8.1"))
implementation("com.google.firebase:firebase-auth")
```

Аутентификация использует метод `signInWithEmailAndPassword()`, принимающий на вход 2 строки электронной почты и пароля.

Код функции аутентификации пользователя представлен ниже:

```
btn_login.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String email = ((EditText)
findViewById(R.id.etEmail)).getText().toString();
        String password = ((EditText)
findViewById(R.id.etPassword)).getText().toString();
        if (email.equals(""))
            Toast.makeText(cnt, "Введите email",
Toast.LENGTH_SHORT).show();
        else if (password.equals(""))
            Toast.makeText(cnt, "Введите пароль",
Toast.LENGTH_SHORT).show();
        else
            signIn(email, password);
    }
});

private void signIn(String email, String password) {
    mAuth.signInWithEmailAndPassword(email,
password).addOnCompleteListener(ProfileActivity.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();

    }

}

});

}

```

Зарегистрированные учётные записи можно посмотреть в консоли Firebase, представленной на рисунке 4.16.

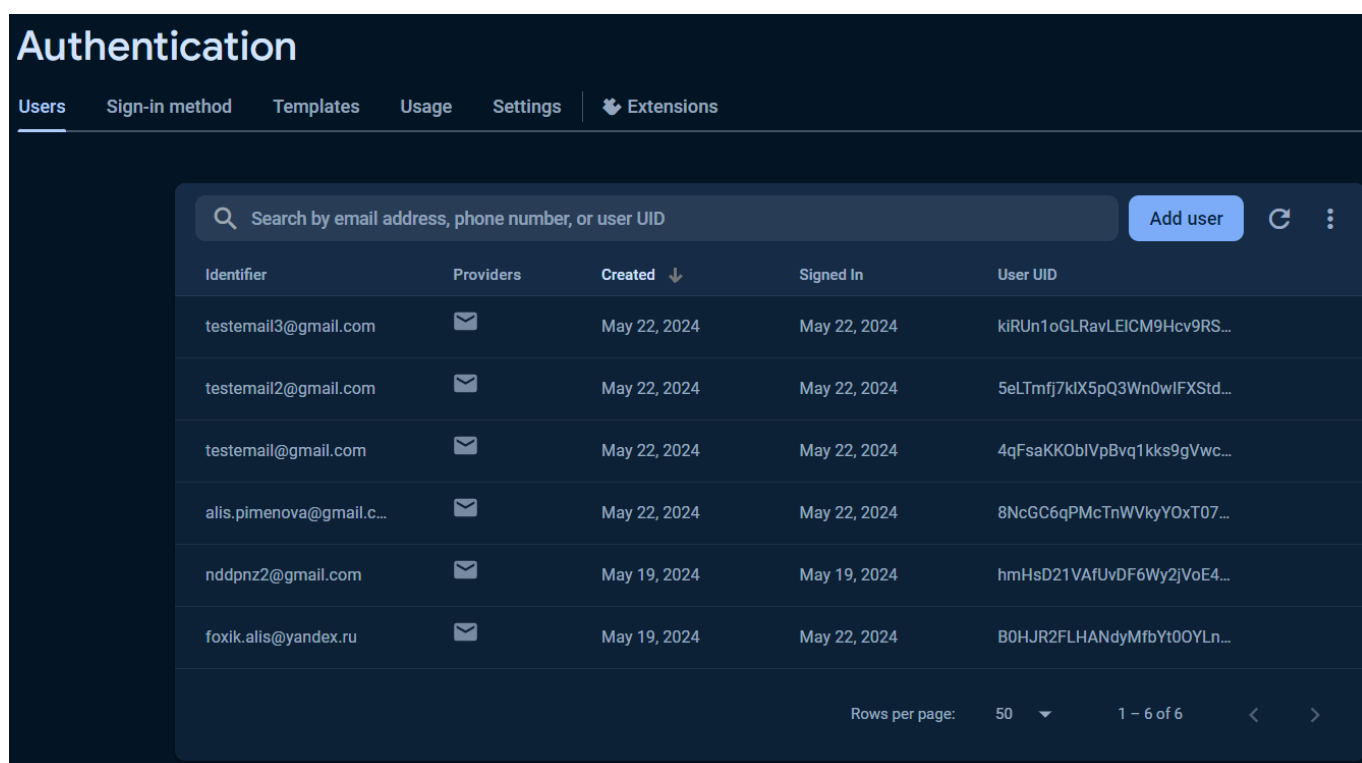


Рисунок 4.16 – Учетные записи пользователей в консоли Firebase

4.3 Реализация БД для хранения информации

Для реализации базы данных использовалась платформа Firebase. Для этого необходимо зарегистрироваться или войти в профиль Firebase, после чего добавить проект. Далее в окне проекта необходимо добавить Realtime Database. Интерфейс БД в Firebase представлен на рисунке 4.17.

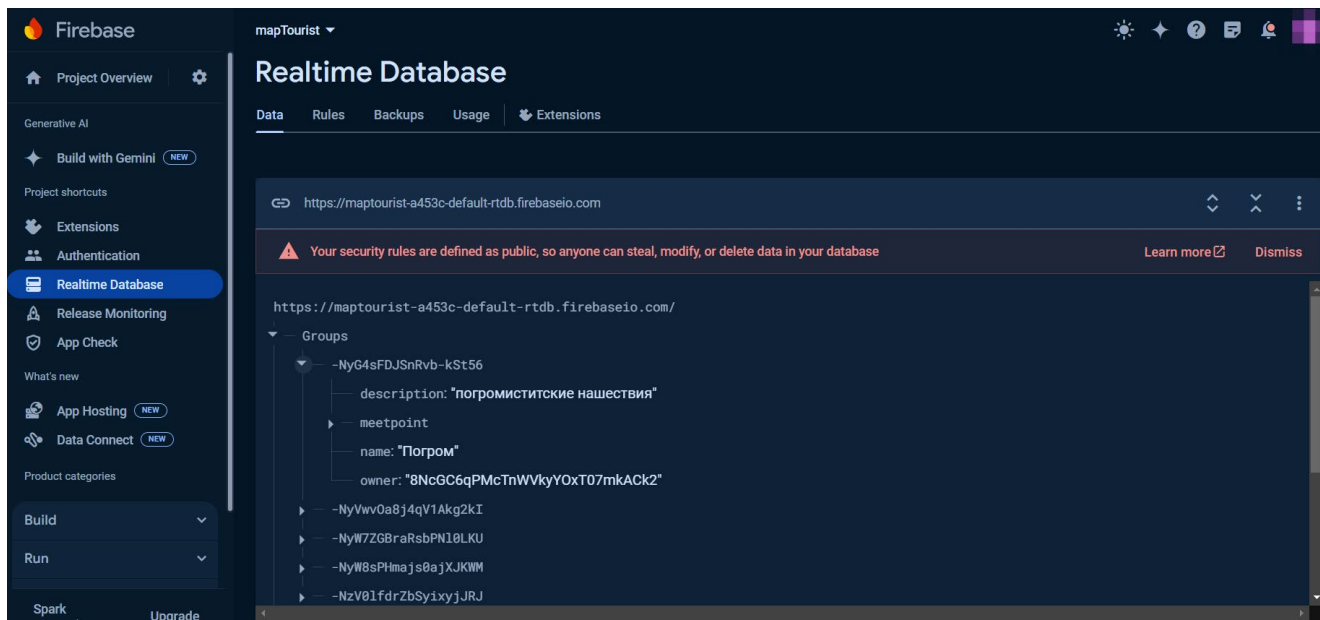


Рисунок 4.17 – База данных в Firebase

Firebase автоматически создаст файл с расширением json с настройками подключения к сервису.

Для подключения сервиса базы данных в проект необходимо в файле зависимостей вписать следующие строки:

```
implementation("com.google.firebase:firebase-database")
```

Для взаимодействий с базой данных необходимо инициализировать Firebase. Фрагмент кода представлен ниже.

```

FirebaseApp.initializeApp(getApplicationContext());
mAuth = FirebaseAuth.getInstance();
user = mAuth.getCurrentUser();
database = FirebaseDatabase.getInstance();
myRef = database.getReference();
if ((user != null)) {
    getUserInfo();
}

```

Для добавления информации в базу данных необходимо:

- инициализировать точку входа для доступа к БД;
- инициализировать определенное местоположение в базе данных;
- установить определенное значение по местоположению.

Фрагмент кода добавления пользователя в базу данных представлен ниже.

```

user = mAuth.getCurrentUser();
database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("People/" +
user.getId() + "/name");myRef.setValue(name);
myRef = database.getReference("People/" + user.getId() + "/eMail");
myRef.setValue(email);

```

Для извлечения информации из базы данных необходимо установить прослушиватель по определенному пути в базе данных. Прослушиватели бывают следующих типов:

- ListenerForSingleValueEvent – используется для одноразового извлечения данных из текущего местоположения;
- ValueEventListener – используется для мониторинга изменений данных из текущего местоположения;
- ChildEventListener – используется для мониторинга изменений дочерних элементов базы данных.

Фрагмент кода извлечения информации о пользователе с определенным UID из базы данных представлен ниже.

```

FirebaseDatabase.getInstance().getReference().child("People").child(
user.getId()).addListenerForSingleValueEvent(new
ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        String name =
dataSnapshot.child("name").getValue(String.class);
        String eMail =
dataSnapshot.child("eMail").getValue(String.class);
        if (dataSnapshot.hasChild("group")) {
            String groupId =
dataSnapshot.child("group").getValue(String.class);
            getGroupInfo(groupId);
        }
        me = new People(name, eMail, dataSnapshot.getKey());
    }
}

```

```

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

```

4.4 Работа с картами

Для работы с инструментом Yandex MapKit SDK необходимо получить уникальный API ключ. Для этого на сайте Яндекса необходимо зарегистрироваться и зайти в кабинет разработчика (рисунок 4.18). После чего выбрать необходимый API и получить ключ.

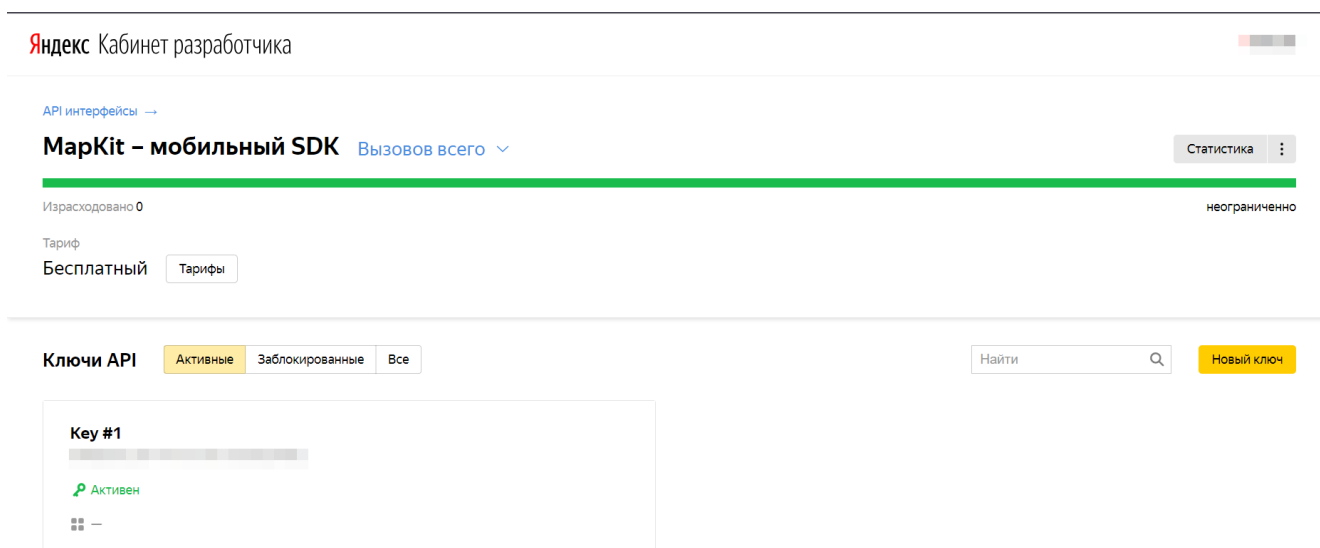


Рисунок 4.18 – Кабинет разработчика Yandex

Для добавления SDK к приложению, необходимо в файле зависимостей добавить следующую строку:

```
implementation 'com.yandex.android:maps.mobile:4.5.1-full'
```

Далее необходимо добавить инициализацию MapKit при запуске приложения. Фрагмент кода представлен ниже [7].

```

MapKitFactory.setApiKey(MAPKIT_API_KEY);
MapKitFactory.initialize(MainActivity.this);

```

Для отображения метки на карте необходимо инициализировать переменную объекта карты. После чего задать ей координаты, изображение и обработчик

события нажатия по ней. Фрагмент кода создания метки на карте представлен ниже.

```
PlacemarkMapObject meetpoint;
Point pnt = new
Point(snapshot.child("meetpoint").child("latitude").getValue(Double.
class),
snapshot.child("meetpoint").child("longitude").getValue(Double.class
));
meetpoint.setGeometry(pnt);
meetpoint.setVisible(true);
meetpoint.addTapListener(meetPointLst = new MapObjectTapListener() {
    @Override
    public boolean onMapObjectTap(@NonNull MapObject mapObject,
@NonNull Point point) {
        Intent rout = new Intent(MainActivity.this,
RouteToMeetpointActivity.class);
        rout.putExtra("long", Longitude);
        rout.putExtra("lat", Latitude);
        rout.putExtra("id", myGroup.id);
        startActivity(rout);
        return true;
    }
});
```

Для отображения маршрута необходимо задать в массиве точки маршрута. Точки маршрута бывают двух типов:

- WAYPOINT – точка конца или начала маршрута;
- VIAPOINT – промежуточная точка.

После этого создать запрос на пешеходный или автомобильный маршрут. После чего происходит создание маршрута по заданным точкам. Фрагмент кода создания пешеходного маршрута приведен ниже.

