МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет Компьютерных наук

Кафедра программирования и информационных технологий

Мобильное приложение для путешествий с аудиогидами «GigaGuide»

Курсовая работа

Направление: 09.03.04. Программная инженерия

Зав. Кафедрой_____д. ф.-м. н, доцент С.Д. Махортов

Руководитель _____ ст. преподаватель В.С. Тарасов

Руководитель практики _____ А.В. Москаленко

Обучающийся _____Л.Н. Бордюгова, 3 курс, д/о

Обучающийся ___ _____М.А.Ячный, 3 курс, д/о

Обучающийся _____Д.В. Роготовский, 3 курс, д/о

Обучающийся _____ А.Р. Демидов, 3 курс, д/о

Воронеж 2025

Содержание

Введение

В современную эпоху стремительного развития информационных технологий и цифровизации всех сфер жизни общества туристическая отрасль переживает значительную трансформацию. Традиционные методы организации путешествий, основанные на использовании бумажных карт и путеводителей, постепенно уступают место цифровым решениям, способным предоставить путешественникам более удобный и персонализированный опыт. Особую актуальность приобретают мобильные приложения, которые позволяют туристам самостоятельно планировать маршруты, получать актуальную информацию о достопримечательностях и эффективно ориентироваться в незнакомой местности.

В контексте этих изменений разработка мобильного приложения «GigaGuide» представляет собой актуальное и перспективное направление развития туристической индустрии. Данное приложение призвано объединить в себе функциональность планирования маршрутов, навигации и информационного сопровождения, предоставляя пользователям комплексное решение для организации путешествий.

В работе будет рассмотрен опыт существующих аналогичных платформ, выявлены ключевые требования пользователей и основные технические и организационные аспекты создания сервиса. Итогом исследования станет разработка мобильного приложения для путешествий с аудиогидами, способного удовлетворить потребности людей, проявляющих интерес к истории посещаемых ими городов.

1 Постановка задачи

1.1 Цели проекта

Целью курсового проекта является создание мобильного приложения для путешествий с аудиогидами «GigaGuide». Данное мобильное приложение разрабатывается с целями:

— предоставить пользователям возможность искать и прослушивать аудиогиды к достопримечательностям;

— повышение интереса пользователей к современным цифровым средствам организации путешествий;

— расширение базы пользователей.

1.1 Задачи приложения

Приложение решает следующие задачи:

— Предоставление пользователям альтернативы устаревшему формату экскурсий с живым экскурсоводом;

— Упрощение поиска информации об интересных достопримечательностях в городе;

— Предоставление пользователям качественных аудиогидов с использованием нейросетевых алгоритмов.

1.2 Требования к разрабатываемой системе

1.2.1 Функциональные требования

К разрабатываемой системе выдвигаются следущие функциональные требования:

Анонимный пользователь:

— регистрация в системе;

— авторизация в системе;

— поиск туров и достопримечательностей;

— фильтрация туров;

— сортировка туров и достопримечательностей;

— просмотр информации о туре и достопримечательности;

— прослушивание аудиогидов;

— просмотр маршрута тура на карте.

Авторизованный пользователь:

— обладает всеми функциональными возможностями анонимного пользователя, за исключением авторизации и регистрации;

— просмотр информации своего профиля;

— обновление пользовательского профиля;

— редактирование своего списка избранных туров и достопримечательностей;

— просмотр своего списка избранных туров и достопримечательностей;

— публикация и удаление своего отзыва на достопримечательность.

1.2.2 Требования к оформлению мобильного приложения

Графический интерфейс мобильного приложения должен быть оформлен в единой цветовой палитре. Основная цветовая палитра должна включать не более трех основных цветов и два-три дополнительных оттенка для акцентов. При этом необходимо предусмотреть различные цветовые схемы для светлой и темной тем оформления.

Навигация между экранами должна сопровождаться плавными анимациями, помогающими пользователю понять логику переходов и текущее положение в структуре приложения.

В мобильном приложении должен присутствовать разработанный логотип. Основной шрифт используемый в мобильном приложении должен совпадать с системным.

1.3 Задачи, решаемые в процессе разработки

Процесс разработки мобильного приложения GigaGuide включает в себя решение следующих задач:

— анализ предметной области;

— исследование аналогов;

— создание UML диаграмм;

— проектирование API;

— проектирование структуры базы данных;

— создание макетов графического дизайна мобильного приложения;

— написание технического задания;

— разработка приложения в соответствии с техническим заданием.

