Федеральное государственное автономное
Образовательное учреждение
Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Вычислительная техника

УТВЕРЖДАЮ
Заведующий кафедрой ВТ
_____ О.В. Непомнящий
подпись инициалы, фамилия
«_____» _____ 2021 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника

Клиент-серверная система для работы с маршрутами на карте

Руководитель	_____ подпись, дата	ст. преподаватель	В.С. Васильев
Выпускник	_____ подпись, дата		А.Е. Зозулин
Нормконтролер	_____ подпись, дата	ст. преподаватель	В.С. Васильев

Красноярск 2021

ОГЛАВЛЕНИЕ

Введение ... 3
1 Спецификация требований ... 4
1.1 Общее описание продукта ... 4
1.2 Анализ существующих аналогов ... 4
1.2.1 AllTrails ... 4
1.2.2 Komoot ... 5
1.2.3 Hiking Project ... 6
1.3 Операционная среда ... 7
1.4 Функциональные требования ... 7
1.4.1 Функциональные требования клиентского приложения ... 7
1.4.2 Функциональные требования к серверу ... 14
1.5 Выводы по главе ... 19
2 Проектирование ... 20
2.1 Общая архитектура системы ... 20
2.2 Динамическая модель системы ... 21
2.3 Базы данных ... 24
2.4 Выводы по главе ... 26
3 Разработка и тестирование ... 27
3.1 Выбор инструментов разработки ... 27
3.2 Реализация элементов слоя View ... 27
3.3 Реализация элементов слоя ViewModel ... 28
3.4 Реализация элементов слоя Model ... 29
3.5 Инструкция к сборке проекта ... 29
3.6 Тестирование ... 31
3.7 Выводы по главе ... 31
Заключение ... 32
Список сокращений ... 33
Список использованных источников ... 34

ВВЕДЕНИЕ

В наше время информационных технологий карты местности все больше представлены не на бумажных носителях, а на цифровых. Кроме того, из-за повсеместного распространения смартфонов каждый человек может иметь в своем кармане карту всего мира. А поскольку интерес людей к исследованию мира не угасает, сейчас набирают популярность приложения для мобильных устройств, которые помогают человеку работать с картами и ориентироваться на местности.

Целью работы является разработка клиент-серверной системы для работы с картами, где клиент – это приложение для мобильных устройств, работающих под управлением операционной системы Android.

Для достижения цели были поставлены следующие задачи:

– выполнить анализ существующих аналогов;

– составить спецификацию требований;

– выполнить проектирование и разработку системы;

– разработать инструкцию для разработчика.

1 Спецификация требований

1.1 Общее описание продукта

Продукт представляет собой клиент-серверную систему, где клиент – это приложение для мобильных устройств, работающих под управлением ОС Android, которое предоставляет возможности для создания маршрутов, а также навигации на местности с использованием уже готовых маршрутов, дополнить список которых может любой пользователь приложения.

Пользовательская информация и информация о маршрутах хранится на сервере.

1.2 Анализ существующих аналогов

Для обеспечения конкурентоспособности разрабатываемого приложения необходим анализ уже существующих приложений со схожим функционалом. Анализ аналогов позволит выявить недостатки анализируемых приложений, что может помочь при проектировании разрабатываемой системы, а также он даст возможность выявить удачные решения, которые впоследствии можно будет реализовать в проекте.

Поскольку приложение разрабатывается для мобильных устройств, работающих под управлением ОС Android, поиск приложений-аналогов проводился среди приложений, опубликованных в Google Play [1]. В результате было найдено 3 приложения с похожим функционалом: AllTrails [2], Komoot [3] и Hiking Project [4].

1.2.1 AllTrails

Данное приложение, помимо функций записи маршрута и фильтрации опубликованных маршрутов, выделяется тем, что:

– позволяет планировать маршруты, т.е. сохранять избранные маршруты в отдельный список;

– дает пользователю доступ к общей статистике (общее пройденное расстояние и т.д.) и статистике каждого маршрута в частности;

– поиск нужного маршрута можно выполнять прямо на карте (на карте местности есть метки, отождествленные с началом маршрута).

Основной недостаток данного приложения заключается в том, что некоторые ключевые функции недоступны в бесплатной версии приложения. Так, например, невозможно сохранение карт локально на устройстве, что делает невозможной навигацию по маршрутам без подключения к интернету.

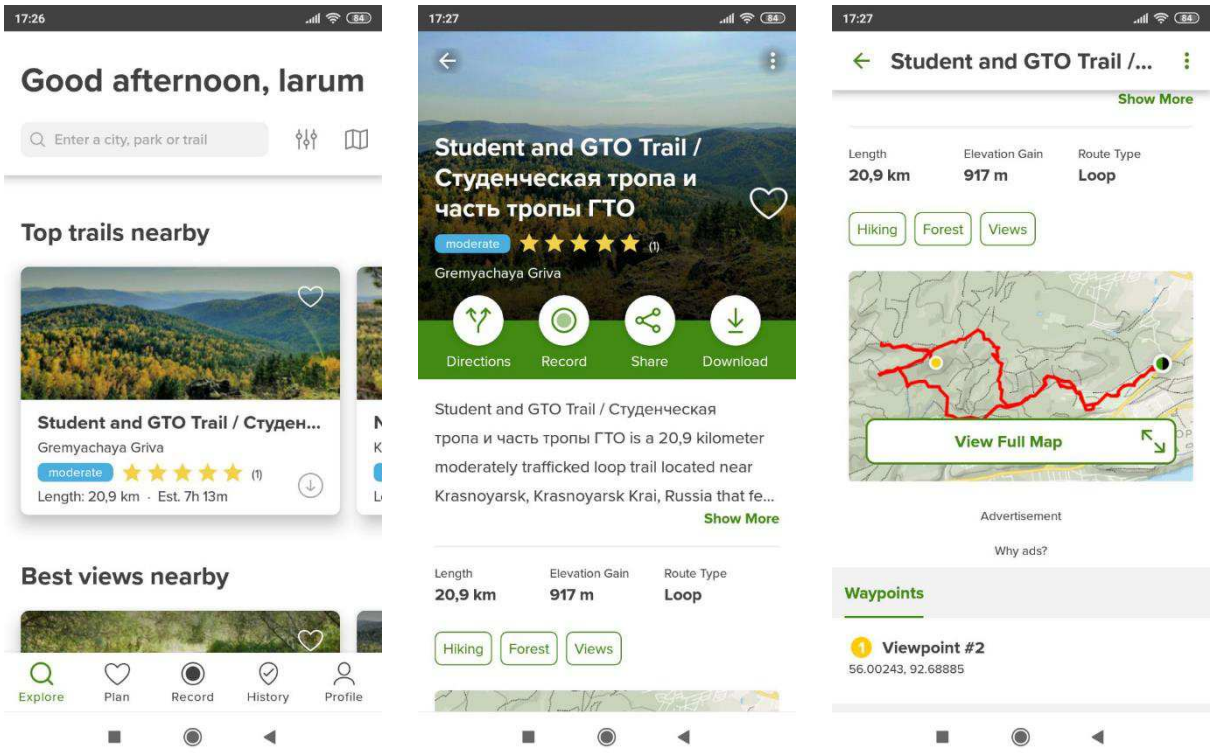

Рисунок 1 – Интерфейс приложения AllTrails

1.2.2 Komoot

Komoot – это приложение для планирования пеших и велосипедных маршрутов.

Данное приложение имеет следующие особенности:

– каждый маршрут хранит множество дополнительной информации (типы поверхностей, высотных профиль);

– возможность с помощью стороннего приложения (Google Maps) построить путь до точки начала маршрута.

Основной недостаток данного приложения заключается в том, что, как и у предыдущего, без платной подписки невозможна навигация при отсутствии подключения к интернету.