```
MapObjectCollection mapOb = null;
PedestrianRouter pedestrianRouter = null;
```

```

pedestrianRouter =
TransportFactory.getInstance().createPedestrianRouter();
mapOb = mapView.getMap().getMapObjects().addCollection();
submitRequest();
ArrayList<RequestPoint> requestPoints = new ArrayList<>();
TimeOptions timeOpt = new TimeOptions();
requestPoints.add(new RequestPoint(rout.get(0),
RequestPointType.WAYPOINT,null,null));
for (int i=1; i<rout.size()-1; i=i+(rout.size())/10)
{
    requestPoints.add(new RequestPoint(rout.get(i),
RequestPointType.VIAPOINT,null,null));
}
requestPoints.add(new RequestPoint(rout.get(rout.size()-1),
RequestPointType.WAYPOINT,null,null));
session = pedestrianRouter.requestRoutes(requestPoints, timeOpt,
this);

```

4.5 Реализация функционала приложения

4.5.1 Главное окно приложения

При запуске приложения выполняется инициализация Firebase и MapKit, проверяется есть ли у приложения разрешение на доступ к местоположению. В случае, если разрешение не предоставлено, приложение запрашивает его (рисунок 4.19). Фрагмент кода проверки и запроса разрешений представлен ниже:

```

if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED
        && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions(MainActivity.this,
        new
String[] {Manifest.permission.ACCESS_FINE_LOCATION},

```

```

1);

Log.d("Permission", "onCreate: Get permission");
}

```

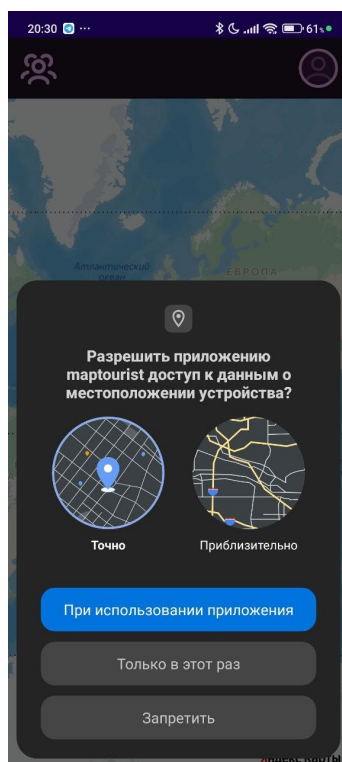


Рисунок 4.19 – Запрос на предоставление разрешения доступа к местоположению

После этого выполняется добавление пользовательского слоя на карту, на котором отображается метка текущего местоположения пользователя. Создается прослушиватель изменения состояния авторизации. Запускается фоновый поток, в котором в базу данных записывается информация о местоположении пользователя. Далее проверяется авторизован ли пользователь. Все последующие действия выполняются только для авторизованных пользователей. Выполняется извлечение информации об авторизованном пользователе из базы данных по UID, а именно имя, email, идентификатор группы (если есть). Если пользователь является членом группы, то устанавливается прослушиватель на изменение данных в корне Groups/groupid, где groupid – идентификатор группы. В прослушивателе выполняется мониторинг изменения названия, описания группы и точки сбора (при существующей точке сбора, она отображается на карте), а также запускается прослушиватель на изменение данных о пользователях, являющихся членами

данной группы. Для этого используется группировка по потомку group с приравниванием к идентификатору группы:

```
FirebaseDatabase.getInstance().getReference().child("People").orderByChild("group").equalTo(id).addValueEventListener();
```

Для каждого члена группы выполняется считывание местоположения из базы данных, после чего создается метка на карте.

При нажатии на кнопку профиля, пользователь переходит в окно профиля (если пользователь авторизован) или в окно авторизации (если пользователь не авторизован). При нажатии на кнопку группы, пользователь перейдет в окно группы (если он является членом группы), в окно выбора способа вступления в группу (если пользователь не является членом группы), если пользователь не авторизован, кнопка не активна.

При нажатии на метку пользователя, выполняется переход в окно истории местоположений пользователя. При нажатии на метку точки сбора, выполняется переход в окно с маршрутом от текущего местоположения к точке сбора.

4.5.2 Окно истории местоположения пользователя

В окне истории местоположения пользователя выполняется извлечение данных о местоположениях и имени пользователя из базы данных в массив координат, после чего с помощью инструментов MapKit выполняется построение маршрута от начальной до конечной точки через промежуточные точки. На форму выводится имя пользователя. Доступна кнопка «Назад», при нажатии на которую, выполняется переход на главную страницу приложения.

4.5.3 Окно маршрута до точки сбора

В окне маршрута до точки сбора выполняется извлечение данных о точке сбора группы из базы данных, после чего с помощью инструментов MapKit выполняется построение маршрута от начальной точки (текущего местоположения) до конечной точки (извлеченной из базы данных). Доступна кнопка «Назад», при нажатии на которую, выполняется переход на главную страницу приложения.

4.5.4 Группа окон аутентификации: окна профиля, авторизации и регистрации

При переходе к группе окон аутентификации, выполняется проверка авторизации пользователя. Если пользователь авторизован, то выполняется переход в окно профиля.

В окне профиля из базы данных извлекаются данные об авторизованном пользователе (имя, email) и вносятся в соответствующие текстовые поля. При нажатии на кнопку карты, выполняется переход на главную страницу. При нажатии на кнопку «Выйти из аккаунта» выполняется выход из аккаунта и возврат в окно авторизации. При нажатии на кнопку изменения имени, открывается диалоговое окно с полем ввода для нового имени и двумя кнопками: «Сохранить» и «Отменить». При нажатии на кнопку «Сохранить», выполняется внесение изменений в базу данных и закрытие диалогового окна. При нажатии на кнопку «Отменить», выполняется закрытие диалогового окна.

Если пользователь не авторизован, то выполняется переход в окно авторизации пользователя.

Окно авторизации имеет два текстовых поля ввода для электронной почты и пароля, а также три кнопки: кнопка возврата в главное окно, «Войти» и «Зарегистрироваться». При нажатии на кнопку карты, выполняется переход на главную страницу приложения. При нажатии на кнопку «Зарегистрироваться», выполняется переход в окно регистрации. При нажатии на кнопку «Войти», выполняется считывание информации из текстовых полей, после чего введенные значения передаются в функцию `signInWithEmailAndPassword()`. Для того, чтобы не передавать пустые строки в метод авторизации выполняется проверка на их заполнение. Устанавливается прослушиватель завершения функции аутентификации. В случае успешного завершения, выполняется переход на страницу профиля. Если функция завершилась с ошибкой, то выводится соответствующее сообщение и пользователь остается в окне авторизации.

Окно регистрации четыре текстовых поля ввода для электронной почты, пароля, подтверждения пароля и имени, а также три кнопки: кнопка возврата, «Зарегистрироваться». При нажатии на кнопку возврата, выполняется переход в окно авторизации. При нажатии на кнопку «Зарегистрироваться», выполняется проверка заполненности полей, соответствия значений полей пароля и подтверждения пароля, после чего вызывается функция `createUserWithEmailAndPassword`, в которую передаются email и пароль. Устанавливается прослушиватель окончания выполнения функции. Если функция выполнена успешно, то в базу данных вносятся данные о пользователе (UID, имя, email). Если функция завершилась с ошибкой, то выводится сообщение об ошибке и пользователь остается в окне регистрации.

4.5.5 Группа окон взаимодействия с группой: окна группы, выбора способа вступления в группу, создания группы

При переходе к группе окон взаимодействия с группой, выполняется проверка является ли пользователь членом группы.

Если пользователь не является членом группы, то выполняется переход в окно выбора способа вступления в группу.

В окне выбора способа вступления в группу доступны три кнопки:

- карта, при нажатии на которую, выполняется переход на главную страницу приложения;
- «Создать группу», при нажатии на которую, выполняется переход в окно создания группы;
- «Вступить в группу», при нажатии на которую, выполняется переход в окно выбора группы.

В окне создания группы доступны два поля ввода для названия и описания группы и две кнопки:

- «Отмена», при нажатии на которую, выполняется переход в окно выбора способа вступления в группу;

– «Создать», при нажатии на которую, выполняется внесение названия и описания группы в базу данных, после чего осуществляется переход в окно группы.

В окне вступления в группу доступны список групп, заполнение которого выполняется с помощью извлечения из базы данных информации о всех группах, и кнопка назад. При нажатии на кнопку назад, выполняется переход в окно выбора способа вступления в группу. При нажатии на элемент списка групп, выполняется внесение идентификатора группы в потомок «group» авторизованного пользователя, после чего осуществляется переход в окно группы.

Если пользователь является членом группы, то выполняется переход в окно группы.

В окне группы выполняется считывание из БД информации о пользователе, считывается и выводится информация о группе и членах группы, проверяется, является ли пользователь владельцем группы.

Если пользователь не является владельцем группы, ему доступны кнопки выхода из группы и возврата в главное окно приложения. При нажатии на кнопку карты, выполняется переход на главную страницу приложения. При нажатии на кнопку выхода из группы, выполняется выход из группы (удаления потомка «group» данного пользователя) и переход на главное окно приложения.

Если пользователь является владельцем группы, то у него не активна кнопка выхода из группы, но доступно выпадающее меню со следующими пунктами:

– «Удалить группу», при нажатии на который выводится диалоговое окно с двумя кнопками: «Удалить», «Отменить». При нажатии на кнопку «Удалить» выполняется удаление группы из базы данных и переход в главное окно приложения. При нажатии на кнопку «Отменить» выполняется закрытие диалогового окна;

– «Изменить группу», при нажатии на который выполняется переход в окно редактирования группы;

– «Создать точку сбора» (если точка сбора не была установлена), при нажатии на которую выполняется переход в окно создания точки сбора;

– «Удалить точку сбора» (если точка сбора была установлена), при нажатии на которую, выводится диалоговое окно с двумя кнопками: «Удалить», «Отменить». При нажатии на кнопку «Удалить» выполняется удаление точки из базы данных и переход в главное окно приложения. При нажатии на кнопку «Отменить» выполняется закрытие диалогового окна.

При переходе в окно изменения группы, информация о названии и описании группы извлекается из базы данных и помещается в соответствующие поля ввода. В окне доступны две кнопки:

- «Отменить», при нажатии на которую, выполняется переход в окно группы;
- «Изменить», при нажатии на которую, выполняется внесение изменений в базу данных, после чего осуществляется переход в окно группы.

В окне создания точки сбора доступны две кнопки «Отменить» и «Сохранить» и карта с меткой точки сбора по центру. При перемещении карты, перемещается метка. При нажатии на кнопку «Отмена» выполняется возврат в окно группы. При нажатии на кнопку «Сохранить», выполняется внесение координат точки сбора в базу данных, после чего осуществляется переход в главное окно приложения.

5 Технико-экономическое обоснование

Себестоимость – это все затраты на производство и реализацию ПО. Себестоимость продукции включает в себя стоимость материалов, заработную плату сотрудников, стоимость оборудования и так далее.

Планирование себестоимости важная часть процесса производства товара. Основными целями расчета себестоимости являются:

- определение цены товара. Себестоимость сильно влияет на ценообразование, так как цена на товар или услугу должна покрыть расходы на производство и включать прогнозируемую прибыль;
- контроль расходов. Благодаря расчету себестоимости можно отследить, какие затраты можно снизить;
- оценка эффективности. Сравнение запланированной себестоимости с фактическими затратами позволяет определить окупается ли продукт и что можно улучшить в производстве.

Существует несколько способов калькуляции себестоимости:

- плановая – рассчитывается на основе прогнозов и экспертных оценок. Данный вид себестоимости используется при подготовке финансовых планов на определенный период или для расчета цены за товар или услугу;
- отчетная – себестоимость, которая рассчитывается после производства продукта по фактическим затратам. Зачастую это делается перед окончанием отчетного периода. Эти расчеты необходимы, в том числе, чтобы сравнить нормативные или плановые затраты с реальными и, в случае сильных отличий, корректировать способ расчета плановой/нормативной себестоимости;
- нормативная – себестоимость, рассчитанная на основе стандартных показателей и заранее определенных нормативов для каждой статьи расходов, в том числе стандартного количества рабочих часов, необходимых на разработку ПО.

Для расчета плановой себестоимости программного обеспечения необходимо учитывать следующие параметры:

- затраты на содержание помещения;
- затраты на приобретение оборудования;
- затраты на приобретение лицензионного программного обеспечения;
- затраты на дополнительные материалы;
- затраты на заработную плату;
- затраты на страховые взносы;
- затраты на обслуживание.

Разработка ПО осуществлена на базе собственного производственного помещения общей площадью 36 кв. м. Содержание помещения в месяц составляет 11 000 рублей. Затраты в период разработки (2 месяца) составят 22000 рублей.

Для разработки ПО приобретена вычислительная техника, лицензионное программное обеспечение и расходные материалы.

Минимальная необходимая конфигурация персонального компьютера:

- модель процессора – Intel Core i5;
- общее количество ядер – 8;
- общий объем оперативной памяти – 16 ГБ;
- модель дискретной видеокарты – GeForce RTX 3050;
- общий объем твердотельных накопителей – 512 ГБ.

Исходя из минимальной конфигурации, был выбран компьютер MSI MAG Infinite S3.

Данные по потребности в оборудовании представлены в таблице 5.1.

Таблица 5.1 – Оборудование для разработки

Оборудование	Источник	Кол-во	Ориентировочная цена за единицу, руб.	Потребляемая мощность, Ватт
ПК MSI MAG Infinite S3	ООО «ДНС»	5	90000	500
Клавиатура	ООО «ДНС»	5	2900	1,7
Мышь	ООО «ДНС»	5	1800	0,8
Монитор ASUS VY249NE	ООО «ДНС»	5	11000	16

На основании таблицы 5.1 затраты на оборудование в период разработки составят 528500 рублей.

Для разработки приобретено лицензионное программное обеспечение для разработки и сопровождения приложения. Данные по потребности в лицензионном ПО представлены в таблице 5.2.

Таблица 5.2 – Лицензионное программное обеспечение для разработки

Лицензионное программное обеспечение	Источник	Цена, руб.	Срок действия лицензии
Microsoft Windows 11 pro	«Ситилинк»	30000	бессрочно
MapKit SDK	Яндекс	45000	1 месяц
Firebase	Google	29000	1 месяц

На основании таблицы 5.2 затраты на лицензионное программное обеспечение (в расчете на 2 месяца разработки) составят 148000 рублей.

Для разработки приобретены дополнительные материалы. Перечень дополнительных материалов представлен в таблице 5.3.

Таблица 5.3 – Перечень дополнительных материалов

Материал	Источник	Количество, штук	Цена за единицу, рублей
Стол компьютерный	ОФИСМАГ (ГК «Самсон»)	5	3500
Кресло	ОФИСМАГ (ГК «Самсон»)	5	9000
Лампа настольная	ОФИСМАГ (ГК «Самсон»)	5	1500
Канцтовары	ОФИСМАГ (ГК «Самсон»)	5	1000
Бумага	ОФИСМАГ (ГК «Самсон»)	5	600

На основании таблицы 5.3 затраты на дополнительные материалы составят 78000 рублей.

Разработка программного обеспечения осуществлялась командой разработчиков. Состав коллектива зависит от поставленных задач. Минимальный базовый набор специалистов: разработчик, тестировщик, руководитель проекта. Так как разрабатываемое приложение должно иметь интерфейс, то в команде присутствует дизайнер. Состав коллектива и заработная плата разработчиков ПО представлена в таблице 5.4.

Таблица 5.4 – Занятость и заработная плата разработчиков

Специалист	Количество, человек	Зарботная плата, рублей
Руководитель проекта	1	120000
Программист	2	100000
Тестировщик	2	95000
Дизайнер	1	100000

Затраты на заработную плату включают в себя:

– затраты на основную заработную плату разработчика, исходящие из установленной заработной платы, занятости специалистов в процессе разработки программного продукта (в днях);

– затраты на дополнительную заработную плату, составляющую 15% от затрат на основную заработную плату.

Учитывая пятидневную рабочую неделю, количество рабочих дней в месяце в среднем равно 21.