2 Анализ предметной области

2.1 Основные понятия и определения

API – набор способов и правил, по которым различные программы общаются между собой и обмениваются данными.

API Gateway – шаблон проектирования, реализующий единую точку входа в микросервисную архитектуру.

REST API – архитектурный стиль веб-служб, который использует протокол HTTP для передачи данных между клиентом и сервером.

Микросервисы – подход к разработке программного обеспечения, при котором приложение строится как набор небольших, автономных, самодостаточных сервисов, каждый из которых решает отдельную бизнес-задачу или функциональность.

Фреймворк – структура, предоставляющая базовый функционал для разработки программного обеспечения. Он представляет собой набор библиотек, инструментов и рекомендаций, которые помогают разработчикам создавать приложения, не начиная с нуля.

Android – Операционная система с открытым исходным кодом, созданная для мобильных устройств на основе модифицированного ядра Linux.

Java – Строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems.

Jetpack Compose – Декларативный набор инструментов от компании Google для создания приложений под ОС Android на языке программирования Kotlin.

JJWT – Библиотека на Java, которая упрощает создание и проверку JSON Web Tokens (JWT).

JWT – Открытый стандарт для создания токенов доступа, основанный на формате JSON.

Kotlin – Кроссплатформенный, статически типизированный, объектно-ориентированный язык программирования, работающий поверх Java Virtual Machine и разрабатываемый компанией JetBrains.

Liquibase – Открытая библиотека для отслеживания, управления и применения изменений схемы базы данных.

PostgreSQL – Объектно-реляционная система управления базами данных (СУБД) с открытым исходным кодом.

Retrofit – Библиотека для языка программирования Java (или Kotlin), которая позволяет удобно выполнять сетевые запросы к удаленным серверам в Android-приложениях.

Аутентификация – Процесс проверки подлинности личности пользователя.

Неавторизованный пользователь – пользователь, не прошедший процесс аутентификации.

Авторизованный пользователь – Пользователь, прошедший процесс аутентификации.

2.2 Актуальность

В современном контексте развития цифровых технологий и трансформации туристической отрасли разработка специализированного мобильного приложения для путешествий с аудиогидами приобретает особую значимость. Статистические данные последних лет демонстрируют устойчивый рост доли самостоятельных путешественников, которые предпочитают организовывать свои поездки без участия туристических агентств. По данным исследований рынка, более 70% туристов активно используют мобильные приложения на различных этапах планирования и осуществления путешествий.

Особую актуальность разработка приобретает в контексте развития внутреннего туризма, где существует значительный дефицит качественных цифровых решений для самостоятельного планирования маршрутов.

Использование современных технологий позволяет создать масштабируемую архитектуру, способную адаптироваться к растущим потребностям пользователей и обеспечивать высокое качество обслуживания.

2.3 Обзор аналогов

Изучение существующих решений в сфере туристических аудиогидов является важным этапом при разработке мобильного приложения GigaGuide. Проведённый анализ позволяет глубже понять ожидания и поведение потенциальных пользователей, определить их ключевые потребности при планировании самостоятельных путешествий, а также сформулировать концептуальные и функциональные ориентиры разрабатываемой системы. Полученные сведения помогают выстроить удобный, технологически современный и ориентированный на пользователя продукт, максимально соответствующий требованиям аудитории.

В рамках исследования были рассмотрены наиболее показательные платформы, которые могут представлять как прямую, так и косвенную конкуренцию GigaGuide. Основываясь на результатах анализа, была составлена структурированная оценка их функционала, интерфейсных решений и пользовательского опыта. Эти данные стали основой для сводных таблиц, обобщающих информацию по ключевым параметрам. Такая систематизация способствует более точной выработке проектных решений и позволяет объективно определить конкурентные преимущества и зоны роста нового приложения.