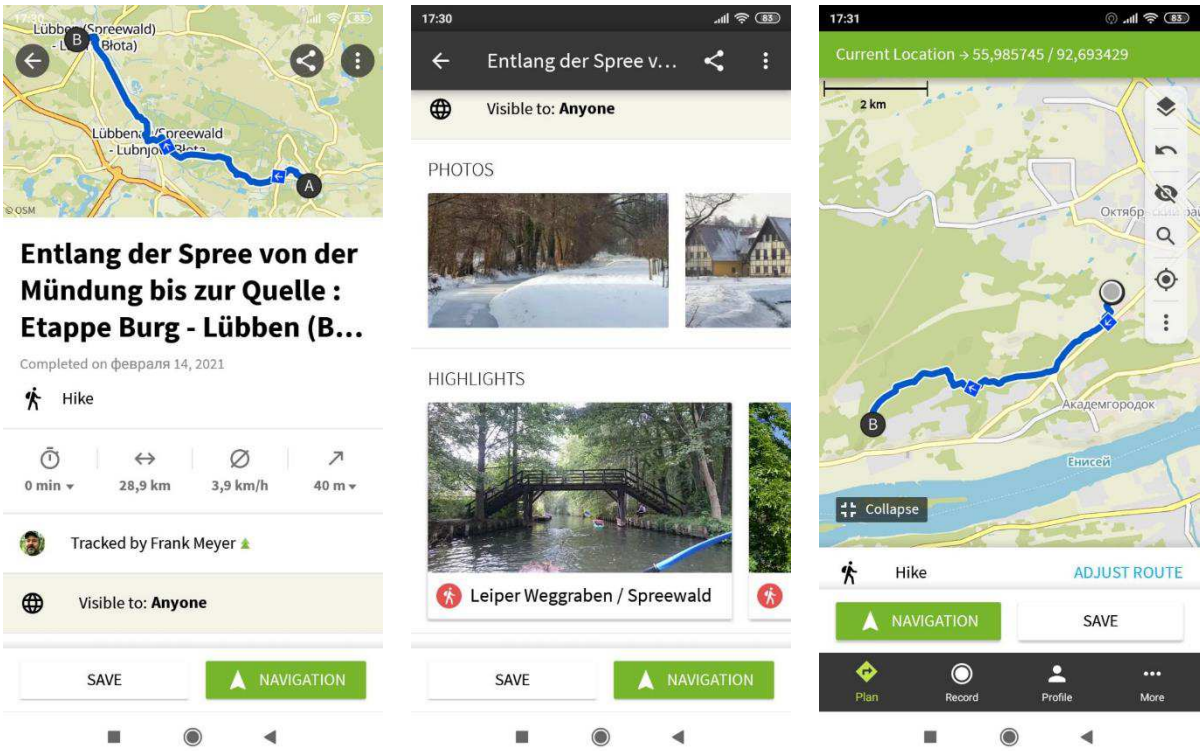

Рисунок 2 – Интерфейс приложения Komoot

1.2.3 Hiking Project

Это приложение для навигации по туристическим маршрутам.

Данное приложение имеет следующие особенности:

– есть возможность сохранять выбранные маршруты локально на устройстве;

– есть возможность поиска маршрутов на карте.

У приложения есть следующие недостатки:

– наличие неочевидных элементов интерфейса;

– сохранять на устройство можно только группы маршрутов (все маршруты группируются по признаку принадлежности к административному субъекту);

– при отсутствии интернета пользователь может увидеть сохраненный маршрут, но не саму карту.

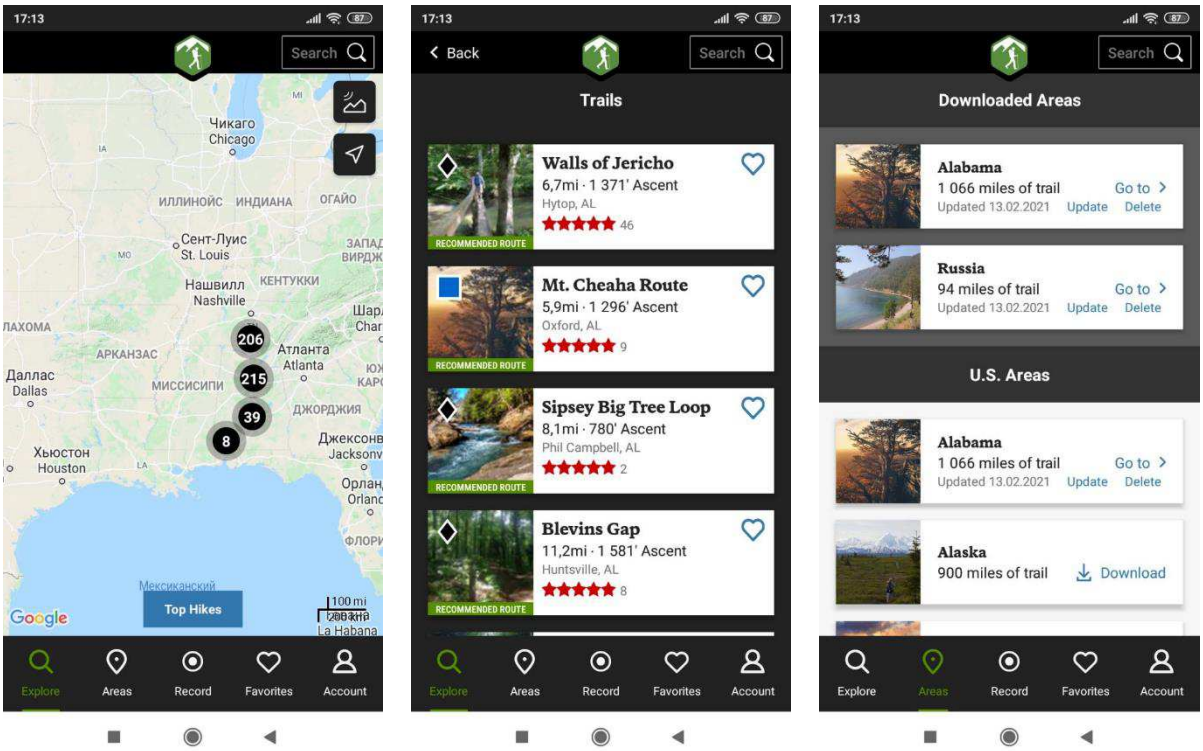

Рисунок 3 – Интерфейс приложения Hiking Project

1.3 Операционная среда

Клиентская часть разрабатывается для мобильных устройств, работающих под управлением ОС Android версии API 29 и выше.

Серверная часть разрабатывается для ОС Linux.

Также для корректной работы продукта устройство должно иметь встроенный модель GPS.

1.4 Функциональные требования

1.4.1 Функциональные требования клиентского приложения

На рисунке 4 изображен макет интерфейса клиентского приложения.