Затраты на основную и дополнительную заработные платы разработчиков представлены в таблице 5.5.

Таблица 5.5 – Затраты на основную и дополнительную заработные платы

Исполнитель	Количество, человек	Сумма заработной платы, рублей/человек	Период занятости, дней	Основная заработная плата, рублей/человек	Дополнительная заработная плата, рублей/человек	Всего рублей
Руководитель проекта	1	120000	42	240000	36000	276000
Программист	2	100000	42	200000	30000	460000
Тестировщик	2	95000	42	190000	28500	437000
Дизайнер	1	100000	42	200000	30000	460000
Итого	6	610000	210	1220000	183000	1633000

Согласно таблице 5.5, затраты на заработную плату разработчиков составят 1633000 рублей.

Согласно статье 419 НК РФ работодатель обязан выплачивать отчисления на страховые взносы сотрудников. Отчисления на страховые взносы по нормам, установленным законодательством в ПФР, ФОМС, ФСС РФ рассчитываются согласно «Налоговому кодексу Российской Федерации (часть вторая)» от 05.08.2000 N 117-ФЗ (ред. от 29.05.2024) НК РФ Раздел XI. Страховые взносы в Российской Федерации.

Единый тариф (единые размеры) страховых взносов для плательщиков, производящих выплаты физическим лицам, установлен на:

- обязательно пенсионное страхование (ОПС);
- обязательное социальное страхование на случай временной нетрудоспособности и в связи с материнством (ОСС);
- обязательное медицинское страхование (ОМС).

В пределах установленной единой предельной величины базы для исчисления страховых взносов единый тариф страховых взносов равен 30%, свыше – 15,1% [8].

Суммарные отчисления на страховые взносы составят 489900 рублей.

Общая сумма затрат на разработку продукта составляет 2899400 рублей.

Ежемесячные затраты на обслуживание ПО зависят от количества пользователей. Примем изначальное количество пользователей равным 20000, в этом случае ежемесячные затраты на продление лицензионного программного обеспечения составят 74000 рублей.

					<i>ПГУ 1.090501.08.001 ПЗ</i>	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

6 Безопасность жизнедеятельности

Правила техники безопасности регламентируются государством с помощью нормативных документов. Они необходимы чтобы минимизировать вредное воздействие на организм человека при различных видах работ. Для профессии инженера-программиста вредными факторами являются: сидячая работа и воздействие компьютера на организм человека. Из нормативных документов можно выделить:

- санитарно-гигиенические правила и нормативы СанПин 2.2.2/2.4.1340-03;
- трудовой кодекс;
- межгосударственный стандарт ГОСТ 12.0.003-2015, определяющий и классифицирующий основные вредные и опасные производственные факторы.

6.1 Требования к рабочему месту инженера-разработчика

При работе за персональным компьютером присутствует большая нагрузка на зрение и мышцы. Для того, чтобы минимизировать негативное воздействие, необходимо обустроить рабочее место пользователя для безопасной позы. Для снижения нагрузки на зрительный и мышечный аппарат необходимо придерживаться следующих правил:

- расстояние от глаз до монитора должно быть в пределах от 45 до 70 сантиметров;
- верхняя часть экрана должна находиться на уровне глаз или немного ниже;
- во время работы на экране не должны появляться блики или отражения;
- плечи должны быть расслаблены;
- локти и кисти рук должны находиться на одной высоте;
- пространство под рабочей поверхностью должно быть свободно;
- ступни стоят на полу или подставке;
- угол между голенью и бедром должен составлять 90 градусов;
- угол между бедром и телом должен быть от 100 до 130 градусов;

– если на столе достаточно места, то весь вспомогательные вещи должны лежать вокруг компьютера.

Правильное расположение пользователя за компьютером представлено на рисунке 6.1.

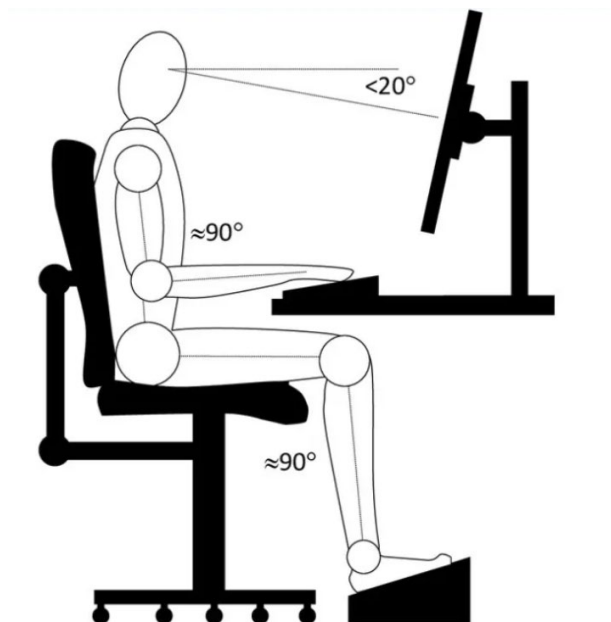


Рисунок 6.1 – Расположение пользователя за персональным компьютером

Кроме правильного расположения за рабочим столом, необходимо каждые 25-30 минут делать легкую разминку и менять положение ног, а также выполнять разминку для глаз. Для большего комфорта глаз можно использовать специальные очки.

Согласно санитарно-эпидемиологическим требованиям к условиям труда на рабочем месте, предназначенном для работы в положении сидя, производственное оборудование и рабочие столы должны иметь пространство для размещения ног высотой не менее 600 мм, глубиной - не менее 450 мм на уровне колен и 600 мм на уровне стоп, шириной не менее 500 мм.

Для лиц, работающих 12 и более часов (при наличии перерыва на сон), должно быть оборудовано место для сна и принятия горячей пищи.

6.2 Требования к микроклимату в рабочем помещении

6.2.1 Требования к шуму

Нормативами, используемыми для оценки уровня шума на рабочих местах, являются:

– эквивалентный уровень звука (L_{pAeqT} , дБА) – уровень воздействия шума за рабочую смену (8 часов). Нормативным эквивалентным уровнем звука является 80 дБА;

– максимальные уровни звука A , измеренные с временными коррекциями S и I ($L_{pA\ max}$) – наибольшая величина уровня звука, измеренная на заданном интервале времени. Максимальными уровнями звука A , измеренными с временными коррекциями S и I , являются 110 дБА и 125 дБА соответственно;

– пиковый скорректированный по C уровень звука ($L_{pC\ peak}$), дБС – C – взвешенное наибольшее значение за время измерений. Пиковым скорректированным по C уровнем звука ($L_{pC\ peak}$) является 137 дБС.

6.2.2 Требования к влажности

Показатель влажности должны варьироваться от 40 до 60%. Допустимо отклонение от нормы 10%. Таким образом допустимый диапазон влажности воздуха составляет от 30 до 70%. При низкой влажности у человека появляется сухость слизистых оболочек дыхательных путей и кожи, а при повышенной влажности усиливается потоотделение и становится душно.

Для корректировки влажности воздуха в рабочем помещении необходимо использовать увлажнитель и осушитель воздуха, которые могут быть как отдельными устройствами, так и являться частью состава системы контроля микроклимата (например, сплит-системы).

6.2.3 Требования к температуре

Температура в офисном помещении зависит от времени года. В холодное время года температура должна варьироваться в пределах от 22 °С до 23 °С. В теплое время года – от 22 °С до 25 °С. При этом допустимый диапазон температур в холодное время года 19-24 °С, а в теплое время года 20-29 °С [9].

					ПГУ 1.090501.08.001 ПЗ	Лист
						63
Изм.	Лист	№ докум.	Подпись	Дата		

Для регуляции температуры в помещении необходимо использовать сплит-систему кондиционирования с функциями охлаждения и нагрева воздуха. Необходимо учитывать, что температура и влажность взаимосвязаны, поэтому желателен постоянный мониторинг этих показателей. Одним из вариантов решения проблемы контроля микроклимата является автоматизированная система управления микроклиматом.

6.3 Окраска и коэффициенты отражения

Окраска помещений и мебели должна исключать зрительное напряжение и способствовать созданию благоприятных условий для зрительного восприятия.

Источники света, дающие отражение от поверхности экрана, ухудшают точность знаков и влекут за собой помехи физиологического характера, которые могут повлечь значительное напряжение, особенно при продолжительной работе. Отражение, включая отражения от вторичных источников света, должно быть сведено к минимуму. Для защиты от избыточной яркости окон могут быть применены шторы, экраны или другие средства, препятствующие проникновению солнечного света.

Важным аспектом освещения помещений является окраска стен, потолка и пола. Разные цвета по-разному влияют на восприятие помещения. Так, например, солнечные лучи, падающие на предмет холодного оттенка, смягчают его и делают более теплым. В зависимости от ориентации окон рекомендуется следующая окраска стен и пола:

- окна ориентированы на юг, стены зеленовато-голубого или светло-голубого цвета, пол зеленый;
- окна ориентированы на север, стены светло-оранжевого или оранжево-желтого цвета, пол красновато-оранжевый;
- окна ориентированы на восток, стены желто-зеленого цвета, пол зеленый или красновато-оранжевый;

— окна ориентированы на запад, стены желто-зеленого или голубовато-зеленого цвета; пол зеленый или красновато-оранжевый.

В помещениях, где находится компьютер, необходимо обеспечить следующие величины коэффициента отражения:

— для потолка 60...70%;

— для стен 40...50%;

— для пола около 30%.

Для других поверхностей и рабочей мебели: 30...40%.

					<i>ПГУ 1.090501.08.001 ПЗ</i>	Лист
						65
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В процессе выполнения выпускной квалификационной работы была изучена предметная область, сформулировано техническое задание на разработку приложения для трекинга распределенных групп людей.

Был произведен анализ и выбор инструментария, в частности выбор облачного сервиса для реализации базы данных. Выполнено проектирование архитектуры приложения и базы данных, в результате которого были разработаны UML диаграммы разрабатываемого ПО и структура базы данных.

Произведено проектирование и реализация частей системы, в результате которого были разработаны блоки интерфейса приложения, логического слоя приложения и модули взаимодействия с СУБД.

Программа реализована в соответствии с техническим заданием.

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

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

Задание на дипломное проектирование выполнено в полном объеме.

					<i>ПГУ 1.090501.08.001 ПЗ</i>	Лист
						66
Изм.	Лист	№ докум.	Подпись	Дата		

Список используемых источников

1. Воскресенскова, А. Ю. Виды активного туризма и их влияние на здоровье человека / А. Ю. Воскресенскова, Е. И. Коробейникова // Наука-2020. – 2020. – № 2(38). – С. 72-79. – EDN WAARGI.
2. Алалалыкин, Д. Т. Разработка информационной системы для автоматизации учета успеваемости обучающихся / Д. Т. Алалалыкин, П. Ю. Бугаков // Интерэкспо Гео-Сибирь. – 2023. – Т. 7, № 1. – С. 130-137. – EDN XUXXUQ.
3. Яндекс.Карты, 2ГИС или всё же Google Maps? / Хабр [Электронный ресурс]: URL: <https://habr.com/ru/articles/242015/> (дата обращения 17.05.2024)
4. Аксенов, К. В. Обзор современных средств для разработки мобильных приложений / К. В. Аксенов // Новые информационные технологии в автоматизированных системах. – 2014. – № 17. – С. 508-513. – EDN SDBZDT.
5. Developer guides | Android Developers [Электронный ресурс]: URL: <https://developer.android.com/guide> (дата обращения 17.05.2024)
6. Документация | Firebase Documentation [Электронный ресурс]: URL: <https://firebase.google.com/docs> (дата обращения 17.05.2024)
7. О MapKit | Mapkit Mobile SDK [Электронный ресурс]: URL: <https://yandex.ru/dev/mapkit/doc/ru/> (дата обращения 17.05.2024)
8. «Налоговый кодекс Российской Федерации (часть вторая)» от 05.08.2000 N 117-ФЗ (ред. от 29.05.2024) (с изм. и доп., вступ. в силу с 01.06.2024)
9. Постановление государственного санитарного врача РФ от 28.01.2021 N 2 (ред. от 30.12.2022) «Об утверждении санитарных правил и норм СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания»

Приложение А

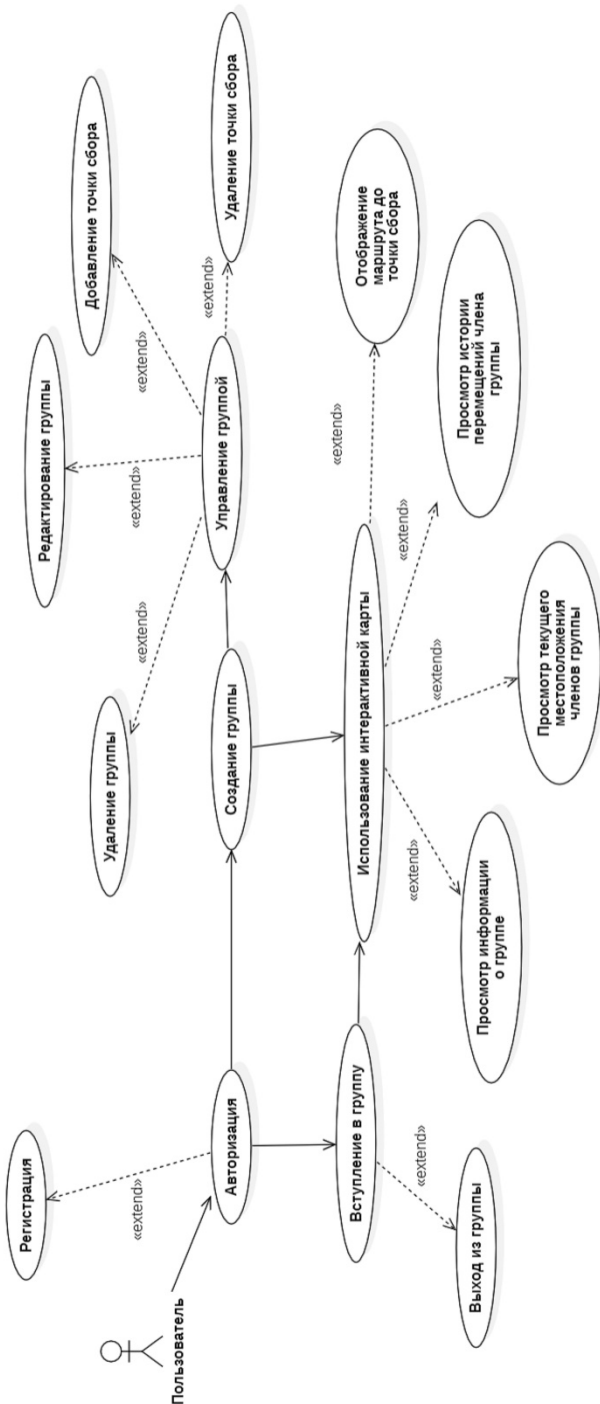
(обязательное)

Программа для трекинга распределенных групп людей

UML-диаграммы

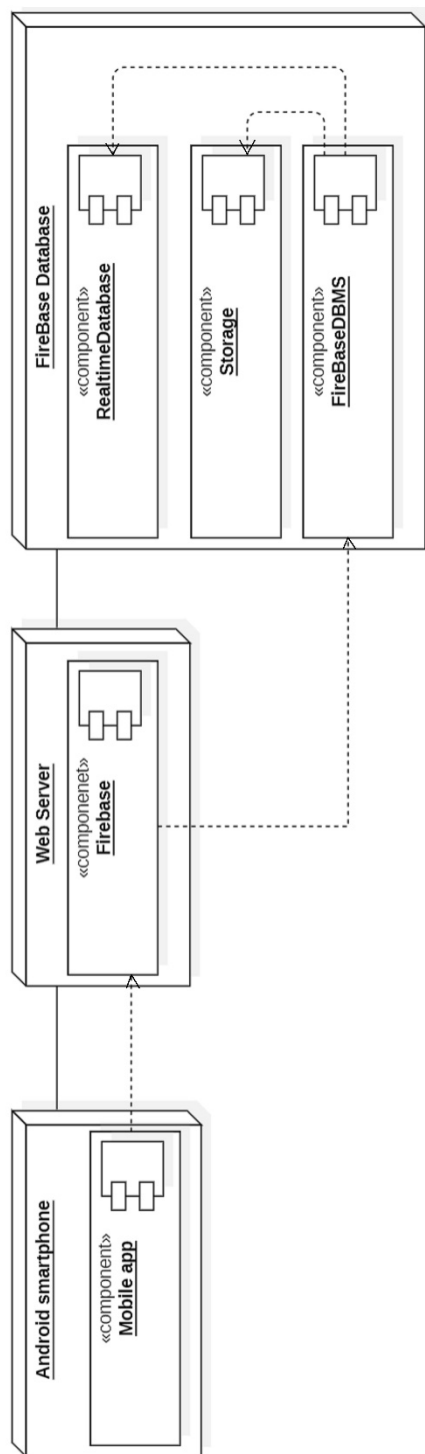
					<i>ПГУ 1.090501.08.001 ПЗ</i>	Лист
						68
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1 – Диаграмма вариантов использования



ПГУ 109050108.002 ПЗ									
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель

Приложение А.2 – Диаграмма развертывания



ПГУ 1.090501.08.003 Д19				
Документ для разработки и внедрения				
содержит: Модель				
Диаграмма развертывания				
ПГУ 1.090501.08.003 Д19				
20.10.2021				

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.08.001 ПЗ

Приложение А.3 – Диаграммы деятельности

Диаграмма деятельности авторизация

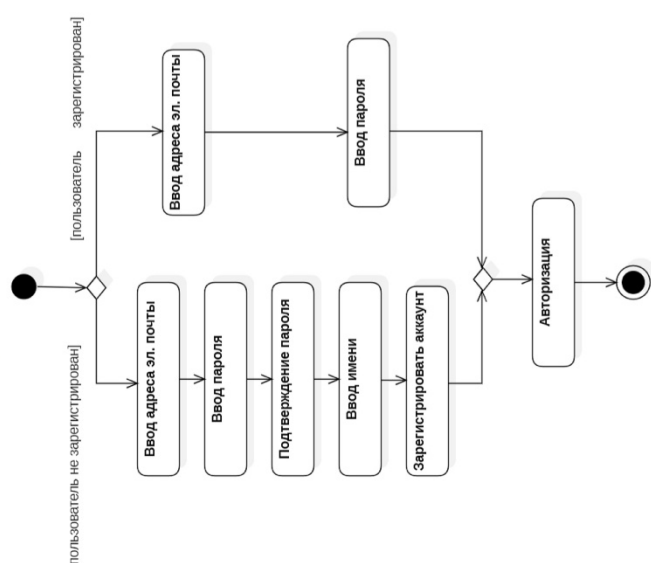


Диаграмма деятельности создание группы

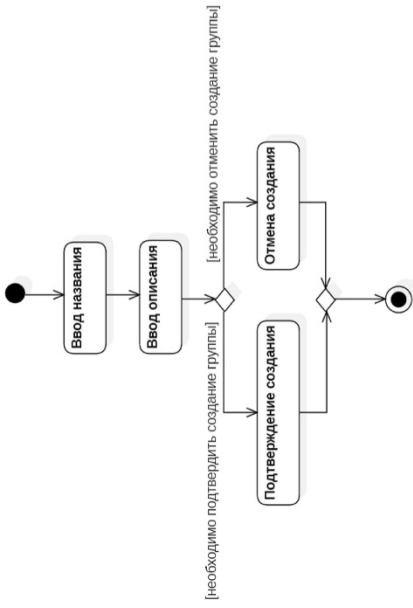
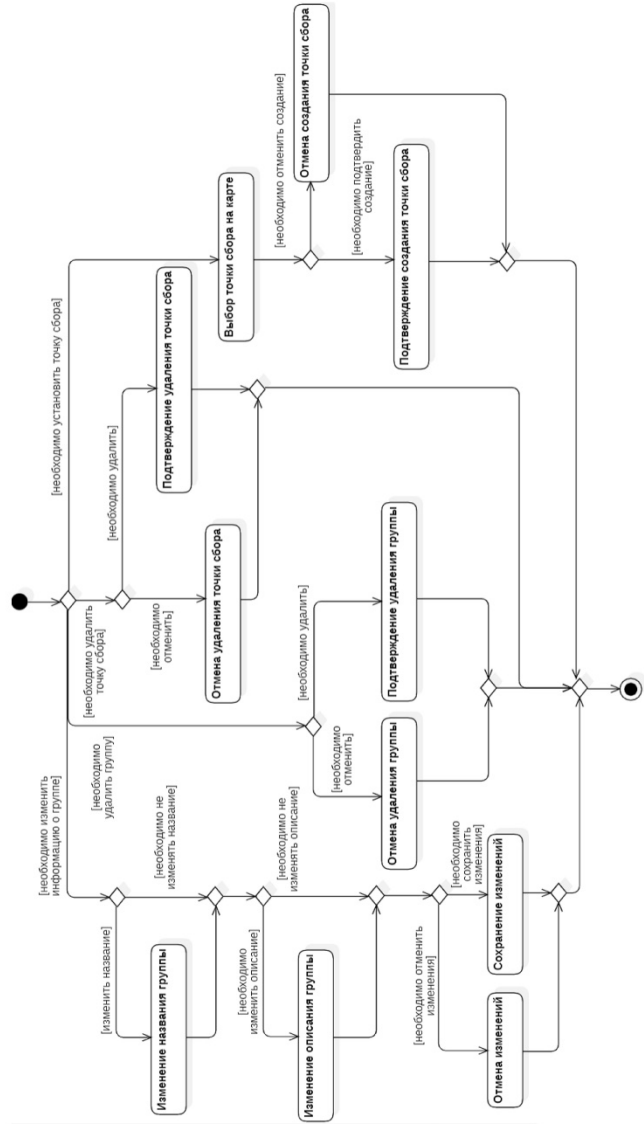
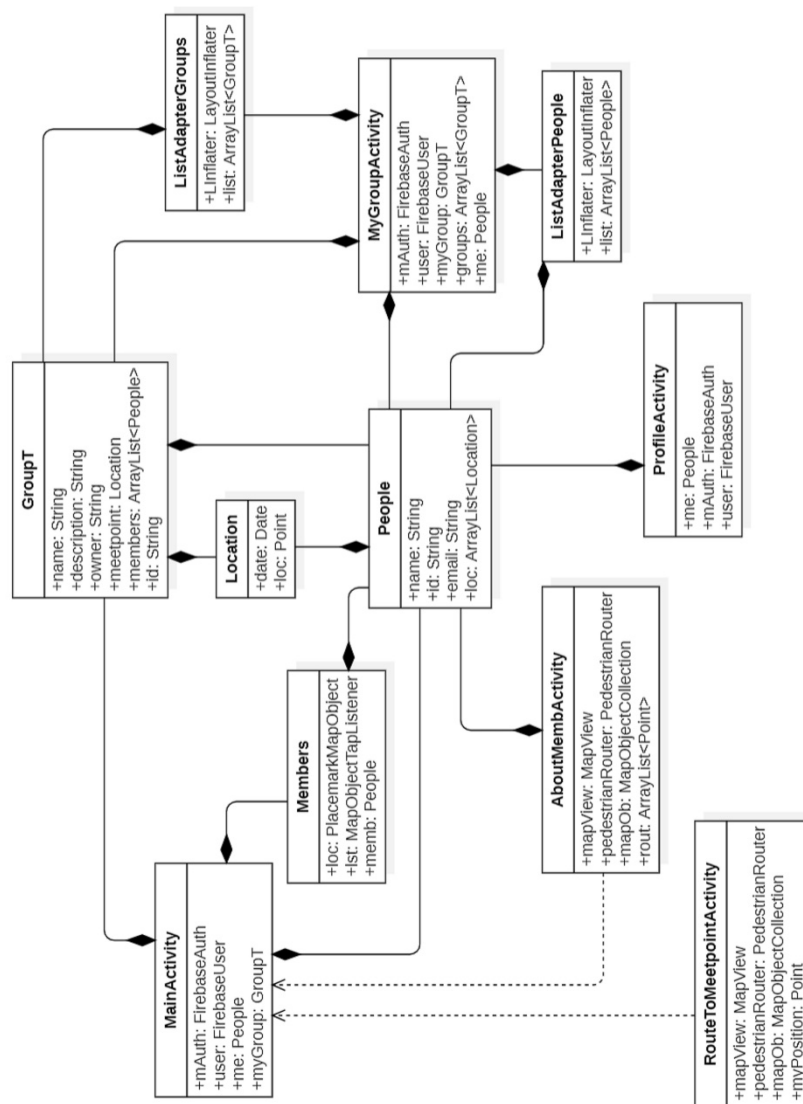


Диаграмма деятельности управление группой

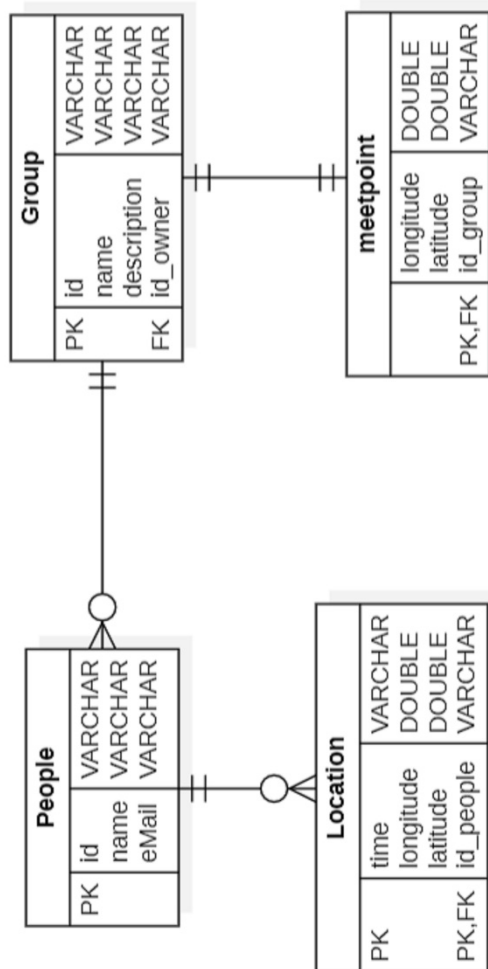


ПГУ 109050108.004 Д17		Вариант	Итого	Всего
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1
5	1	1	1	1
6	1	1	1	1
7	1	1	1	1
8	1	1	1	1
9	1	1	1	1
10	1	1	1	1
11	1	1	1	1
12	1	1	1	1
13	1	1	1	1
14	1	1	1	1
15	1	1	1	1
16	1	1	1	1
17	1	1	1	1
18	1	1	1	1
19	1	1	1	1
20	1	1	1	1
21	1	1	1	1
22	1	1	1	1
23	1	1	1	1
24	1	1	1	1
25	1	1	1	1
26	1	1	1	1
27	1	1	1	1
28	1	1	1	1
29	1	1	1	1
30	1	1	1	1
31	1	1	1	1
32	1	1	1	1
33	1	1	1	1
34	1	1	1	1
35	1	1	1	1
36	1	1	1	1
37	1	1	1	1
38	1	1	1	1
39	1	1	1	1
40	1	1	1	1
41	1	1	1	1
42	1	1	1	1
43	1	1	1	1
44	1	1	1	1
45	1	1	1	1
46	1	1	1	1
47	1	1	1	1
48	1	1	1	1
49	1	1	1	1
50	1	1	1	1
51	1	1	1	1
52	1	1	1	1
53	1	1	1	1
54	1	1	1	1
55	1	1	1	1
56	1	1	1	1
57	1	1	1	1
58	1	1	1	1
59	1	1	1	1
60	1	1	1	1
61	1	1	1	1
62	1	1	1	1
63	1	1	1	1
64	1	1	1	1
65	1	1	1	1
66	1	1	1	1
67	1	1	1	1
68	1	1	1	1
69	1	1	1	1
70	1	1	1	1
71	1	1	1	1
72	1	1	1	1
73	1	1	1	1
74	1	1	1	1
75	1	1	1	1
76	1	1	1	1
77	1	1	1	1
78	1	1	1	1
79	1	1	1	1
80	1	1	1	1
81	1	1	1	1
82	1	1	1	1
83	1	1	1	1
84	1	1	1	1
85	1	1	1	1
86	1	1	1	1
87	1	1	1	1
88	1	1	1	1
89	1	1	1	1
90	1	1	1	1
91	1	1	1	1
92	1	1	1	1
93	1	1	1	1
94	1	1	1	1
95	1	1	1	1
96	1	1	1	1
97	1	1	1	1
98	1	1	1	1
99	1	1	1	1
100	1	1	1	1

Приложение А.4 – Диаграмма классов

[illegible]

Приложение А.5 – Диаграмма данных

[illegible]

Приложение Б

(обязательное)

Программа для трекинга распределенных групп людей

Листинг программы

					<i>ПГУ 1.090501.08.001 ПЗ</i>	Лист
						74
Изм.	Лист	№ докум.	Подпись	Дата		

MainActivity.java