Таблица 1 – Анализ мобильных приложений конкурентов

Параметры/категории (наличие)	izi.TRAVEL	WeGoTrip	SmartGuide
Российские города	Да	Да	Нет

Продолжение таблицы 1

Параметры/категории (наличие)	izi.TRAVEL	WeGoTrip	SmartGuide
Просмотр достопримечательностей на карте	Да	Нет	Да
Выбор города	Нет	Да	Да
Выбор цветовой гаммы	Нет	Нет	Да
Нейросетевая озвучка	Нет	Нет	Нет
Оставлять отзывы об аудиогидах	Да	Да	Нет
Добавление аудиогидов в избранное	Да	Да	Да

2.3.1 izi.TRAVEL

izi.TRAVEL — одно из самых известных приложений в области цифровых аудиогидов. Оно предлагает бесплатный доступ к большому количеству экскурсионных маршрутов и музейных туров по всему миру. Сервис активно используется как отдельными пользователями, так и культурными учреждениями, благодаря возможности самостоятельной загрузки контента.

Преимущества:

— международная доступность, поддержка десятков языков;

— большой объём контента — охвачены сотни городов и тысячи достопримечательностей;

— большинство туров можно прослушивать офлайн, что удобно в условиях отсутствия интернета;

— сотрудничество с музеями и официальными структурами культуры;

— встроенная GPS-навигация с автоматическим запуском аудиофайлов при приближении к точке.

Недостатки:

— неравномерное качество материалов, особенно тех, что создаются неофициальными авторами;

— устаревший пользовательский интерфейс, сложный для быстрой ориентации;

— отсутствие гибкой персонализации — пользователь не может адаптировать контент под себя;

— часть контента недоступна без подключения к сети, несмотря на заявленную офлайн-поддержку.

Платформа представляет собой зрелый продукт, но в её структуре присутствует перегруженность, отсутствие интеллектуальной сортировки и персональных рекомендаций. Несмотря на это, izi.TRAVEL остаётся одним из ключевых ориентиров в сфере туристических аудиогидов. На рисунке 1 представлен интерфейс приложения izi.TRAVEL.

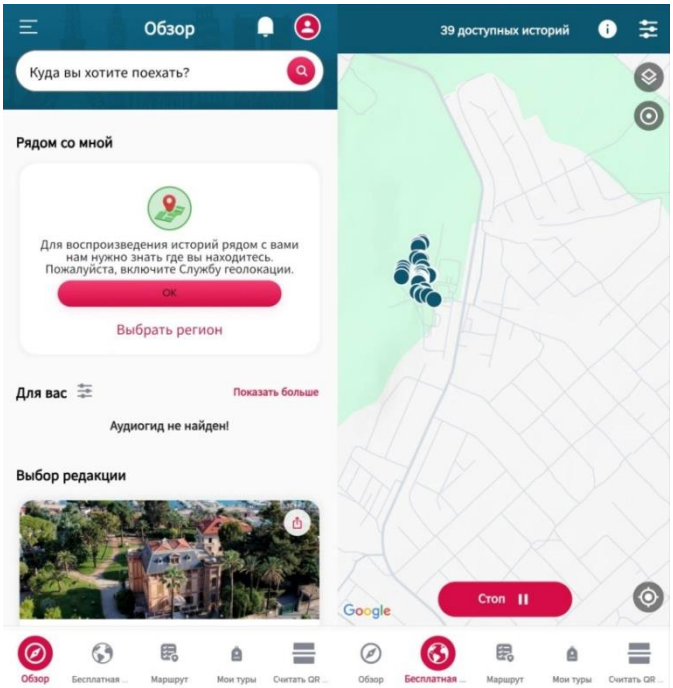

Рисунок 1 – Интерфейс izi.TRAVEL

2.3.2 WeGoTrip

WeGoTrip — российский стартап, ориентированный на международный рынок. Он представляет собой маркетплейс цифровых

экскурсионных туров, доступных для мгновенной покупки и загрузки. Основное внимание уделяется платному контенту, подготовленному профессиональными гидами и локальными экспертами.

Плюсы:

— авторский подход к созданию контента;

— высокое качество подбора маршрутов и озвучивания;

— продуманная структура экскурсий — чёткая логика, достоверность информации;

— наличие англоязычного интерфейса и туров, что открывает доступ к широкой аудитории.

Минусы:

— большая часть контента доступна только за плату;

— фокус на популярных направлениях — слабое покрытие менее известных регионов;

— отсутствие встроенной карты с геотрекингом — маршрут просматривается, но не сопровождается автоматическим GPS-навигационным аудио;

— ограниченные возможности настройки внешнего вида и интерфейса.

WeGoTrip демонстрирует качественный, коммерчески ориентированный подход, но остаётся узкоспециализированной платформой без открытого доступа к базовым функциям для всех категорий пользователей. Это ограничивает его привлекательность для широкой аудитории, особенно внутри стран СНГ. Интерфейс мобильного приложения WeGoTrip изображён на рисунке 2.