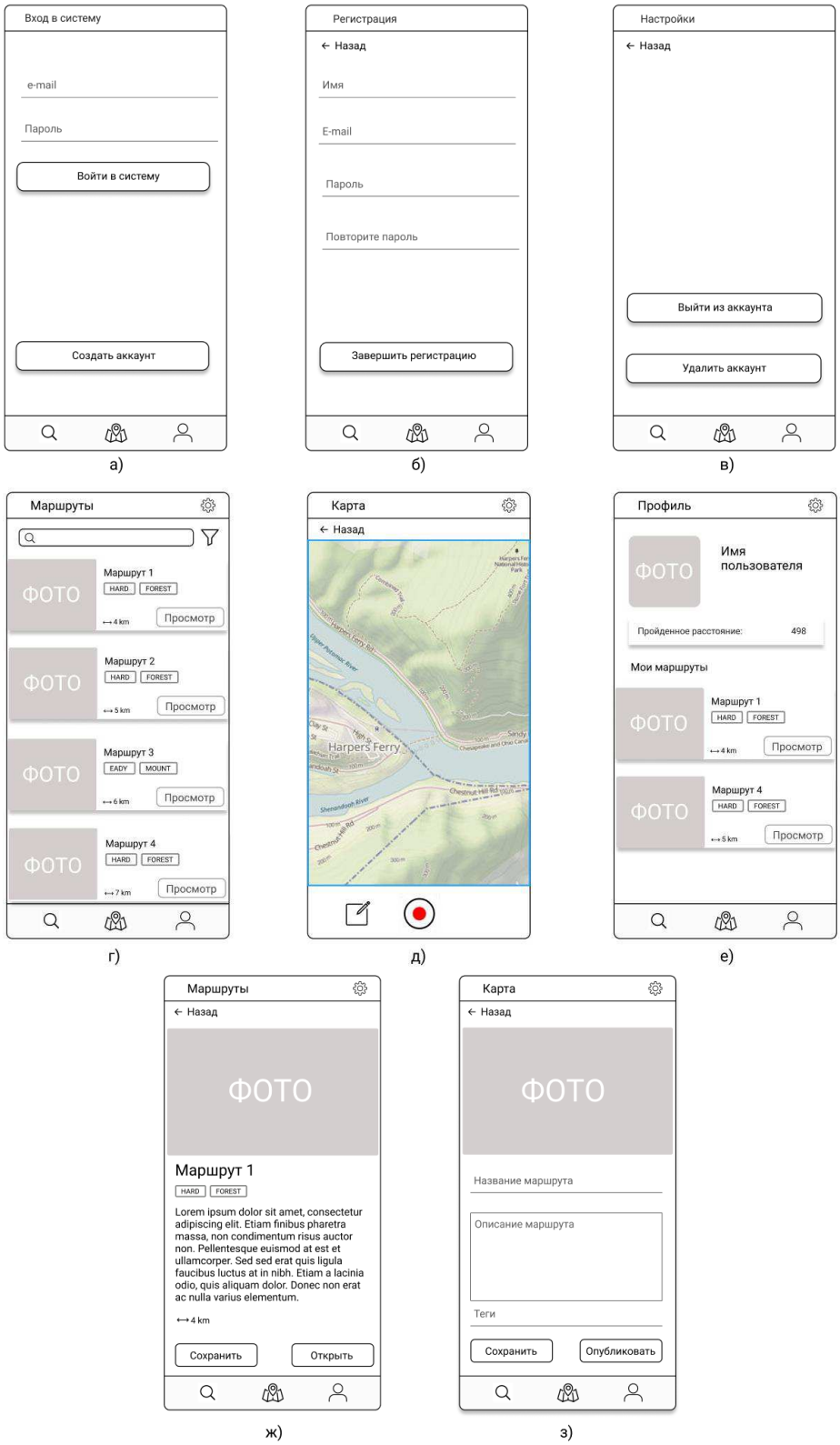

Рисунок 4 – Макет интерфейса приложения: а) экран «Вход в систему» б) экран «Регистрации» в) экран «Параметры» г) экран «Список маршрутов» д) экран «Карта» е) экран «Профиль» ж) экран «Описание маршрута з) экран «Завершение создания маршрута».

Взаимодействие между экранами представлено на диаграмме потоков экранов (Рисунок 5).

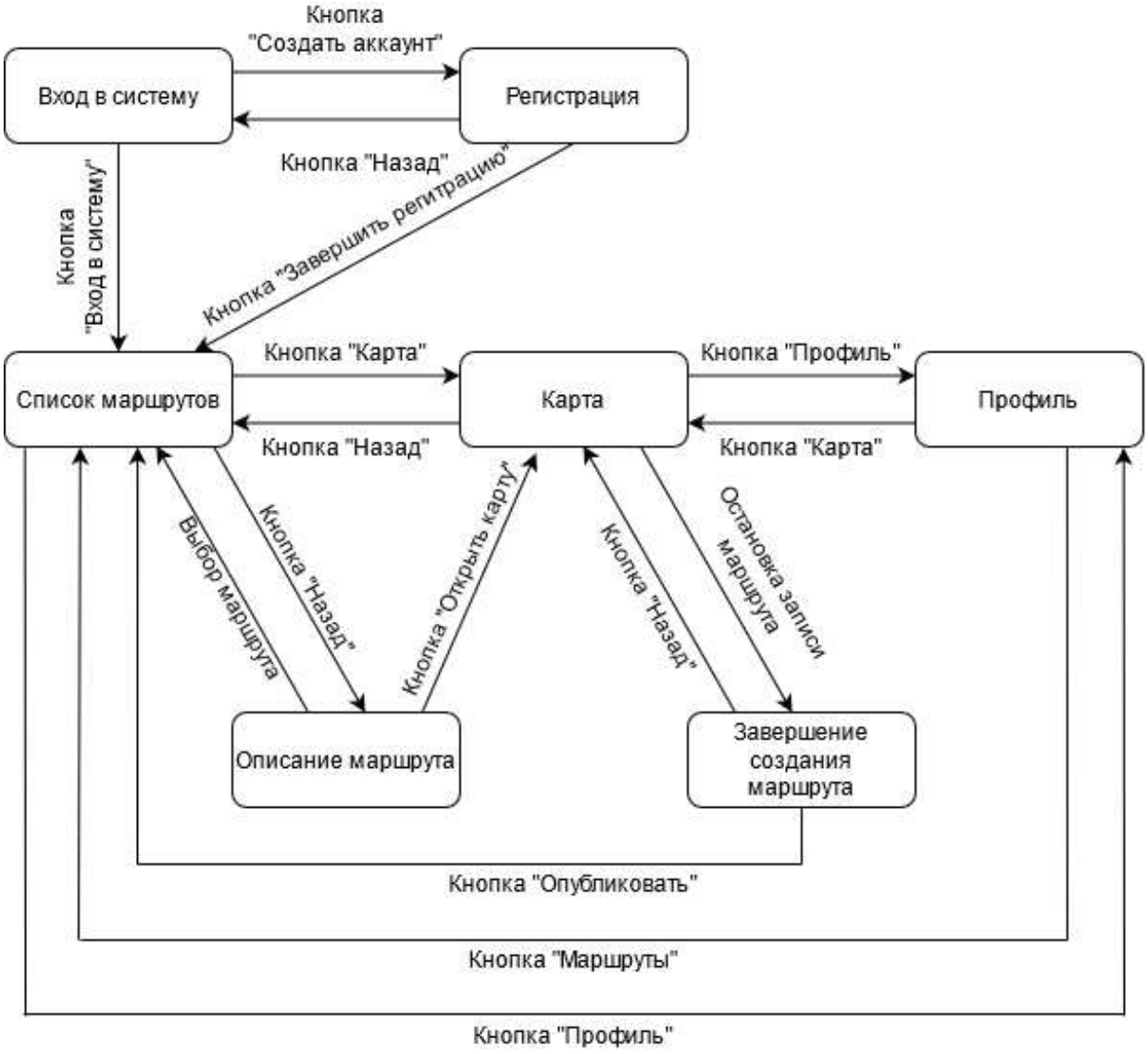

Рисунок 5 – Диаграмма потока экранов

Функциональные требования к программе выражены с помощью диаграммы прецедентов [6] (Рисунок 6).

Рисунок 6 – Диаграмма прецедентов

Текстовое описание прецедентов

Название прецедента: Вход в систему.

Цель сценария: осуществить вход пользователя в систему или зарегистрировать пользователя в системе.

Предусловия: пользователь впервые запустил приложение на устройстве либо вышел из аккаунта и находится на экране «Вход в систему».

Основной сценарий:

A. Уже зарегистрированный в системе пользователь вводит корректный логин/email и пароль, после чего нажимает кнопку «Войти в систему».

B. Незарегистрированный в системе пользователь нажимает кнопку «Создать аккаунт», после чего попадает на экран «Регистрация». После

заполнения требуемых полей данных пользователь нажимает кнопку «Завершить регистрацию».

С. Пользователь выбирает один из пунктов в поле «Авторизация с помощью».

Постусловия: пользователь попадает на экран «Список маршрутов».

Условия ввода в действие альтернативных сценариев:

Условие 1. Пользователь вводит неверные логин/email и/или пароль при попытке входа в систему.

Приложение выдает ошибку о некорректности введенных данных.

Условие 2. Пользователь вводит некорректные данные в полях ввода на экране «Регистрация».