```
package com.example.maptourist;

import android.Manifest;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.PointF;
import android.icu.text.SimpleDateFormat;
import android.icu.util.Calendar;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Build;
import android.os.Bundle;
import android.os.Looper;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import com.google.firebase.FirebaseApp;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.ChildEventListener;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
```

					<i>ПГУ 1.090501.08.001 ПЗ</i>	Лист
						75
Изм.	Лист	№ докум.	Подпись	Дата		

```

import com.yandex.mapkit.MapKit;
import com.yandex.mapkit.MapKitFactory;
import com.yandex.mapkit.geometry.Point;
import com.yandex.mapkit.layers.ObjectEvent;
import com.yandex.mapkit.map.CameraPosition;
import com.yandex.mapkit.map.CompositeIcon;
import com.yandex.mapkit.map.IconStyle;
import com.yandex.mapkit.map.MapObject;
import com.yandex.mapkit.map.MapObjectTapListener;
import com.yandex.mapkit.map.PlacemarkMapObject;
import com.yandex.mapkit.map.RotationType;
import com.yandex.mapkit.map.TextStyle;
import com.yandex.mapkit.mapview.MapView;
import com.yandex.mapkit.user_location.UserLocationLayer;
import com.yandex.mapkit.user_location.UserLocationObjectListener;
import com.yandex.mapkit.user_location.UserLocationView;
import com.yandex.runtime.image.ImageProvider;
import java.util.ArrayList;
import java.util.Objects;

public class MainActivity extends AppCompatActivity implements
UserLocationObjectListener {

    private final String MAPKIT_API_KEY = "39892a6b-25f4-4149-914e-
beaaae290a92";

    private FirebaseAuth mAuth;
    FirebaseUser user;
    FirebaseDatabase database;
    DatabaseReference myRef;
    PlacemarkMapObject meetpoint;
    MapObjectTapListener meetPointLst;
    Context cnt;

    private MapView mapView;
    private UserLocationLayer userLocationLayer;
    boolean flPos;
    double Latitude;
    double Longitude;

```

```

GroupT myGroup;
People me;
TextStyle styleT;
ArrayList<Members> members;
@Override
protected void onCreate(Bundle savedInstanceState) {
    firebaseIniz();
    getPermissions();
    members = new ArrayList<>();
    styleT = new TextStyle();
    styleT.setSize(8);
    styleT.setOffsetFromIcon(true);
    styleT.setPlacement(TextStyle.Placement.BOTTOM);
    cnt = getApplicationContext();
    MapKitFactory.setApiKey(MAPKIT_API_KEY);
    MapKitFactory.initialize(MainActivity.this);
    super.onCreate(savedInstanceState);
    setContentView(R.layout.user_location);
    flPos = true;
    mapView = findViewById(R.id.mapview);
    mapView.getMap().setRotateGesturesEnabled(false);
    mapView.getMapWindow().getMap().move(new CameraPosition(new
Point(0, 0), 2, 0, 0));
    MapKit mapKit = MapKitFactory.getInstance();
    userLocationLayer =
mapKit.createUserLocationLayer(mapView.getMapWindow());
    userLocationLayer.setVisible(true);
    userLocationLayer.setHeadingEnabled(false);
    userLocationLayer.setObjectListener(this);
    meetpoint =
mapView.getMapWindow().getMap().getMapObjects().addPlacemark(new
Point(0, 0), ImageProvider.fromResource(MainActivity.this,
R.drawable.ic_meetpoint));
    meetpoint.setVisible(false);

```

```

        mAuth.addAuthStateListener(new
FirebaseAuth.AuthStateListener() {
            @Override
            public void onAuthStateChanged(@NonNull FirebaseAuth
firebaseAuth) {
                user = mAuth.getCurrentUser();
                if (user != null) {
                    getUserInfo();
                    Button btn_nav = findViewById(R.id.btn2);
                    btn_nav.setEnabled(true);
                } else {
                    Button btn_nav = findViewById(R.id.btn2);
                    btn_nav.setEnabled(false);
                    myGroup = null;
                }
            }
        });

mapView.getMapWindow().getMap().getMapObjects().clear();
        meetpoint =
mapView.getMapWindow().getMap().getMapObjects().addPlacemark(new
Point(0, 0), ImageProvider.fromResource(MainActivity.this,
R.drawable.ic_meetpoint));
        meetpoint.setVisible(false);
        members.clear();
    }
}

Thread MyThread = new Thread(MyGetLocationThread);
MyThread.start();
Button btn_nav = findViewById(R.id.btn2);
btn_nav.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent i = new Intent(MainActivity.this,
MyGroupActivity.class);
        startActivity(i);
    }
}

```

```

    });
    Button btn_log = findViewById(R.id.btn);
    btn_log.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent i = new Intent(MainActivity.this,
ProfileActivity.class);
            startActivity(i);
        }
    });
}

Runnable MyGetLocationThread = new Runnable() {
    public void run() {
        Looper.prepare();
        LocationListener myLocationListener;
        LocationManager myLocationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
        myLocationListener = new LocationListener() {
            @RequiresApi(api = Build.VERSION_CODES.N)
            @Override
            public void onLocationChanged(@NonNull Location
location) {
                user = mAuth.getCurrentUser();
                if (user != null) {
                    Longitude = location.getLongitude();
                    Latitude = location.getLatitude();
                    Calendar c = Calendar.getInstance();
                    SimpleDateFormat sdf = new
SimpleDateFormat("dd,MMM,YYYY hh,mm,a");
                    String strDate = sdf.format(c.getTime());
                    DatabaseReference myRef =
database.getReference("People/" + user.getId() + "/Location/" +
strDate + "/Longitude");
                    myRef.setValue(Longitude);

```

```

        myRef = database.getReference("People/" +
user.getId() + "/Location/" + strDate + "/Latitude");
        myRef.setValue(Latitude);
    }
}
@Override
public void onStatusChanged(String provider, int
status, Bundle extras) {
    LocationListener.super.onStatusChanged(provider,
status, extras);
}
@Override
public void onProviderEnabled(@NonNull String
provider) {
    LocationListener.super.onProviderEnabled(provider);
}
@Override
public void onProviderDisabled(@NonNull String
provider) {
    LocationListener.super.onProviderDisabled(provider);
}
};
if (ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.ACCESS_FINE_LOCATION) == 0) {

myLocationManager.requestLocationUpdates(LocationManager.GPS_PROVIDE
R, 10000, 0, myLocationListener);
}
Looper.loop();
}
};
public void firebaseIniz() {
    FirebaseApp.initializeApp(getApplicationContext());
}

```

```

        mAuth = FirebaseAuth.getInstance();
        user = mAuth.getCurrentUser();
        database = FirebaseDatabase.getInstance();
        myRef = database.getReference();
        if ((user != null)) {
            getUserInfo();
        }
    }

    public void getPermissions() {
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED
            && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(MainActivity.this,
                new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                1);
            Log.d("Permission", "onCreate: Get permission");
        }
    }

    public void getUserInfo() {

FirebaseDatabase.getInstance().getReference().child("People").child(
user.getId())

                .addListenerForSingleValueEvent(new
ValueEventListener() {
                    @Override
                    public void onDataChange(DataSnapshot
dataSnapshot) {

                        String name =
dataSnapshot.child("name").getValue(String.class);
                        String eMail =
dataSnapshot.child("eMail").getValue(String.class);

```

					ПГУ 1.090501.08.001 ПЗ	Лист
						81
Изм.	Лист	№ докум.	Подпись	Дата		

```

        if (dataSnapshot.hasChild("group")) {
            String groupId =
dataSnapshot.child("group").getValue(String.class);
            getGroupInfo(groupId);
        }
        me = new People(name, eMail,
dataSnapshot.getKey());
    }
    @Override
    public void onCancelled(DatabaseError
databaseError) {
    }
});

FirebaseDatabase.getInstance().getReference().child("People").child(
user.getId()).addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(@NonNull DataSnapshot snapshot,
@Nullable String previousChildName) {
        if (Objects.equals("group", snapshot.getKey())) {
            String groupId =
snapshot.getValue(String.class);
            getGroupInfo(groupId);
        }
    }
    @Override
    public void onChildChanged(@NonNull DataSnapshot
snapshot, @Nullable String previousChildName) {
    }
    @Override
    public void onChildRemoved(@NonNull DataSnapshot
snapshot) {
        if (Objects.equals("group", snapshot.getKey())) {
            myGroup = null;

```

```

mapView.getMapWindow().getMap().getMapObjects().clear();
        members.clear();
    }
}
@Override
public void onChildMoved(@NonNull DataSnapshot snapshot,
@Nullable String previousChildName) {
    }
@Override
public void onCancelled(@NonNull DatabaseError error) {
    }
});
}

public void getGroupInfo(String id) {

FirebaseDatabase.getInstance().getReference().child("Groups").child(
id).addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot)
    {

        String name =
snapshot.child("name").getValue(String.class);
        String descr =
snapshot.child("description").getValue(String.class);
        String owner =
snapshot.child("owner").getValue(String.class);
        if (snapshot.hasChild("meetpoint")) {
            Point pnt = new
Point(snapshot.child("meetpoint").child("latitude").getValue(Double.
class),
snapshot.child("meetpoint").child("longitude").getValue(Double.class
));

            meetpoint.setGeometry(pnt);
            meetpoint.setVisible(true);

```

```

        meetpoint.addTapListener(meetPointLst = new
MapObjectTapListener() {
            @Override
            public boolean onMapObjectTap(@NonNull
MapObject mapObject, @NonNull Point point) {
                Intent rout = new
Intent(MainActivity.this, RouteToMeetpointActivity.class);
                rout.putExtra("long", Longitude);
                rout.putExtra("lat", Latitude);
                rout.putExtra("id", myGroup.id);
                startActivity(rout);
                return true;
            }
        });
    } else meetpoint.setVisible(false);
    myGroup = new GroupT(name, owner, id, descr);
    getMembersGroupInfo(id);
}
@Override
public void onCancelled(@NonNull DatabaseError error) {
}
});
}

public void getMembersGroupInfo(String id) {

FirebaseDatabase.getInstance().getReference().child("People").orderBy
Child("group").equalTo(id).addValueEventListener(new
ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot)
{
        if (myGroup != null) {
            for (DataSnapshot dsp : snapshot.getChildren())
{

```

```

        String name =
dsp.child("name").getValue(String.class);
        String eMail =
dsp.child("eMail").getValue(String.class);
        String id = dsp.getKey();
        People newMemb = new People(name, eMail,
id);

        boolean fl = true;
        if (members.size() != 1)
            for (int i = 0; i < members.size() - 1;
i++)

                if
(newMemb.id.equals(members.get(i).memb.id))

                    fl = false;
            if ((!(newMemb.id.equals(me.id))) && (fl)) {
                myGroup.addMembers(newMemb);
                Members nmb = new Members(newMemb);
                members.add(nmb);
            }
        }
    }

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

    });
}

@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 onObjectAdded(UserLocationView userLocationView) {
        userLocationView.getArrow().setIcon(ImageProvider.fromResource(MainActivity.this, R.drawable.ic_me, false));

        userLocationView.getPin().setIcon(ImageProvider.fromResource(
            MainActivity.this, R.drawable.user_arrow));
        CompositeIcon pinIcon =
        userLocationView.getPin().useCompositeIcon();
        pinIcon.setIcon(
            "icon",
            ImageProvider.fromResource(MainActivity.this,
R.drawable.user_arrow),
            new IconStyle().setAnchor(new PointF(0f, 0f))
                .setRotationType(RotationType.NO_ROTATION)
                .setZIndex(0f)
                .setScale(1f)
        );
        pinIcon.setIcon(
            "pin",
            ImageProvider.fromResource(MainActivity.this,
R.drawable.user_arrow),
            new IconStyle().setAnchor(new PointF(0f, 0f))
                .setRotationType(RotationType.NO_ROTATION)
                .setZIndex(0f)
                .setScale(1f)
        );

        userLocationView.getAccuracyCircle().setFillColor(R.color.purple_500
        );
    }

```

					ПГУ 1.090501.08.001 ПЗ	Лист
						86
Изм.	Лист	№ докум.	Подпись	Дата		

```

@Override
public void onObjectRemoved(UserLocationView view) {
}

@Override
public void onObjectUpdated(UserLocationView view, ObjectEvent
event) {
}

public class Members {
    PlacemarkMapObject loc;
    MapObjectTapListener lst;
    People memb;
    Members(People newMemb) {
        memb = newMemb;
        loc =
mapView.getMapWindow().getMap().getMapObjects().addPlacemark(new
Point(0, 0), ImageProvider.fromResource(MainActivity.this,
R.drawable.ic_other_people));
        loc.setText(newMemb.name, styleT);

FirebaseDatabase.getInstance().getReference().child("People").child(
memb.id).child("Location").limitToLast(1).addValueEventListener(new
ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot
snapshot) {
        for (DataSnapshot child :
snapshot.getChildren()) {
            String str =
(child.child("Latitude").getValue(Double.class).toString());
            Point pnt = new
Point(child.child("Latitude").getValue(Double.class),
child.child("Longitude").getValue(Double.class));
            loc.setGeometry(pnt);

```