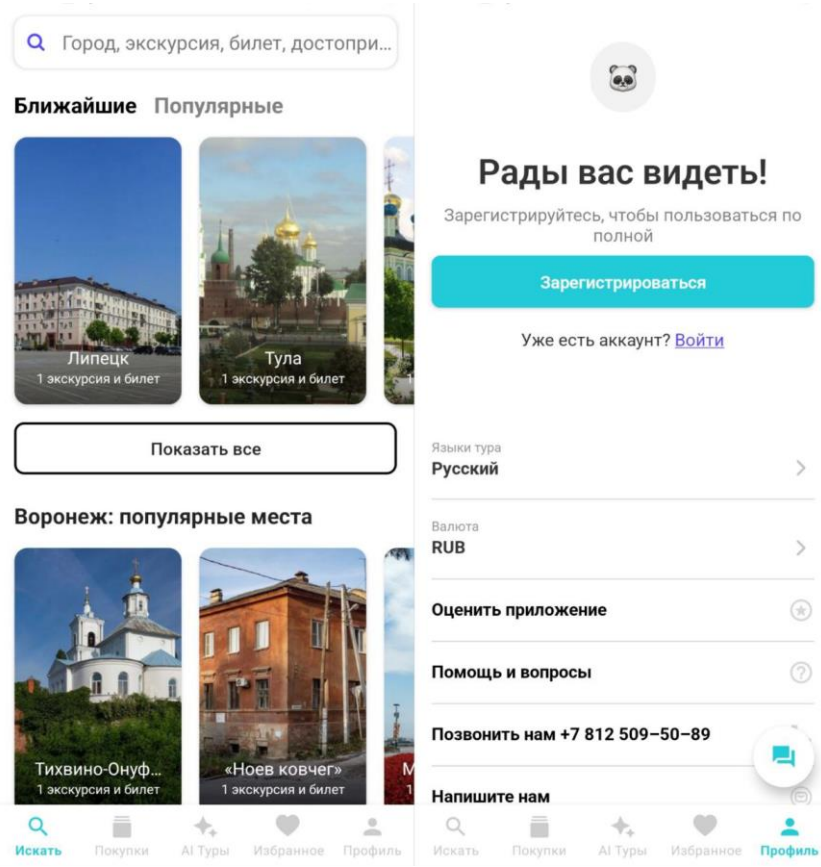

Рисунок 2 – Интерфейс WeGoTrip

2.3.3 SmartGuide

SmartGuide позиционируется как универсальное решение для цифрового туризма. Приложение предлагает как голосовые туры, так и текстовые маршруты. Ключевая особенность — использование автоматизированного искусственного интеллекта для генерации описаний и рекомендаций.

Сильные стороны:

— автоматическая адаптация маршрутов под интересы пользователя;

— поддержка офлайн-режима;

— широкий выбор тем и форматов — от городских экскурсий до прогулок по природе;

— наличие функции «интерактивного гида» — пользователь может задать интересующую тему и получить релевантный маршрут.

Слабые стороны:

— слабое покрытие русскоязычного сегмента — малое количество туров по территории России;

— интерфейс перегружен второстепенными функциями, что затрудняет восприятие основного контента;

— некоторые маршруты автоматически сгенерированы, что сказывается на логике и глубине подачи материала.

SmartGuide представляет собой инновационную платформу с хорошим технологическим потенциалом, однако недостаточно адаптированную под русскоязычного пользователя. Интерфейс приложения SmartGuide изображён на рисунке 3.