Приложение выдает ошибку о некорректности введенных данных.

Название прецедента: Поиск маршрута в списке.

Цель сценария: нахождение пользователем подходящего маршрута.

Предусловия: пользователь находится на экране «Список маршрутов»

Основной сценарий:

Пользователь выбирает необходимые ему фильтры маршрутов, после чего выбирает подходящий маршрут в списке.

Постусловия: пользователь переходит на экран «Описание маршрута».

Условия ввода в действие альтернативных сценариев

Условие 1. В базе нет маршрутов, подходящих под критерии пользователя.

Отображается сообщение об отсутствии подходящих под критерии пользователя маршрутах.

Название прецедента: Оценка маршрута.

Цель сценария: дать пользовательскую оценку маршруту.

Предусловия: пользователь находится на экране «Описание маршрута».

Основной сценарий:

Пользователь нажимает на кнопку «Оценить», после чего в появившемся окне выбирает цифру от 1 до 10.

Постусловия: сохранены данные о пользовательской оценке маршрута.

Название прецедента: Создание маршрута.

Цель сценария: Создать пользовательский маршрут.

Предусловия: Пользователь находится на экране «Карта», на устройстве запущен модуль GPS.

Основной сценарий:

Пользователь нажимает кнопку начала записи маршрута, после чего приложение начинает запрашивать у модуля GPS данные о текущем местоположении устройства (долгота, широта) и записывать их в локальную базу данных устройства. Чтобы завершить маршрут, пользователь вновь нажимает кнопку записи маршрута, после чего попадает в окно «Завершение создания маршрута».

Постусловия: Маршрут создан.

Условия ввода в действие альтернативных сценариев

Условие 1. **Во время записи маршрута пропал сигнал GPS.**

A. Если экран устройства включен, генерируется сообщение «Слабый сигнал GPS».

B. Приложение прекращает запись данных о текущем местоположении устройства. Запись возобновляется при появлении GPS сигнала.

Название прецедента: Публикация маршрута.

Цель сценария: добавить пользовательский маршрут с общедоступный список маршрутов.

Предусловия: пользователь находится на экране «Завершение создания маршрута».

Основной сценарий:

Пользователь заполняет необходимые текстовые поля, по желанию добавляет фото, после чего нажимает кнопку «Опубликовать», после чего созданный пользователем маршрут добавляется в список общедоступных маршрутов и удаляется из локальной базы данных.

Постусловия:

Маршрут опубликован.

Условия ввода в действие альтернативных сценариев:

Условие 1. Пользователь не желает публиковать маршрут.

Пользователь нажимает кнопку «Сохранить».

Постусловия: Маршрут не опубликован, но при этом и не удален из локальной базы данных устройства.

Название прецедента: Выход из системы

Цель сценария: выйти из пользовательского аккаунта.

Предусловия: пользователь находится на экране «Параметры»

Основной сценарий:

Пользователь нажимает на кнопку «Выйти из аккаунта».

Постусловия: пользователь попадает на экран «Аутентификация».

Название прецедента: Удаление аккаунта

Цель сценария: удалить пользовательский аккаунт.

Предусловия: пользователь находится на экране «Параметры»

Основной сценарий:

Пользователь нажимает на кнопку «Удалить аккаунта»,

Постусловия: пользовательский аккаунт удалён, пользователь попадает на экран «Аутентификация».

Название прецедента: Просмотр профиля.

Цель сценария: просмотреть профиль пользователя.

Предусловия: пользователь находится на экране «Профиль».

Основной сценарий:

A. Пользователь ознакомляется со статистикой профиля.

B. Пользователь выбирает маршрут из списка пройденных.

Постусловия:

A. Пользователь ознакомлен со статистикой.

B. Пользователь попадает на экран «Описание маршрута».

Название прецедента: Отображение маршрута на карте

Цель сценария: Открыть выбранный маршрут на карте

Предусловия: Пользователь находится на экране «Описание маршрута»

Основной сценарий:

Пользователь нажимает на кнопку «Открыть на карте», после чего происходит автоматический переход на экран «Карта», где отображается выбранный маршрут.

Постусловия:

Выбранный маршрут отображен на карте.

1.4.2 Функциональные требования к серверу

Ниже описано API сервера:

Метод: getRouteCoords

Вид: result getRouteCoords (int routeId)

Описание: возвращает координаты выбранного маршрута.

Список входных параметров:

Таблица 1 – Список входных параметров метода getRouteCoords

Параметр	Тип	Описание
routeId	int	Идентификатор маршрута из списка

Возвращает:

JSON [7] [файл следующего формата:

```
[
        {"lat": "123.1214", "lng": "123.1214” "},
        {"lat": "123.1213", "lng": "123.1215"},
        {"lat": "123.1212", "lng": "123.1216"},
        {"lat": "123.1211", "lng": "123.1217"},
        {"lat": "123.1210", "lng": "123.1218"}
]
```

Метод: getRoutesList

Вид: result getRoutesList(string[] filters, int amount)

Описание: Возвращает список маршрутов, отфильтрованный согласно переданным параметрам.

Список входных параметров:

Таблица 2 – Список входных параметров метода getRoutesList

Параметр	Тип	Описание
filters	string[]	Список характеристик, по которым будут отфильтрованы маршруты в базе данных.
amount	int	Количество отфильтрованных маршрутов, которое будет передано на клиент.

Возвращает:

JSON файл следующего формата:

```
[
        {
                "id": "3183264663",
                "name": "Pathway1",
                "length": "21.3",
                "rating": "8.2",
                "tags": [
```

```
                "easy", "long", "forest"
            ]
            },

        {
            "id": "3183264773",
            "name": "Pathway2",
            "length": "22.3",
            "rating": "9.2",
            "tags": [
                "hard"
            ]
        }
]
```


Метод: publicRoute

Вид: int publicRoute(json data)

Описание:

Принимает JSON файл, содержащий информацию о маршруте, сохраняет её в базе данных на сервере. Если все прошло корректно, возвращает id маршрута, в противном случае возвращает код ошибки.

Список входных параметров:

В качестве входного параметра отправляется JSON файл следующего формата:


```
{
    "id": "3183264663",
    "name": "Pathway",
    "length": "21.3",
    "rating": "8.2",
    "description": "Route's description ... ",
    "creator_id": "1"
    "tags": [
        "easy", "long", "forest"
    ],
    "img": "encoded image",
    "route" : {
        "routeCoords":  [
            {"lat": "123.1214", "lng": "123.1214" "},
            {"lat": "123.1213", "lng": "123.1215"},
            {"lat": "123.1212", "lng": "123.1216"},
            {"lat": "123.1211", "lng": "123.1217"},
            {"lat": "123.1210", "lng": "123.1218"}
        ],
        "routeLocations": [
{
```

```
                        "coords": {
                                "lat": "123.1234",
                                "lng":"234.1234",
                        },
                        "name": "location name",
                        "description": "location description",
                        "id": "1",
"img": "encoded image"
                        }
                ]
        }
}
```


Возвращает:

Таблица 3 – Список возвращаемых параметров метода publicRoute

Тип	Описание
int	Возвращаемое значение. Возможные значения: id маршрута – если маршрут корректно загружен -1 – если при загрузке возникла ошибка

Метод: registerUser

Вид: int registerUser(string name, string email, string password)

Описание:

Проверяет, есть ли в системе пользователи с такими же именем/электронной почтой. Если таковых нет, регистрирует нового пользователя в системе и возвращает id нового пользователя. В противном случае возвращает код ошибки.

Входные параметры:

Таблица 4 – Список входных параметров метода registerUser

Параметр	Тип	Описание
name	string	Имя пользователя
email	string	Электронная почта пользователя
password	string	Пароль пользователя

Возвращает:

JSON файл следующего формата:

{

```
        "id": "1",
        "name": "username"
}
```

Метод: authorizeUser

Вид: int authorizeUser(string email, string password)

Описание:

Проверяет, есть ли в системе пользователь с переданным email. Если такой пользователь есть, проверяет соответствие пароля. Если пароль корректен, возвращает id пользователя. В противном случае возвращает код ошибки.