```

        loc.addTapListener(lst = new
MapObjectTapListener() {
            @Override
            public boolean onMapObjectTap(@NonNull
MapObject mapObject, @NonNull Point point) {
                Intent rout = new
Intent(MainActivity.this, AboutMembActivity.class);
                rout.putExtra("id", memb.id);
                startActivity(rout);
                Toast.makeText(MainActivity.this,
"Выбран пользователь: " + newMemb.name, Toast.LENGTH_SHORT).show();
                return true;
            }
        });
    }
    @Override
    public void onCancelled(@NonNull DatabaseError
error) {
    }
});
}
}
}

```

ProfileActivity.java

```

package com.example.maptourist;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

```

```

import android.widget.ImageButton;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
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.FirebaseAuthException;
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 ProfileActivity extends AppCompatActivity {
    private FirebaseAuth mAuth;
    FirebaseUser user;
    FirebaseDatabase database;
    DatabaseReference myRef;
    Context cnt;
    EditText etName;
    People me;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        me = null;
        mAuth = FirebaseAuth.getInstance();
        database = FirebaseDatabase.getInstance();
        myRef = database.getReference();
        user = mAuth.getCurrentUser();
        if ((user != null)) {
            getUserInfo();
        }
        else logInActivity();
    }
}

```

```

    }

    public void profileActivity() {
        setContentView(R.layout.activity_profile);
        etName = findViewById(R.id.etNamePr);
        etName.setText(me.name);
        EditText etEmail = findViewById(R.id.etEmailPr);
        etEmail.setText(me.eMail);
        ImageButton btnEdit = findViewById(R.id.btn_edit);
        btnEdit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                showDialogEditName(ProfileActivity.this, "Изменение
имени", "Введите новое имя");
            }
        });
        Button btn_exit = findViewById(R.id.btn_exit);
        btn_exit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(ProfileActivity.this,
MainActivity.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
                startActivity(intent);
            }
        });
        Button btn_out = findViewById(R.id.btn_out_acc);
        btn_out.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                signOut();
                loginActivity();
            }
        });
    }
}

```

```

public void logInActivity() {
    setContentView(R.layout.activity_login);
    cnt = getApplicationContext();
    Button btn_exit = findViewById(R.id.btn_out_acc);
    btn_exit.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(cnt, MainActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
            startActivity(intent);
        }
    });
    Button btn_login = findViewById(R.id.btn_login_auth);
    btn_login.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String email = ((EditText)
findViewById(R.id.etEmail)).getText().toString();
            String password = ((EditText)
findViewById(R.id.etPassword)).getText().toString();
            signIn(email, password);
        }
    });
    Button btn_register = findViewById(R.id.btn_login_register);
    btn_register.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            registrationActivity();
        }
    });
}

public void registrationActivity() {
    setContentView(R.layout.activity_registration);
    cnt = getApplicationContext();

```

```

        Button btn_exit = findViewById(R.id.btn_registration_exit);
        btn_exit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                loginActivity();
            }
        });
        Button btn_reg = findViewById(R.id.btn_registration_reg);
        btn_reg.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String email = ((EditText)
findViewById(R.id.etEmail)).getText().toString();
                String password = ((EditText)
findViewById(R.id.etPassword)).getText().toString();
                String name = ((EditText)
findViewById(R.id.etName)).getText().toString();
                String confirmPassword = ((EditText)
findViewById(R.id.etConPassword)).getText().toString();
                if (email.equals(""))
                    Toast.makeText(cnt, "Введите email",
Toast.LENGTH_SHORT).show();
                else if (name.equals(""))
                    Toast.makeText(cnt, "Введите имя",
Toast.LENGTH_SHORT).show();
                else if (password.equals(""))
                    Toast.makeText(cnt, "Введите пароль",
Toast.LENGTH_SHORT).show();
                else if (!password.equals(confirmPassword))
                    Toast.makeText(cnt, "Пароли не совпадают",
Toast.LENGTH_SHORT).show();
                else
                    createAccount(email, password, name);
            }
        });
    });

```

```

    }

    private void signIn(String email, String password) {
        Log.d("sign", "signIn:" + email);
        mAuth.signInWithEmailAndPassword(email,
password).addOnCompleteListener(ProfileActivity.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();
                }
            }
        });
    }

    private void createAccount(String email, String password, String
name) {
        Log.d("create", "createAccount:" + email);
        mAuth.createUserWithEmailAndPassword(email,
password).addOnCompleteListener(ProfileActivity.this, task -> {
            if (task.isSuccessful()) {
                Log.d("create", "createUserWithEmail:success");
                user = mAuth.getCurrentUser();
                database = FirebaseDatabase.getInstance();
                DatabaseReference myRef =
database.getReference("People/" + user.getId() + "/name");
                myRef.setValue(name);
                myRef = database.getReference("People/" +
user.getId() + "/eMail");          //set path to get the multiple
dropdown nodes
                myRef.setValue(email);
                getUserInfo();
            }
        });
    }

```

```

        } else {
            Toast.makeText(cnt, task.getException().toString(),
Toast.LENGTH_SHORT).show();
            Log.d("create", "createUserWithEmail:abortively");
        }
    });
}

public void getUserInfo() {
    user = mAuth.getCurrentUser();

    FirebaseDatabase.getInstance().getReference().child("People").child(
user.getId())
        .addListenerForSingleValueEvent(new
ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot
dataSnapshot) {

                // Get user information
                String name =
dataSnapshot.child("name").getValue(String.class);
                String eMail =
dataSnapshot.child("eMail").getValue(String.class);
                me = new People(name, eMail,
dataSnapshot.getKey());

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

            }
        });
}

private void signOut() {
    mAuth.signOut();
    user = null;
}

```

```

        me = null;
    }

    public void showDialogEditName(Context c, String title, String
msg) {
        final EditText txtEdit = new EditText(c);
        txtEdit.setHint(msg);
        final String[] newName = new String[1];
        AlertDialog dialogEdit = new AlertDialog.Builder(c)
            .setTitle(title)
            .setView(txtEdit)
            .setPositiveButton("Сохранить", (dialog,
whichButton) -> {
                newName[0] = txtEdit.getText().toString();
                DatabaseReference myRef =
database.getReference("People/" + user.getUid() + "/name");
                myRef.setValue(newName[0]);
                etName.setText(newName[0]);
                dialog.cancel();
            })
            .setNegativeButton("Отменить", new
DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int
whichButton) {
                    dialog.cancel();
                }
            })
            .create();
        dialogEdit.show();
    }
}

```

MyGroupActivity.java

```

package com.example.maptourist;

import android.annotation.SuppressLint;
import android.app.AlertDialog;

```

					<p style="text-align: center; font-size: 1.2em;"><i>ПГУ 1.090501.08.001 ПЗ</i></p>	Лист
						95
Изм.	Лист	№ докум.	Подпись	Дата		

```

import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.drawable.Drawable;
import android.os.Build;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toolbar;

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

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 com.yandex.mapkit.ScreenPoint;
import com.yandex.mapkit.geometry.Point;
import com.yandex.mapkit.map.CameraListener;
import com.yandex.mapkit.map.CameraPosition;
import com.yandex.mapkit.map.CameraUpdateReason;
import com.yandex.mapkit.map.Map;
import com.yandex.mapkit.map.PlacemarkMapObject;
import com.yandex.mapkit.mapview.MapView;
import com.yandex.runtime.image.ImageProvider;

```

```

import java.util.ArrayList;

@SuppressLint("NewApi")
public class MyGroupActivity extends AppCompatActivity {

    ListAdapterGroups adapterGr;
    ListAdapterPeople adapterP;
    private FirebaseAuth mAuth;
    FirebaseUser user;
    FirebaseDatabase database;
    DatabaseReference myRef;
    private MapView mapView;
    MenuItem meetI;

    final int DELETE_GROUP = 1;
    final int OUT_GROUP = 2;
    final int DELETE_MEETPOINT = 3;

    boolean hasMeet;

    PlacemarkMapObject meet;

    Context cnt;

    ArrayList<GroupT> groups;
    GroupT myGroup;
    People me;

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

        cnt = getApplicationContext();
        myGroup = null;
        mAuth = FirebaseAuth.getInstance();

```

```

        user = mAuth.getCurrentUser();
        database = FirebaseDatabase.getInstance();
        myRef = database.getReference();
        hasMeet = false;
        if ((user != null)) {
            getUserInfo();
        }
    }

    public void transActivity() {
        setContentView(R.layout.activity_trans);
        Button btn_exit = findViewById(R.id.btn_nav_exit);
        btn_exit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(cnt, MainActivity.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
                startActivity(intent);
            }
        });
        Button btn_add_group =
findViewById(R.id.btn_nav_create_group);
        btn_add_group.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View view) {
                addGroupActivity();
            }
        });
        Button btn_join_group =
findViewById(R.id.btn_nav_join_group);
        btn_join_group.setOnClickListener(new View.OnClickListener()
{
            @Override

```

```

        public void onClick(View view) {
            joinGroupActivity();
        }
    });
}

@RequiresApi(api = Build.VERSION_CODES.M)
public void groupActivity() {
    setContentView(R.layout.activity_group);
    TextView tvName = findViewById(R.id.tv_name_gr);
    tvName.setText(myGroup.name);
    TextView tvDesc = findViewById(R.id.tv_descr_gr);
    tvDesc.setText(myGroup.description);
    adapterP = new ListAdapterPeople(cnt, myGroup.members);
    ListView listView = findViewById(R.id.listView);
    listView.setAdapter(adapterP);
    if (equalsID(me.id, myGroup.owner)) {
        Toolbar toolbar = findViewById(R.id.toolbar3);
        toolbar.inflateMenu(R.menu.group_menu);
        meetI = toolbar.getMenu().getItem(2);
        if (hasMeet)
            meetI.setTitle("Удалить точку сбора");
        else meetI.setTitle("Создать точку сбора");
        toolbar.setOnMenuItemClickListener(new
Toolbar.OnMenuItemClickListener() {
            @Override
            public boolean onMenuItemClick(MenuItem menuItem) {
                switch (menuItem.getItemId()) {
                    case R.id.group_menu_delete: {
                        showDialogDelete(MyGroupActivity.this,
"Вы уверены, что хотите удалить группу?", "Удалить", DELETE_GROUP);
                        break;
                    }
                    case R.id.group_menu_edit: {
                        editGroupActivity();
                    }
                }
            }
        });
    }
}

```

```

        break;
    }
    case R.id.group_menu_set_meet: {
        if (hasMeet) {

showDialogDelete(MyGroupActivity.this, "Вы уверены, что хотите
удалить точку сбора?", "Удалить", DELETE_MEETPOINT);

        } else setMeet();
        break;
    }
}
return true;
}

});

Drawable d = getDrawable(R.drawable.ic_btn_menu);
toolbar.setOverflowIcon(d);
}

Context cnt = getApplicationContext();
Button btn_exit = findViewById(R.id.btn_nav_exit);
btn_exit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(cnt, MainActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
        startActivity(intent);
    }
});

Button btn_out_group = findViewById(R.id.btn_nav_out_group);
if (equalsID(me.id, myGroup.owner))
btn_out_group.setEnabled(false);
else btn_out_group.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View view) {

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.08.001 ПЗ

Лист

100

```

        showDialogDelete(MyGroupActivity.this, "Вы уверены,
что хотите выйти из группы?", "Выйти", OUT_GROUP);
    }
});
}

private boolean equalsID(String id, String owner) {
    for (int i=0; i<id.length(); i++)
    {
        char a = id.charAt(i);
        char b = owner.charAt(i);
        if (a!=b) return false;
    }

    return true;
}

private void setMeet() {
    setContentView(R.layout.activity_set_meetpoint);
    mapView = (MapView) findViewById(R.id.mapview);

    mapView.getMapWindow().getMap().setRotateGesturesEnabled(false);
    Float centerX = mapView.getMapWindow().width() / 2f;
    Float centerY = mapView.getMapWindow().height() / 2f;
    ScreenPoint centerPoint = new ScreenPoint(centerX, centerY);
    Point worldPoint =
mapView.getMapWindow().screenToWorld(centerPoint);
    meet =
mapView.getMapWindow().getMap().getMapObjects().addPlacemark(worldPo
int, ImageProvider.fromResource(MyGroupActivity.this,
R.drawable.ic_create_meetpoint));

    mapView.getMapWindow().getMap().addCameraListener(new
CameraListener() {
        @Override

```

```

        public void onCameraPositionChanged(@NonNull Map map,
@NonNull CameraPosition cameraPosition, @NonNull CameraUpdateReason
cameraUpdateReason, boolean b) {
            meet.setGeometry(cameraPosition.getTarget());
        }
    });
    Button btn_exit = findViewById(R.id.btn_cancel_meet);
    btn_exit.setOnClickListener(new View.OnClickListener() {
        @SuppressWarnings("NewApi")
        @Override
        public void onClick(View view) {
            groupActivity();
        }
    });
    Button btn_save = findViewById(R.id.btn_add_meet);
    btn_save.setOnClickListener(new View.OnClickListener() {
        @SuppressWarnings("NewApi")
        @Override
        public void onClick(View view) {
            Point worldPoint =
mapView.getMapWindow().screenToWorld(centerPoint);
            DatabaseReference myRef =
database.getReference("Groups/" + myGroup.id +
"/meetpoint/latitude");          //set path to get the multiple
dropdown nodes
            myRef.setValue(worldPoint.getLatitude());
            myRef = database.getReference("Groups/" + myGroup.id
+ "/meetpoint/longitude");          //set path to get the multiple
dropdown nodes
            myRef.setValue(worldPoint.getLongitude());
            hasMeet = true;
            meetI.setTitle("Удалить точку сбора");
            groupActivity();
        }
    });
}

```

```

}

public void addGroupActivity() {
    setContentView(R.layout.activity_create_group);
    Button btn_add = findViewById(R.id.btn_create_group_ok);
    btn_add.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String name = ((EditText)
findViewById(R.id.et_name_group)).getText().toString();
            String description = ((EditText)
findViewById(R.id.etm_about_group)).getText().toString();
            String owner = user.getId();
            DatabaseReference myRef =
database.getReference("Groups/");          //set path to get the
multiple dropdown nodes
            String key = myRef.push().getKey();
            myRef = database.getReference("Groups/" + key +
"/name");
            myRef.setValue(name);
            myRef = database.getReference("Groups/" + key +
"/description");          //set path to get the multiple dropdown nodes
            myRef.setValue(description);
            myRef = database.getReference("Groups/" + key +
"/owner");          //set path to get the multiple dropdown nodes
            myRef.setValue(owner);
            myRef = database.getReference("People/" +
user.getId() + "/group");
            myRef.setValue(key);
            myGroup = new GroupT(name, owner, key, description);
            groupActivity();
        }
    });
    Button btn_cancel =
findViewById(R.id.btn_create_group_cancel);