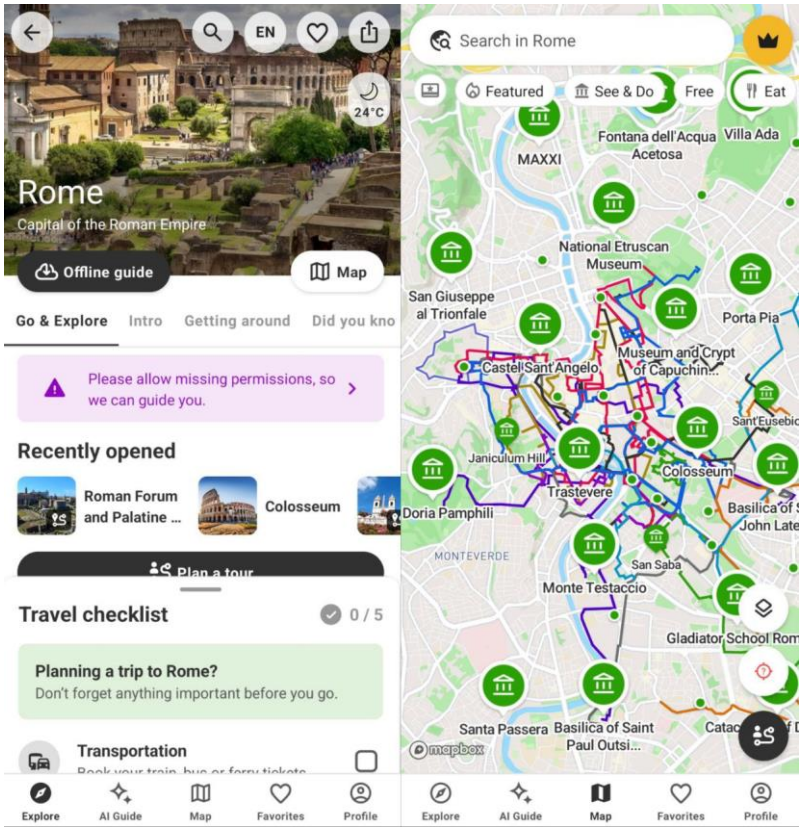

Рисунок 3 – Интерфейс SmartGuide

2.4 Итог анализа

Анализ существующих платформ в сфере цифрового сопровождения туристов позволяет выявить несколько важных закономерностей и направлений для проектирования собственных решений. Во-первых, большинство популярных решений уже покрывают базовые потребности: предоставление маршрутов, озвучка, офлайн-доступ, отображение объектов на карте. Это задаёт высокую планку для функциональности новых приложений.

Во-вторых, очевидна проблема дифференциации. Многие приложения дублируют друг друга по сути, но различаются по степени реализации пользовательского интерфейса, доступности, уровню локализации и ориентации на конкретные группы пользователей. Например, izi.TRAVEL ориентирован на массовую аудиторию, «WeGoTrip» — на платную профессиональную аудиторию, а «SmartGuide» — на эксперименты с автоматизацией.

Проект «GigaGuide» должен учитывать выявленные слабые стороны конкурентов: отсутствие персонализации, перегруженность интерфейса, слабую русификацию, неактуальный контент. Важной задачей становится выработка уникального пользовательского опыта — простого, настраиваемого и визуально понятного. Также важно учесть реализацию голосового сопровождения на базе нейросетевых алгоритмов, что обеспечит живую и естественную подачу материала.

Таким образом, конкурентный анализ не только выявляет сильные стороны рынка, но и демонстрирует ниши, где «GigaGuide» может занять устойчивое положение: это отечественный турист, ожидающий бесплатный, интуитивный, русскоязычный и качественный цифровой гид — решение, которое должно сочетать в себе и полноту информации, и её культурную релевантность.

3 Реализация

3.1 Структура системы

Система реализована на основе микросервисной архитектуры, в которой каждый микросервис отвечает за свою область функциональности. Несмотря на то, что микросервисы логически разделены, все они работают с общей базой данных, что упрощает синхронизацию данных на этапе учебного проекта и позволяет сконцентрироваться на реализации бизнес-логики и взаимодействия сервисов.

Для маршрутизации запросов от клиентов к соответствующим микросервисам в системе используется API-шлюз (API Gateway). Этот шаблон архитектуры был выбран, так как он обеспечивает централизованное управление сквозной функциональностью и независимость клиента от реализации конкретных микросервисов.

Клиентское приложение взаимодействует только с API-шлюзом, а он уже перенаправляет запросы во внутренние микросервисы. Это позволяет обеспечить модульность и гибкость архитектуры, даже при использовании общей базы данных.

Каждый микросервис предоставляет REST API, реализует свою бизнес-логику и взаимодействует с общей базой данных, обращаясь к соответствующим таблицам. Такой подход позволяет упростить локальную разработку и развертывание, разделить ответственность между модулями, обеспечить масштабируемость и возможность дальнейшего перехода к полной изоляции данных при необходимости.