Входные параметры:

Таблица 5 – Список входных параметров метода authorizeUser

Параметр	Тип	Описание
email	string	Электронная почта пользователя
password	string	Пароль пользователя

Возвращает:

JSON файл следующего формата:

```
{
        "id": "1",
        "name": "username",
        "email": "user email"
}
```

Метод: rateRoute

Вид**:** int rateRoute(float rate)

Описание:

Принимает на вход данную пользователем оценку маршруту, после чего высчитывает среднеарифметическое всех данных маршруту оценок. Полученное значение возвращает.

Входные параметры:

Таблица 6 – Входные параметры метода rateRoute

Параметр	Тип	Описание
rate	float	Пользовательская оценка маршруту

Возвращает:

Таблица 7 – Выходные парамерты метода rateRoute

Тип	Описание
float	Среднеарифметическое всех данных маршруту оценок

1.5 Выводы по главе

В результате анализа приложений–аналогов были найдены удачные решения, которые можно реализовать в разрабатываемом приложении:

– использование элемента BottomNavigationView для навигации между экранами приложения;

– возможность сохранения маршрутов локально на устройстве, для использования их при отсутствии подключения к интернету;

– сортировка маршрутов по тегам.

С учетом этих решений были сформулированы требования к разрабатываемой системе.

2 Проектирование

Прецеденты описывают поведение пользователя в системе. Для визуализации взаимодействия компонентов системы между собой используется диаграмма последовательностей, которая отражает динамическую модель системы.

2.1 Общая архитектура системы

Архитектура клиентского приложения спроектирована с использованием архитектурного шаблона MVVM [8], который обширно используется при разработке приложений для ОС Android.

Клиентское приложение использует базу данных для хранения общей информации о созданных пользователем маршрутах, а также о маршрутах из списка опубликованных другими пользователями, но сохраненных локально на устройство. Сами маршруты (их координаты), а также информация о локациях на маршруте, хранятся в локальной файловой системе в виде файлов формата JSON.

Сервер использует базу данных для хранения общей информации о маршрутах, а также для хранения данных о зарегистрированных в системе пользователях. Координаты маршрутов, локации, а также изображения хранятся в файловой системе устройства.

Обмен данными между клиентским приложением и сервером осуществляется с помощью протокола HTTP.

На рисунке 7 представлена архитектура разрабатываемой системы.

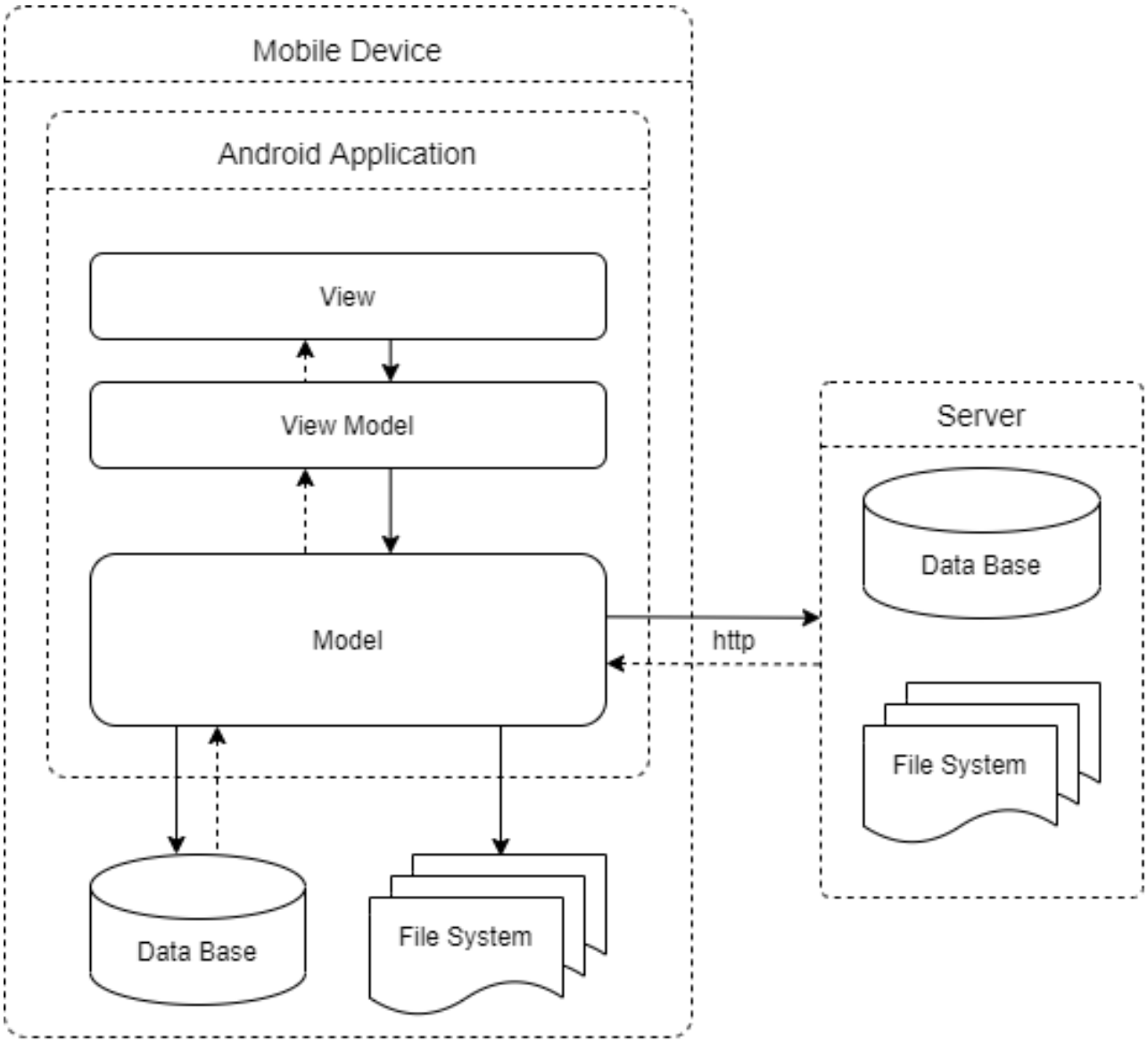

Рисунок 7 – Архитектура разрабатываемой системы

2.2 Динамическая модель системы

Ключевая логика приложения – это работа с маршрутами. В данном разделе приведены диаграммы последовательностей [6], которые наглядно отображают взаимодействие различных компонентов системы при работе с маршрутами.

На рисунке 8 представлена диаграмма последовательности для прецедента «Получение списка маршрутов», на которой отображен процесс получения клиентским приложением списка маршрутов с сервера.