```

```

        btn_cancel.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(cnt, MainActivity.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
                startActivity(intent);
            }
        });
    }

    public void editGroupActivity() {
        setContentView(R.layout.activity_edit_group);
        EditText etName = findViewById(R.id.et_name_group);
        etName.setText(myGroup.name);
        EditText etDesc = findViewById(R.id.etm_about_group);
        etDesc.setText(myGroup.description);
        Button btn_add = findViewById(R.id.btn_edit_group_ok);
        btn_add.setOnClickListener(new View.OnClickListener() {
            @SuppressWarnings("NewApi")
            @Override
            public void onClick(View view) {
                String name = ((EditText)
findViewById(R.id.et_name_group)).getText().toString();
                String description = ((EditText)
findViewById(R.id.etm_about_group)).getText().toString();
                myRef = database.getReference("Groups/" + myGroup.id
+ "/name");
                myRef.setValue(name);
                myRef = database.getReference("Groups/" + myGroup.id
+ "/description"); //set path to get the multiple dropdown
nodes
                myRef.setValue(description);
                myGroup.name = name;
                myGroup.description = description;
            }
        });
    }
}

```

```

        groupActivity();
    }
});
Button btn_cancel =
findViewById(R.id.btn_edit_group_cancel);
btn_cancel.setOnClickListener(new View.OnClickListener() {
    @SuppressWarnings("NewApi")
    @Override
    public void onClick(View view) {
        groupActivity();
    }
});
}

public void joinGroupActivity() {
    setContentView(R.layout.activity_join_group);
    groups = new ArrayList<>();
    adapterGr = new ListAdapterGroups(cnt, groups);
    ListView listView = findViewById(R.id.listView);
    listView.setAdapter(adapterGr);
    listView.setOnItemClickListener((parent, itemClicked,
position, id) -> {
        GroupT group = groups.get(position);
        String idGroup = group.id;
        DatabaseReference myRef =
database.getReference("People/" + user.getUid() + "/group");
//set path to get the multiple dropdown nodes
        myRef.setValue(idGroup);
        getGroupInfo(idGroup);
    });
    findGroups();
    Button btn_exit = findViewById(R.id.btn_edit_group_cancel);
    btn_exit.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {

```

```

        Intent intent = new Intent(cnt, MainActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
        startActivity(intent);
    }
    });
}

public void findGroups() {

FirebaseDatabase.getInstance().getReference().child("Groups").addLis
tenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot)
    {
        for (DataSnapshot dsp : snapshot.getChildren()) {
            String id = dsp.getKey();
            String name =
dsp.child("name").getValue(String.class);
            String descr =
dsp.child("description").getValue(String.class);
            String owner =
dsp.child("owner").getValue(String.class);
            GroupT newGroup = new GroupT(name, owner, id,
descr);

            setList(newGroup);
        }
    }

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

    }

    });
}

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.08.001 ПЗ

Лист

106

```

public void setList(GroupT gr) {
    adapterGr.addGroup(gr);
    adapterGr.notifyDataSetChanged();
}

public void getUserInfo() {

FirebaseDatabase.getInstance().getReference().child("People").child(
user.getUid())

    .addListenerForSingleValueEvent(new
ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot
dataSnapshot) {

            // Get user information
            String name =
dataSnapshot.child("name").getValue(String.class);
            String eMail =
dataSnapshot.child("eMail").getValue(String.class);
            me = new People(name, eMail,
dataSnapshot.getKey());

            if (dataSnapshot.hasChild("group")) {
                String groupId =
dataSnapshot.child("group").getValue(String.class);
                getGroupInfo(groupId);
            } else transActivity();
        }

        @Override
        public void onCancelled(DatabaseError
databaseError) {

        }

    });
}

```

```

    public void getGroupInfo(String id) {

FirebaseDatabase.getInstance().getReference().child("Groups").child(
id).addListenerForSingleValueEvent(new ValueEventListener() {
    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot)
{
        String name =
snapshot.child("name").getValue(String.class);
        String descr =
snapshot.child("description").getValue(String.class);
        String owner =
snapshot.child("owner").getValue(String.class);
        if (snapshot.hasChild("meetpoint")) {
            hasMeet = true;
        } else hasMeet = false;
        myGroup = new GroupT(name, owner, id, descr);
        findMembers(id);
        groupActivity();
    }

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

    }

});
}

    private void findMembers(String id) {

FirebaseDatabase.getInstance().getReference().child("People").orderBy
yChild("group").equalTo(id).addValueEventListener(new
ValueEventListener() {

```

```

        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot)
        {
            for (DataSnapshot dsp : snapshot.getChildren()) {
                String name =
dsp.child("name").getValue(String.class);
                String eMail =
dsp.child("eMail").getValue(String.class);
                String id = dsp.getKey();
                People newMemb = new People(name, eMail, id);

                boolean fl = true;
                if (myGroup.members.size() != 0)
                    for (int i = 0; i < myGroup.members.size();
i++)

                        if
(newMemb.id.equals(myGroup.members.get(i).id))
                            fl = false;
                            if ((!(newMemb.id.equals(me.id))) && (fl)) {
                                adapterP.addPeople(newMemb);
                                adapterP.notifyDataSetChanged();
                            }
                        }
                    }

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

        }

    });
}

    public void showDialogDelete(Context c, String title, String
msgBtn, int type) {
        AlertDialog dialogEdit = new AlertDialog.Builder(c)

```

```

        .setTitle(title)
        .setPositiveButton(msgBtn, (dialog, whichButton) ->
{
    switch (type) {
        case OUT_GROUP: {
            DatabaseReference myRef =
database.getReference("People/" + user.getUid() + "/group");
            myRef.removeValue();
            transActivity();
            break;
        }
        case DELETE_GROUP: {
            deleteGroup(myGroup.id);
            break;
        }
        case DELETE_MEETPOINT: {
            DatabaseReference myRef =
database.getReference("Groups/" + myGroup.id + "/meetpoint");
            hasMeet = false;
            meetI.setTitle("Создать точку сбора");
            myRef.removeValue();
            break;
        }
    }
    dialog.cancel();
})
        .setNegativeButton("Отменить", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int
whichButton) {
                dialog.cancel();
            }
        })
        .create();
dialogEdit.show();

```

```

    }

    public void deleteGroup(String id){
        DatabaseReference myRef = database.getReference("Groups/" +
id);

        myRef.removeValue();
        transActivity();

FirebaseDatabase.getInstance().getReference().child("People").orderBy
yChild("group").equalTo(id).addListenerForSingleValueEvent(new
ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot)
{
        for (DataSnapshot dsp : snapshot.getChildren()) {
            DatabaseReference myRef =
database.getReference("People/" + dsp.getKey() + "/group");
            myRef.removeValue();
        }
    }
    @Override
    public void onCancelled(@NonNull DatabaseError error) {
    }
    });
}
}

```

AboutMembActivity.java

```

package com.example.maptourist;

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

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

```

					ПГУ 1.090501.08.001 ПЗ	Лист
						111
Изм.	Лист	№ докум.	Подпись	Дата		

```

import android.view.View;
import android.widget.Button;
import android.widget.TextView;

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.MapKit;
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.DrivingRouter;
import com.yandex.mapkit.directions.driving.DrivingRouterType;
import com.yandex.mapkit.directions.driving.DrivingSession;
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.mapview.MapView;
import com.yandex.mapkit.transport.TransportFactory;
import
com.yandex.mapkit.transport.masstransit.MasstransitRouteSerializer;
import com.yandex.mapkit.transport.masstransit.Session;
import com.yandex.mapkit.transport.masstransit.PedestrianRouter;
import com.yandex.mapkit.transport.masstransit.Route;
import com.yandex.mapkit.transport.masstransit.TimeOptions;
import
com.yandex.mapkit.transport.masstransit.internal.PedestrianRouterBin
ding;
import com.yandex.runtime.Error;

```

```

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

public class AboutMembActivity extends AppCompatActivity implements
com.yandex.mapkit.transport.masstransit.Session.RouteListener {

    private MapView mapView;
    private MapObjectCollection mapOb = null;
    private PedestrianRouter pedestrianRouter = null;
    private Session session = null;

    String id;
    ArrayList<Point> rout;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_about_memb);
        mapView = findViewById(R.id.mapview);
        mapView.getMap().setRotateGesturesEnabled(false);
        MapKit mapKit = MapKitFactory.getInstance();
        Intent intent = getIntent();
        id = intent.getStringExtra("id");
        getRoutInfo(id);
        Button exit = findViewById(R.id.btn_rout_exit);
        exit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(AboutMembActivity.this,
MainActivity.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
                startActivity(intent);
            }
        });
    }
}

```

```

    }

    public void getRoutInfo(String id) {
        rout = new ArrayList<>();

        FirebaseDatabase.getInstance().getReference().child("People").child(
            id).child("name").addValueEventListener(new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot
                    snapshot) {

                    TextView txtv =
                        findViewById(R.id.tv_name_memb);

                    txtv.setText(snapshot.getValue(String.class));
                }

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

                }

            });

        FirebaseDatabase.getInstance().getReference().child("People").child(
            id).child("Location")
            .addListenerForSingleValueEvent(new
                ValueEventListener() {
                    @Override
                    public void onDataChange(DataSnapshot
                        dataSnapshot) {

                        for (DataSnapshot child :
                            dataSnapshot.getChildren()) {

                            Point pnt = new
                                Point(child.child("Latitude").getValue(Double.class),
                                    child.child("Longitude").getValue(Double.class));

```

```

        rout.add(pnt);
    }
    showRout();
}
@Override
public void onCancelled(DatabaseError
databaseError) {
    }
});
}
public void showRout(){
    pedestrianRouter =
TransportFactory.getInstance().createPedestrianRouter();
    mapOb = mapView.getMap().getMapObjects().addCollection();
    submitRequest();
}
private void submitRequest() {
    ArrayList<RequestPoint> requestPoints = new ArrayList<>();
    TimeOptions timeOpt = new TimeOptions();
    requestPoints.add(new RequestPoint(rout.get(0),
RequestPointType.WAYPOINT,null,null));
    for (int i=1; i<rout.size()-1; i=i+(rout.size())/10)
    {
        requestPoints.add(new RequestPoint(rout.get(i),
RequestPointType.VIAPOINT,null,null));
    }
    requestPoints.add(new RequestPoint(rout.get(rout.size()-1),
RequestPointType.WAYPOINT,null,null));
    session = pedestrianRouter.requestRoutes(requestPoints,
timeOpt, this);
}
@Override
public void onMasstransitRoutes(@NonNull List<Route> list) {
    if (list != null && !list.isEmpty()) {
        Route route = list.get(0);

```

```

        mapOb.addPolyline(route.getGeometry());
        mapView.getMapWindow().setFocusRect(new ScreenRect(new
ScreenPoint(30f, 0f), new
ScreenPoint(mapView.getMapWindow().width()-30f,
mapView.getMapWindow().height())));
        CameraPosition cmp =
mapView.getMap().cameraPosition(Geometry.fromPolyline(route.getGeome
try()), 0, 0, null);
        mapView.getMap().move(cmp);
    }
}
@Override
public void onMasstransitRoutesError(@NonNull Error error) {
}
}

```

RouteToMeetpointActivity.java

```

package com.example.maptourist;

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

import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

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.RequestPoint;
import com.yandex.mapkit.RequestPointType;
import com.yandex.mapkit.ScreenPoint;
import com.yandex.mapkit.ScreenRect;

```

```

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.DrivingSession;
import com.yandex.mapkit.directions.driving.VehicleOptions;
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.map.PolylineMapObject;
import com.yandex.mapkit.mapview.MapView;
import com.yandex.mapkit.transport.TransportFactory;
import com.yandex.mapkit.transport.masstransit.PedestrianRouter;
import com.yandex.mapkit.transport.masstransit.Route;
import com.yandex.mapkit.transport.masstransit.Session;
import com.yandex.mapkit.transport.masstransit.TimeOptions;
import com.yandex.runtime.Error;
import com.yandex.runtime.image.ImageProvider;

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

public class RouteToMeetpointActivity extends AppCompatActivity
implements
com.yandex.mapkit.transport.masstransit.Session.RouteListener {

    private MapView mapView;
    private MapObjectCollection mapOb = null;
    private PedestrianRouter pedestrianRouter = null;
    private Session session = null;
    private DrivingRouter drivingRouter = null;
    private DrivingSession drivingSession = null;
    String id;
    double Latitude;

```

```

double Longitude;

PlacemarkMapObject loc;
PlacemarkMapObject meet;

Point rout;
Point myPosition;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_route_to_meetpoint);
    mapView = findViewById(R.id.mapview);
    mapView.getMap().setRotateGesturesEnabled(false);
    Intent intent = getIntent();
    id = intent.getStringExtra("id");
    Latitude = intent.getDoubleExtra("lat", 0);
    Longitude = intent.getDoubleExtra("long", 0);
    myPosition = new Point(Latitude, Longitude);
    getRoutInfo(id);
    Button exit = findViewById(R.id.btn_rout_exit);
    exit.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new
Intent(RouteToMeetpointActivity.this, MainActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
            startActivity(intent);
        }
    });
}

public void getRoutInfo(String id) {

```

```

FirebaseDatabase.getInstance().getReference().child("Groups").child(
id).child("meetpoint").addListenerForSingleValueEvent(new
ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot)
{
        DataSnapshot child = snapshot.child("latitude");
        Double l = child.getValue(Double.class);
        rout = new
Point(snapshot.child("latitude").getValue(Double.class),
snapshot.child("longitude").getValue(Double.class));
        showRout();
    }
    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }
});
}

public void showRout(){
    pedestrianRouter =
TransportFactory.getInstance().createPedestrianRouter();
    mapOb = mapView.getMap().getMapObjects().addCollection();
    submitRequest();
}
private void submitRequest() {
    ArrayList<RequestPoint> requestPoints = new ArrayList<>();
    TimeOptions timeOpt = new TimeOptions();
    requestPoints.add(new RequestPoint(myPosition,
RequestPointType.WAYPOINT,null,null));
    requestPoints.add(new RequestPoint(rout,
RequestPointType.WAYPOINT,null,null));
}

```