3.2 Средства реализации

3.2.1 Сервер

Для реализации серверной части системы были выбраны следующие технологии:

— язык программирования Java;

— фреймворк SpringBoot;

— СУБД PostgreSQL;

— объектноехранилище Yandex Object Storage;

— платформа контейнеризации Docker;

— система управления миграциями базы данных Liquibase.

Основным языком разработки был выбран Java. Его преимущества включают платформенную независимость, широкую экосистему библиотек и инструментов, высокую производительность и масштабируемость, а также активное сообщество. Благодаря этим качествам Java позволяет эффективно разрабатывать приложения и адаптировать их под различные среды исполнения без значительных доработок.

В качестве основного фреймворка для серверной логики использован Spring Boot. Он значительно упрощает создание приложений на основе Spring, предоставляя разработчику такие возможности, как автоматическая конфигурация, встроенный веб-сервер, внедрение зависимостей, а также гибкость и модульность при построении микросервисов.

Для хранения данных была выбрана реляционная СУБД PostgreSQL. Её ключевые преимущества — это надежность, высокая производительность, расширяемость, поддержка сложных запросов и полнотекстового поиска, а также широкая система типов**.**

Хранение мультимедийных файлов организовано с использованием YandexObjectStorage. Это надежное и масштабируемое объектное хранилище, совместимое с протоколом Amazon S3. Оно обеспечивает высокую доступность, отказоустойчивость, а также простую интеграцию с приложением и эффективное управление хранимыми данными.

Для упаковки и развертывания микросервисов используется Docker. Он обеспечивает изоляцию приложений на уровне операционной системы, что

позволяет абстрагироваться от инфраструктурных особенностей и гарантирует воспроизводимость окружения.

Для управления схемой базы данных выбрана система Liquibase. Она позволяет автоматизировать миграции структуры базы, сохранять историю изменений, работать в команде, а также обеспечивает согласованность и целостность данных при переходе между версиями схемы. Это упрощает внедрение новых функций и минимизирует ошибки при обновлении структуры БД.

3.2.2 Клиент

Для реализации клиентской части системы были выбраны следующие технологии:

— фреймворк Jetpack Compose;

— библиотека OSMDroid;

— библиотека Retrofit.

— библиотека Coil.

Выбор фреймворка Jetpack Compose обусловлен такими преимуществами, как полная интеграция с экосистемой Android, максимальная производительность за счёт нативного исполнения кода, современный лаконичный декларативный подход к построению интерфейсов, встроенная поддержка Material Design, а также ускоренная разработка с минимальным количеством шаблонного кода. Кроме того, Compose полностью поддерживает аппаратное ускорение и современные API Android, включая Jetpack-библиотеки (ViewModel, Navigation), что упрощает реализацию сложной бизнес-логики без необходимости писать обходные решения.

Для поддержки карт в мобильном приложении была выбрана Android-библиотека OSMDroid, представляющая собой готовый клиент провайдера карт OpenStreetMap, с помощью которого возможно быстро и бесплатно

добавить интерактивную карту в приложение посредством использования всего одного виджета.

Выбор библиотеки Retrofit для выполнения сетевых запросов обусловлен такими преимуществами как максимальная производительность, удобство работы с REST API, строгая типизация запросов и автоматическая сериализация данных, что значительно ускоряет разработку и уменьшает количество шаблонного кода.

Для загрузки изображений с сервера и их отображения была использована бибилиотека Coil, которая предлагает лаконичный и удобный API, позволяющий загружать, кэшировать и отображать изображения посредством простой передачи ссылки в Compose-виджет AsyncImage. Это значительно ускоряет разработку по сравнению с ручным управлением загрузкой

3.3 Диаграмма вариантов использования

На рисунке 4 представлена UML диаграмма вариантов использования, иллюстрирующая функционал анонимного пользователя.