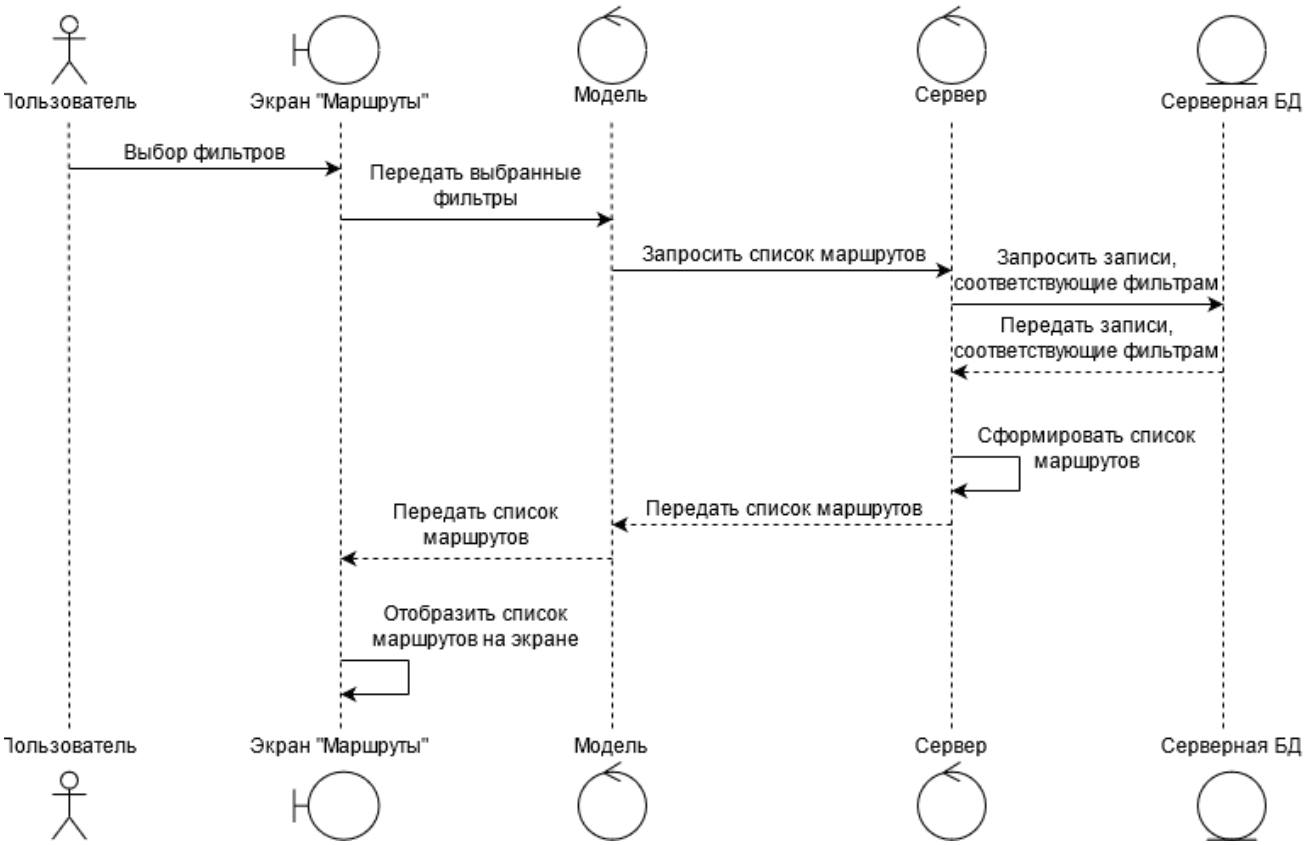

Рисунок 8 – Диаграмма последовательности «Получение списка маршрутов»

На рисунке 9 изображена диаграмма последовательности для прецедента «Запись маршрута», представляющая процесс получение текущих GPS координат устройства, а также их запись в локальную БД.

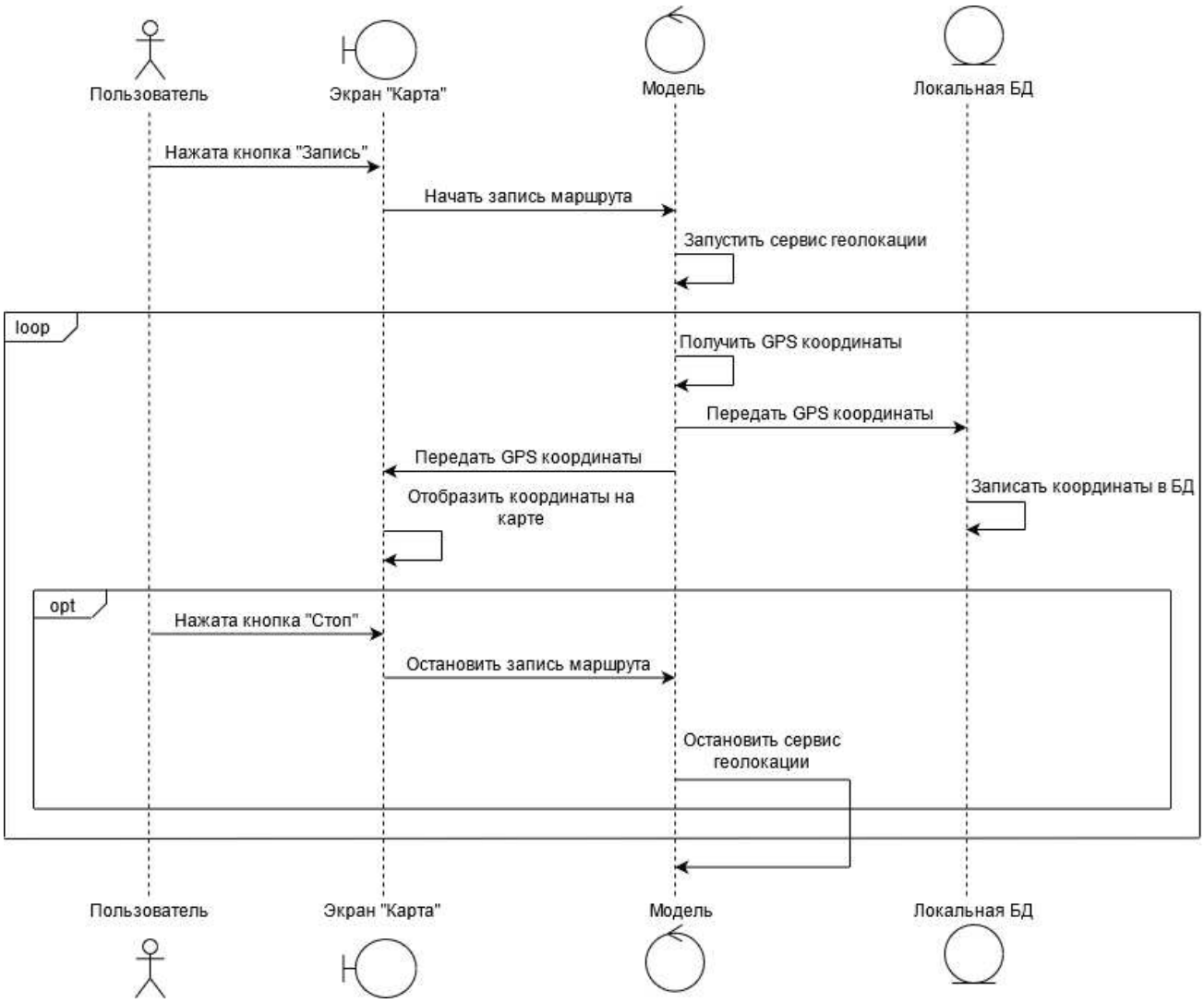

Рисунок 9 – Диаграмма последовательности «Запись маршрута»

На рисунке 10 изображена диаграмма последовательности для прецедента «Сохранение маршрута», которая описывает процесс сохранения созданного маршрута на локальное устройство или на сервер.