```

        loc =
mapView.getMapWindow().getMap().getMapObjects().addPlacemark(myPosit
ion, ImageProvider.fromResource(RouteToMeetpointActivity.this,
R.drawable.ic_other_people));

        meet =
mapView.getMapWindow().getMap().getMapObjects().addPlacemark(rout,
ImageProvider.fromResource(RouteToMeetpointActivity.this,
R.drawable.ic_meetpoint));

        session = pedestrianRouter.requestRoutes(requestPoints,
timeOpt, this);
    }

    @Override
    public void onMasstransitRoutes(@NonNull List<Route> list) {
        if (list != null && !list.isEmpty()) {
            Route route = list.get(0);
            PolylineMapObject polyline =
mapOb.addPolyline(route.getGeometry());
            polyline.setStrokeColor(Color.MAGENTA);
            mapView.getMapWindow().setFocusRect(new ScreenRect(new
ScreenPoint(100f, 100f), new
ScreenPoint(mapView.getMapWindow().width() - 100f,
mapView.getMapWindow().height() - 100f)));
            CameraPosition cmp =
mapView.getMap().cameraPosition(Geometry.fromPolyline(route.getGeome
try()), 0, 0, null);
            mapView.getMap().move(cmp);
        }
    }

    @Override
    public void onMasstransitRoutesError(@NonNull Error error) {
        submitDriveRequest();
    }

```

```

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

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

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

    drivingSession = drivingRouter.requestRoutes(requestPoints,
drivingOptions, vehicleOptions, new
DrivingSession.DrivingRouteListener() {
        @Override
        public void onDrivingRoutes(@NonNull List<DrivingRoute>
routes) {
            if (routes != null && !routes.isEmpty()) {
                DrivingRoute route = routes.get(0);
                PolylineMapObject polyline =
mapOb.addPolyline(route.getGeometry());
                polyline.setStrokeColor(Color.LTGRAY);
                mapView.getMapWindow().setFocusRect(new
ScreenRect(new ScreenPoint(100f, 100f), new
ScreenPoint(mapView.getMapWindow().width() - 100f,
mapView.getMapWindow().height() - 100f)));
                CameraPosition cmp =
mapView.getMap().cameraPosition(Geometry.fromPolyline(route.getGeome
try()), 0, 0, null);
                mapView.getMap().move(cmp);
            }
        }
        @Override
        public void onDrivingRoutesError(@NonNull Error error) {
        }
    }
}

```

```

        });
    }
}

GroupT.java
package com.example.maptourist;

import java.util.ArrayList;

public class GroupT {
    ArrayList<People> members;
    String owner;
    String name;
    String id;
    String description;

    GroupT(String newName, String newOwner, String newId, String
newDescription){
        name = newName;
        owner = newOwner;
        id = newId;
        description = newDescription;
        members = new ArrayList<>();
    }
    public void addMembers(People memb){
        boolean fl = true;
        for (int i=0; i< members.size(); i++)
            if (members.get(i).id.equals(memb.id))
            {
                members.get(i).name = memb.name;
                members.get(i).eMail = memb.eMail;
                fl = false;
            }
        if (fl) members.add(memb);
    }
    public void renameMember(String id, String newName){

```

```

        for (int i=0; i< members.size(); i++)
        {
            if (members.get(i).id==id)
                members.get(i).editName(newName);
        }
    }

    public void deleteMember(String id){
        for (int i=0; i< members.size(); i++)
        {
            if (members.get(i).id==id)
                members.remove(i);
        }
    }
}

```

People.java

```

package com.example.maptourist;
import android.location.Location;
import java.util.ArrayList;
public class People {
    String name;
    String eMail;
    String id;
    People(){
        name = "";
        eMail = "";
        id = "";
    }
    People(String newName, String newEMail, String newId){
        name = newName;
        eMail = newEMail;
        id = newId;
    }
    void editName(String newName){
        name = newName;
    }
}

```

```

        String getName(){
            return name;
        }
    }

ListAdapterGroups.java
package com.example.maptourist;
import android.annotation.SuppressLint;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;
import java.util.ArrayList;
public class ListAdapterGroups extends BaseAdapter {
    private LayoutInflater LInflater;
    private ArrayList<GroupT> list;
    ListAdapterGroups(Context context, ArrayList<GroupT> data) {
        list = data;
        LInflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    }
    @Override
    public int getCount() {
        return list.size();
    }
    @Override
    public GroupT getItem(int i) {
        return list.get(i);
    }
    @Override
    public long getItemId(int i) {
        return i;
    }
    public void addGroup(GroupT gr){

```

```

        list.add(gr);
    }
    @SuppressWarnings({"MissingPermission", "NewApi"})
    @Override
    public View getView(int position, View convertView, ViewGroup
parent) {
        ViewHolder holder;
        View v = convertView;
        if (v == null) {
            holder = new ViewHolder();
            v = LayoutInflater.inflate(R.layout.group_list, parent,
false);

            holder.name = v.findViewById(R.id.tv_name);
            holder.descr = v.findViewById(R.id.tv_descr);
            v.setTag(holder);
        }
        holder = (ViewHolder) v.getTag();
        GroupT gr = getItem(position);
        holder.name.setText(gr.name);
        holder.descr.setText(gr.description);
        return v;
    }
    private static class ViewHolder{
        private TextView name;
        private TextView descr;
    }
}

```

ListAdapterPeople.java

```

package com.example.maptourist;
import android.annotation.SuppressLint;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;

```

```

import android.widget.TextView;
import com.example.maptourist.GroupT;
import com.example.maptourist.R;
import java.util.ArrayList;

public class ListAdapterPeople extends BaseAdapter {
    private LayoutInflater LInflater;
    private ArrayList<People> list;
    ListAdapterPeople(Context context, ArrayList<People> data) {
        list = data;
        LInflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    }
    @Override
    public int getCount() {
        return list.size();
    }
    @Override
    public People getItem(int i) {
        return list.get(i);
    }
    @Override
    public long getItemId(int i) {
        return i;
    }
    public void addPeople(People gr){
        list.add(gr);
    }
    @SuppressWarnings({"MissingPermission", "NewApi"})
    @Override
    public View getView(int position, View convertView, ViewGroup
parent) {
        ViewHolder holder;
        View v = convertView;
        if (v == null) {
            holder = new ViewHolder();

```

```

        v = LInflater.inflate(R.layout.group_list, parent,
false);

        holder.name = v.findViewById(R.id.tv_name);
        holder.eMail= v.findViewById(R.id.tv_descr);
        v.setTag(holder);
    }
    holder = (ViewHolder) v.getTag();
    People ppl = getItem(position);
    holder.name.setText(ppl.name);
    holder.eMail.setText(ppl.eMail);
    return v;
}
private static class ViewHolder{
    private TextView name;
    private TextView eMail;
}
}

```

Приложение В

(обязательное)

Программа для трекинга распределенных групп людей

Презентация

					<i>ПГУ 1.090501.08.001 ПЗ</i>	Лист
						128
Изм.	Лист	№ докум.	Подпись	Дата		

ПРОГРАММА ДЛЯ ТРЕКИНГА РАСПРЕДЕЛЁННОЙ ГРУППЫ ЛЮДЕЙ

Автор работы: студент гр. 19В031 Пименова Алиса Андреевна
Научный руководитель: к.т.н., доцент каф. ВТ Федюнин Роман Николаевич

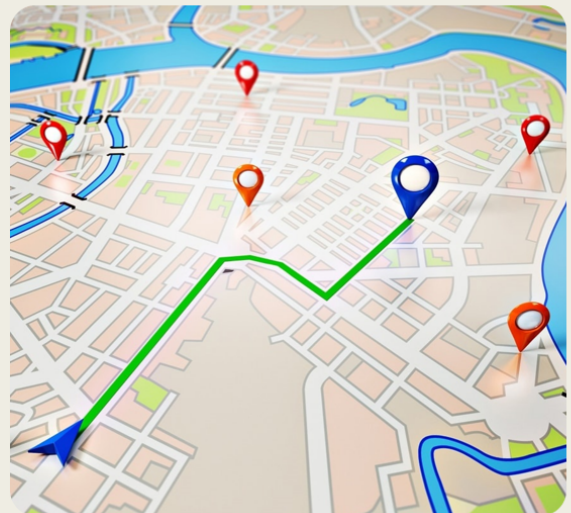
Пенза, 2024

Актуальность

Мониторинг — это специальная форма наблюдения за текущим изменением тех или иных процессов или объектов в пространстве и во времени, осуществляемая на постоянной основе. Для наблюдения за изменениями местоположения используется системы спутникового слежения.

Имеет широкое применение, например в:

- логистических компаниях
- таксопарках
- сельскохозяйственных компаниях
- авиации и мореходстве
- туризме



Цель работы и объекты

Цель работы - разработка программы для трекинга распределенных групп людей.

Объектом работы является программа для трекинга распределенной группы людей, выполняющая следующие функции:

- регистрация и авторизация пользователей;
- создание, редактирование и удаление групп;
- отображение местоположения членов группы на общей карте;
- отображение истории перемещений членов группы;
- установка и редактирование точки сбора, а также построение маршрута до нее от каждого участника группы.

Сравнение аналогов



iSharing

- Отсутствие возможности создать группу
- Отсутствие точки сбора
- Отсутствие интуитивно понятного интерфейса



Connected

- Отсутствие точки сбора
- Отсутствие интуитивно понятного интерфейса



Blink

- Отсутствие возможности создать группу
- Отсутствие точки сбора
- Отсутствие возможности просмотра истории перемещений

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.08.001 ПЗ

Лист

130

Используемые технологии

База данных и аутентификация



Firestore

Разработка мобильного приложения для ОС Android



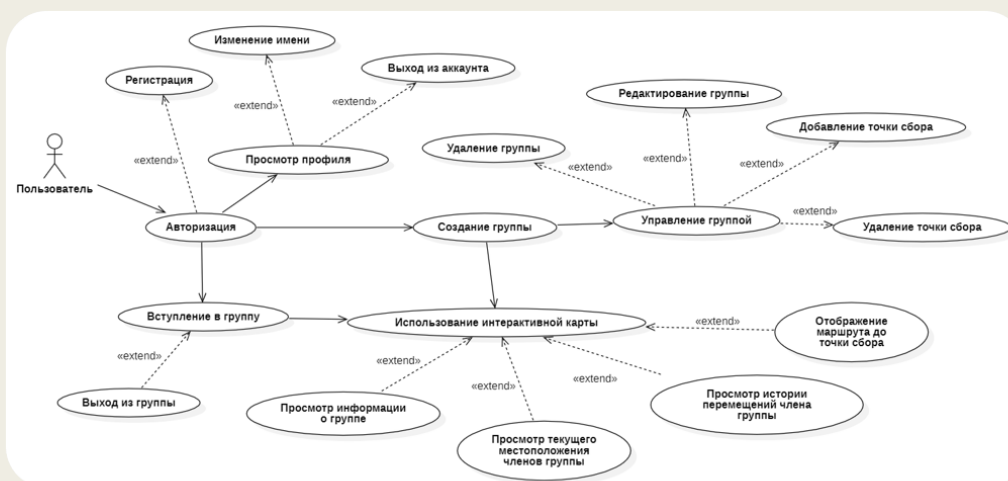
Android Studio

Работа с картами



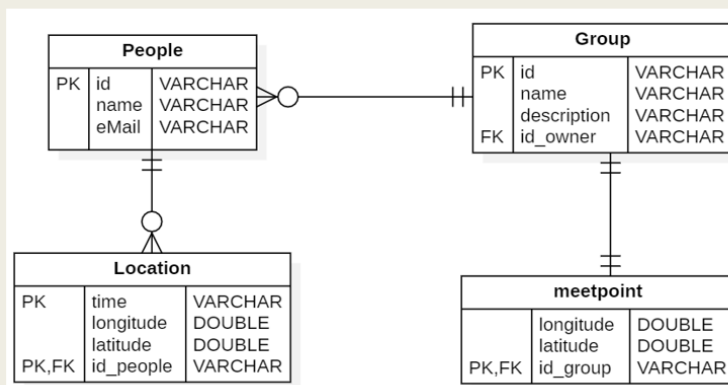
MapKit

Функции приложения

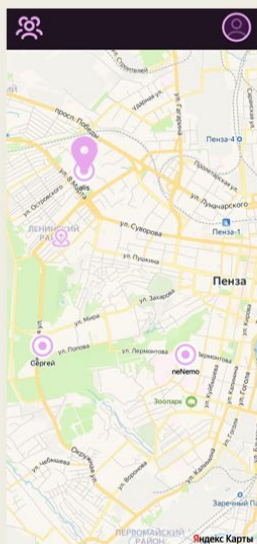


Сущности разрабатываемой базы данных

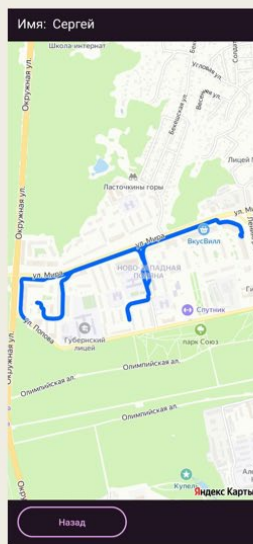
- Таблица People для хранения информации и пользователя
- Таблица Group для хранения информации о группах
- Таблица Location для хранения данных о перемещениях
- Таблица meetpoint для хранения информации о точках сбора



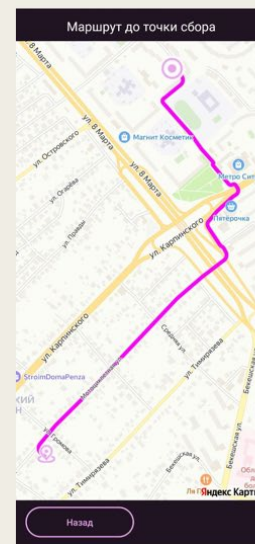
Интерфейс приложения



Главное окно



История перемещений

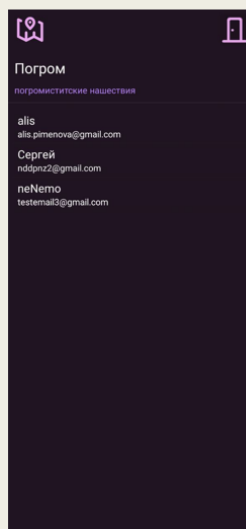


Маршрут до точки сбора

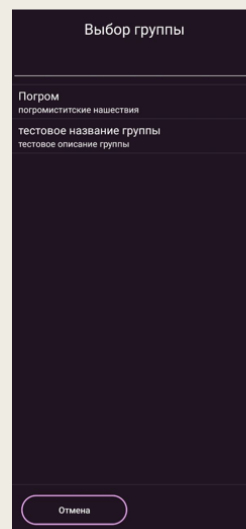
Интерфейс приложения



Профиль



Группа



Выбор группы

Спасибо за внимание!