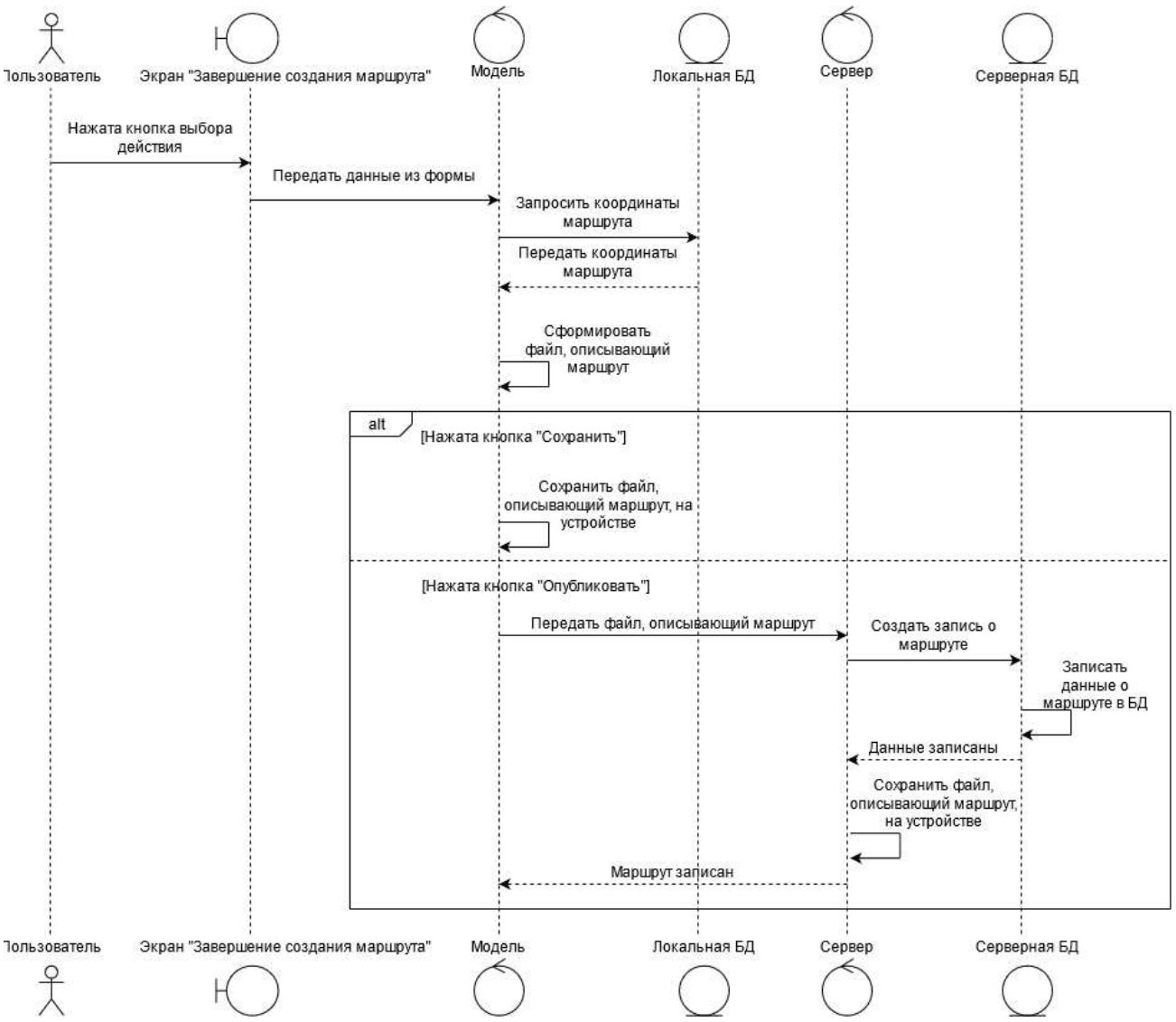

Рисунок 10 – Диаграмма последовательности «Сохранение маршрута»

2.3 Базы данных

Обе части системы, как клиентская, так и серверная, работают с базами данных.

База данных на клиенте содержит 2 таблицы: saved_routes и new_route. Таблица saved_routes хранит записи о маршрутах, сохраненных пользователем локально на устройство, и содержит следующие поля:

– идентификатор маршрута;

– название маршрута;

– длина маршрута;

– описание маршрута;

– путь до файла с координатами маршрута;

– путь до изображения.

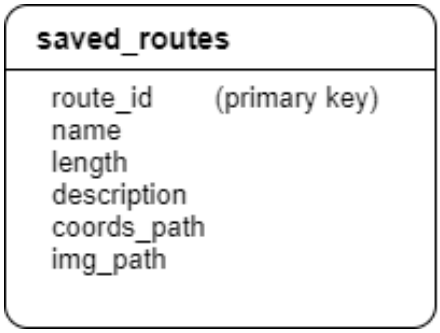

Рисунок 11 – Структура базы данных клиента

Серверная база данных содержит таблицы public_routes и users. Таблица public_routes хранит записи о маршрутах, выложенных в общий доступ, и содержит следующие поля:

– идентификатор маршрута;

– идентификатор создателя маршрута;

– название маршрута;

– длина маршрута;

– рейтинг маршрута;

– описание маршрута;

– связанные теги;

– путь до файла с координатами маршрута;

– путь до изображения;

– сумма оценок рейтинга;

– количество оценок рейтинга.

Таблица users содержит информацию о зарегистрированных в системе пользователях и содержит следующие поля:

– идентификатор пользователя;

– имя пользователя;

– пароль.

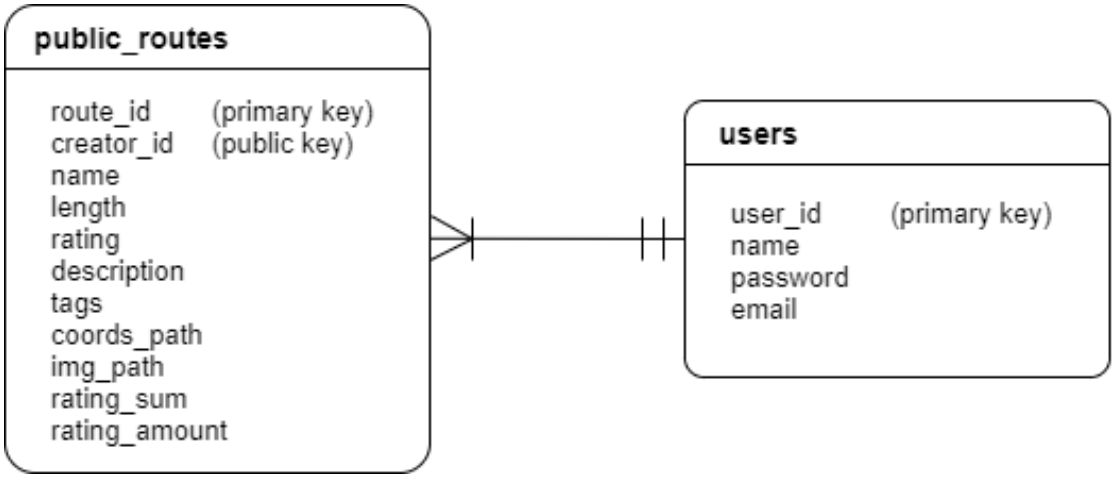

Рисунок 12 – Структура базы данных сервера

2.4 Выводы по главе

В соответствии со спецификацией требований:

– предложена архитектура разрабатываемой системы и структура баз данных;

– с помощью нотации диаграмм последовательностей визуализированы наиболее сложные отношения в системе, более детально проработано взаимодействие объектов.

3 Разработка и тестирование

3.1 Выбор инструментов разработки

Поскольку клиентское приложение разрабатывается для устройств, работающих под управлением ОС Android, наиболее целесообразно для его разработки использовать язык программирования Kotlin с использованием Android SDK.

Для работы с картами выбран Mapbox SDK [9], поскольку он поддерживает разработку на языке Kotlin [10], а также имеет обширную документацию.

Для разработки серверной логики выбран язык Golang [11].

Для работы с базой данных на сервере выбрана система управления базами данных MariaDB [12].

3.2 Реализация элементов слоя View

Основное назначение слоя View – это описание того, как будут отображаться данные на экране устройства.

Поскольку приложение разрабатывается с использованием Android SDK, элементы View представлены в виде стандартного компонента Activity, а макеты экранов описываются файлами формата xml. То есть каждый экран приложения реализуется с помощью двух компонент:

– layout – описывает графические элементы;

– activity – описывает поведение графических элементов и их взаимодействие с остальной системой.

Также для описания экранов в приложении используется компонент Fragment, который представляет Activity, которую можно использовать внутри другой Activity.

Для смены основных экранов используется компонент BottomNavigationView, который не входит в Android SDK и поставляется с библиотекой компонентов Android Material Components [13].

BottomNavigationView – это компонент, который предоставляет инструмент для создания навигационной панели, на которой можно разместить управляющие элементы, к которым могут быть привязаны определенные события. Например, смена текущей активности или фрагмента.

Компонент BottomNavigationView в разрабатываемом приложении содержит три управляющих элемента (три пункта меню):

– Маршруты

– Карта

– Профиль

При этом по нажатию пользователем на элементы «Маршруты» и «Профиль» на родительской активности сменяются соответствующие фрагменты, а по нажатию на элемент «Карта» сменяется сама активность.

Фрагмент, который реализует экран «Маршруты», для отображения списка маршрутов использует компонент RecyclerView, который по умолчанию реализует паттерн ViewHolder.

Для компонента RecyclerView созданы элементы, необходимые для его корректной работы:

– Файл recycler_item.xml, описывающий элементы, составляющие список;

– RecyclerViewAdapter, необходимый для отображения данных на элементах списка.

3.3 Реализация элементов слоя ViewModel

Слой ViewModel предназначен для хранения данных, отображаемых на экране приложения. При этом, при изменении хранимых данных, ViewModel должна оповещать View, что данные изменились.

По этой причине данные, которые хранят информацию для отображения на экране, реализованы с помощью элемента LiveData. LiveData реализует паттерн Observable, то есть позволяет View подписаться на обновление данных внутри себя: если данные LiveData были изменены, то генерируется обратный вызов, который View впоследствии обрабатывает и обновляет данные на экране.

3.4 Реализация элементов слоя Model

В слое Model представлена основная логика приложения.

Получение текущих координат устройства происходит с использованием компонента FusedLocationProviderClient, который поставляется с библиотекой компонентов Google Play Services [14]. С помощью компонента FusedLocationProviderClient задаются параметры, необходимые для обращения к устройству GPS.

После запуска FusedLocationProviderClient с определенной периодичность генерирует обратные вызовы, в которых передается геолокация устройства.

Поскольку приложение должно иметь возможность получать геолокацию устройства при отключенном экране, то целесообразно вынести логику получения геолокации в отдельный поток, который не будет зависеть от жизненного цикла активности. Исходя из этого был создан класс GeolocationService, который наследует стандартный класс Service, позволяющий запускать некоторую логику в потоке, чей жизненный цикл не зависит от активности.

3.5 Инструкция к сборке проекта

Для сборки клиентской части системы необходимо установить на устройство среду разработки Android Studio. Если разработка ведется на ОС

Windows, Android Studio можно установить с официального сайта [8]. Если же разработка ведется на Linux, установить Android Studio можно с помощью пакетного менеджера Snap [15]. Для этого в терминале нужно ввести следующую последовательность команд:

```
sudo pacman -S snapd
sudo systemctl enable –now snapd.socket
sudo ln -s /var/lib/snapd/snap /snap
sudo snap install android-studio –classic
```

При установке компонентов необходимо следовать инструкциям по умолчанию.

Когда все компоненты установились, в главном меню выбрать пункт «Get from version control». В открывшемся окне ввести в поле ввода URL адрес git-репозитория проекта [16] и нажать кнопку «Clone».

Для сборки серверной части системы на устройство необходимо установить систему управления базами данных MariaDB и средства для разработки на языке Golang, а также систему управления версиями git.

Для этого в терминале нужно ввести следующую последовательность команд:

```
sudo pacman -S mariadb go git
sudo mysql_install_db --user=mysql --basedir=/usr --datadir=/var/lib/mysql
sudo systemctl enable mysqld
sudo systemctl start mysqld
```

Чтобы создать тестовую базу данных и запустить сервер, нужно выполнить следующие команды:

```
git clone https://github.com/Introhart/vkr_server.git

mysql -u root -p
mysql> CREATE DATABASE vkr_db;
mysql -u root -p vkr_db < vkr_server/db_dump/data-dump.sql
go run vkr_server/server/main.go
```

Используемые команды могут отличаться, в зависимости от изначально установленного в системе пакетного менеджера.

Для корректной работы клиентского приложения в файле проекта /app/src/main/res/values.xml в свойстве server_address необходимо прописать IP адрес сервера.

3.6 Тестирование

Тестирование клиентской части системы проводилось вручную. Тестирование проводилось на основе прецедентов, описанных в главе 1.

Для тестирования серверной части системы был использован инструмент для тестирования API Google Postman [17]. Были составлены тестовые наборы данных, описывающие возможные запросы к серверу, а также описаны возможные ответы сервера на данные запросы. Тестовые наборы данных доступны в git репозитории с проектом [18].

3.7 Выводы по главе

Были получены следующие результаты:

– реализованы клиентская и серверная части системы, в соответствии со спецификацией требований.

– проведено ручное тестирование клиентского приложения;

– для тестирования API сервера составлены тестовые наборы данных;

– разработана инструкция разработчика, которая содержит информацию по установке среды разработки и запуске проекта.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы была разработана система для работы с маршрутами на карте, состоящая из клиентской части, представляющей собой приложение для устройств, работающих под управлением ОС Android, и серверной части.

Также было проведено тестирование используемой системы. Клиентское приложение было протестировано вручную и основано на прецедентах использования приложения, описанных в спецификации требований. Для тестирования API сервера были составлены тестовые наборы.

Разработана документация для разработчиков, в которой описаны настройка окружения для разработки, а также установка и запуск клиентского приложения и сервера.

СПИСОК СОКРАЩЕНИЙ

БД – база данных

GPS – Global Position System

API – Application Programming Interface

SDK – Software Development Kit

MVVM – Model – View – View Model

JSON – JavaScript Object Notation

XML – Extensible Markup Language

URL – Uniform Resource Locator

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Google Play [Электронный ресурс]: – Режим доступа: https://play.google.com/store

2. AllTrails [Электронный ресурс]: – Режим доступа: https://play.google.com/store/apps/details?id=com.alltrails.alltrails

3. Komoot [Электронный ресурс]: – Режим доступа: https://play.google.com/store/apps/details?id=de.komoot.android

4. Hiking Project [Электронный ресурс]: – Режим доступа: https://play.google.com/store/apps/details?id=com.hikingproject.android

5. GPS [Электронный ресурс]: – Режим доступа: https://www.gps.gov/Material Desing [Электронный ресурс]: – Режим доступа: https://material.io/design

6. Буч, Г. Объектно-ориентированный анализ и проектирование с примерами приложений / Г. Буч, Р. А. Максимчук, К. Энгл, Б. Д. Янг, Д. Коналлен, К.А. Хьюстон – Калифорния, 2010.

7. JSON [Электронный ресурс]: – Режим доступа: https://www.json.org/json-en.html

8. Developers Android [Электронный ресурс]: – Режим доступа: https://developer.android.com

9. Mapbox [Электронный ресурс]: – Режим доступа: https://www.mapbox.com/

10. Kotlinlang [Электронный ресурс]: – Режим доступа: https://kotlinlang.org/

11. Golang [Электронный ресурс]: – Режим доступа: https://golang.org/

12. MariaDB [Электронный ресурс]: – Режим доступа: https://mariadb.org/

13. Material Desing [Электронный ресурс]: – Режим доступа: https://material.io/design

14. Developers Google [Электронный ресурс]: – Режим доступа: https://developers.google.com/maps/documentation/javascript/geolocation

15. Snapcraft [Электронный ресурс]: – Режим доступа: https://snapcraft.io/

16. Репозиторий проекта [Электронный ресурс]: – Режим доступа: https://github.com/Introhart/VKR_Again.git

17. Google Postman [Электронный ресурс]: – Режим доступа: https://www.postman.com/

18. Наборы тестовых данных для Google Postman [Электронный ресурс]: – Режим доступа: https://github.com/Introhart/vkr_server/tree/master/test.json

Федеральное государственное автономное
Образовательное учреждение
Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Вычислительная техника

УТВЕРЖДАЮ
Заведующий кафедрой ВТ
_____ О.В. Непомнящий
подпись инициалы, фамилия
«10» 06 2021 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника

Клиент-серверная система для работы с маршрутами на карте

Руководитель Васильев 08.06.2021 ст. преподаватель В.С. Васильев
подпись, дата

Выпускник АЗ 07.06.21 А.Е. Зозулин
подпись, дата

Нормконтролер Васильев 08.06.2021 ст. преподаватель В.С. Васильев
подпись, дата

Красноярск 2021