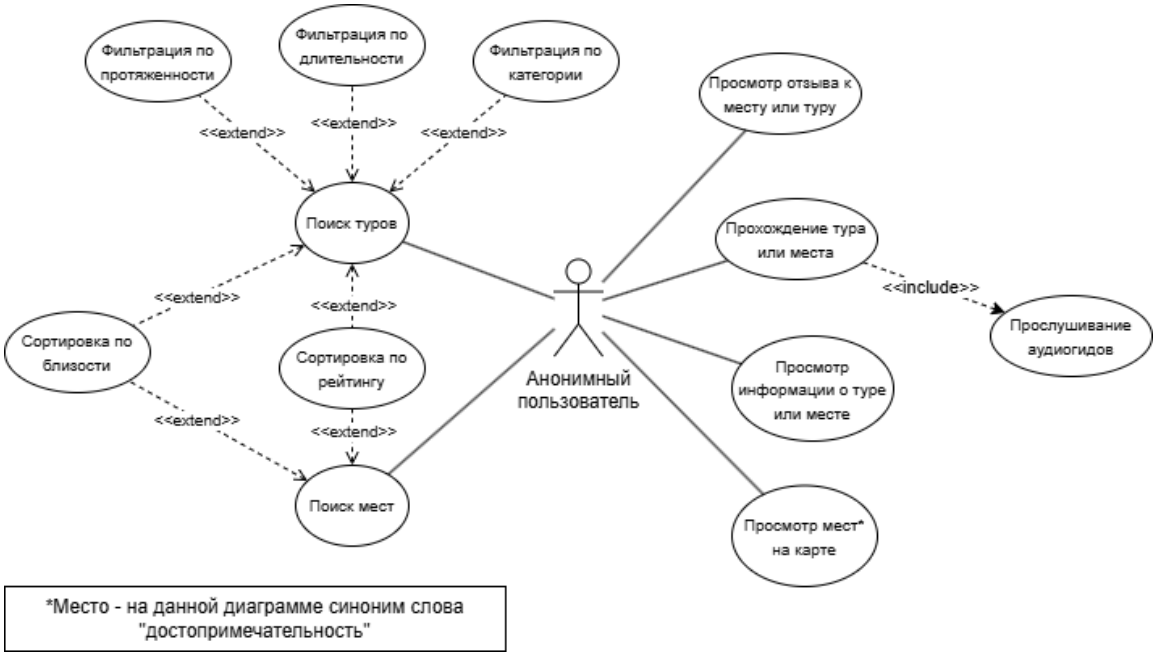

Рисунок 4 – Диаграмма анонимного пользователя

На рисунке 5 представлена UML диаграмма вариантов использования, иллюстрирующая функционал авторизованного пользователя.

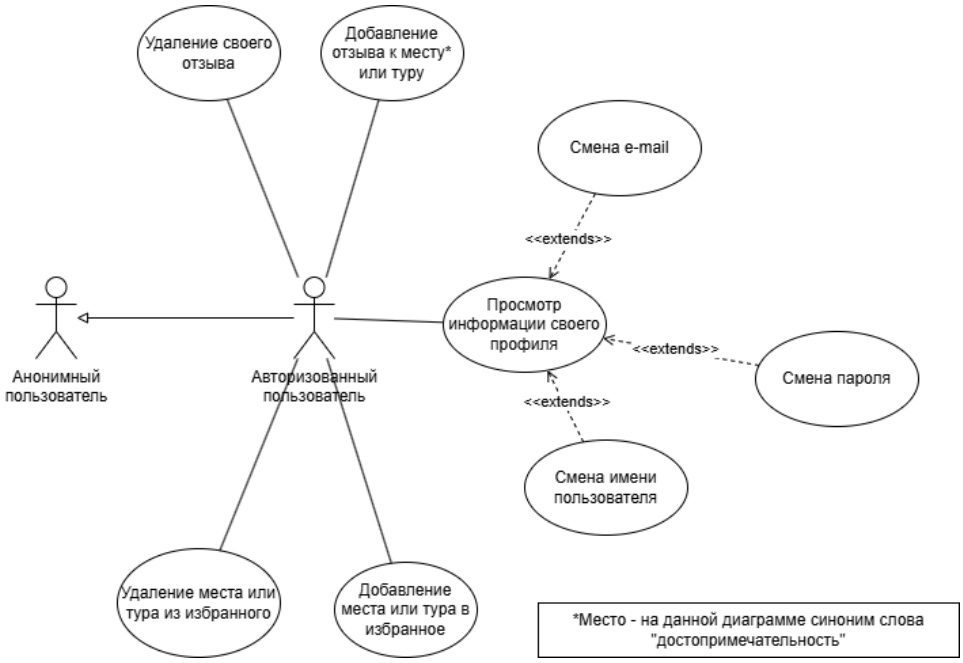

Рисунок 5 – Диаграмма авторизованного пользователя

На рисунке 6 представлена UML диаграмма вариантов использования, иллюстрирующая функционал администратора системы.

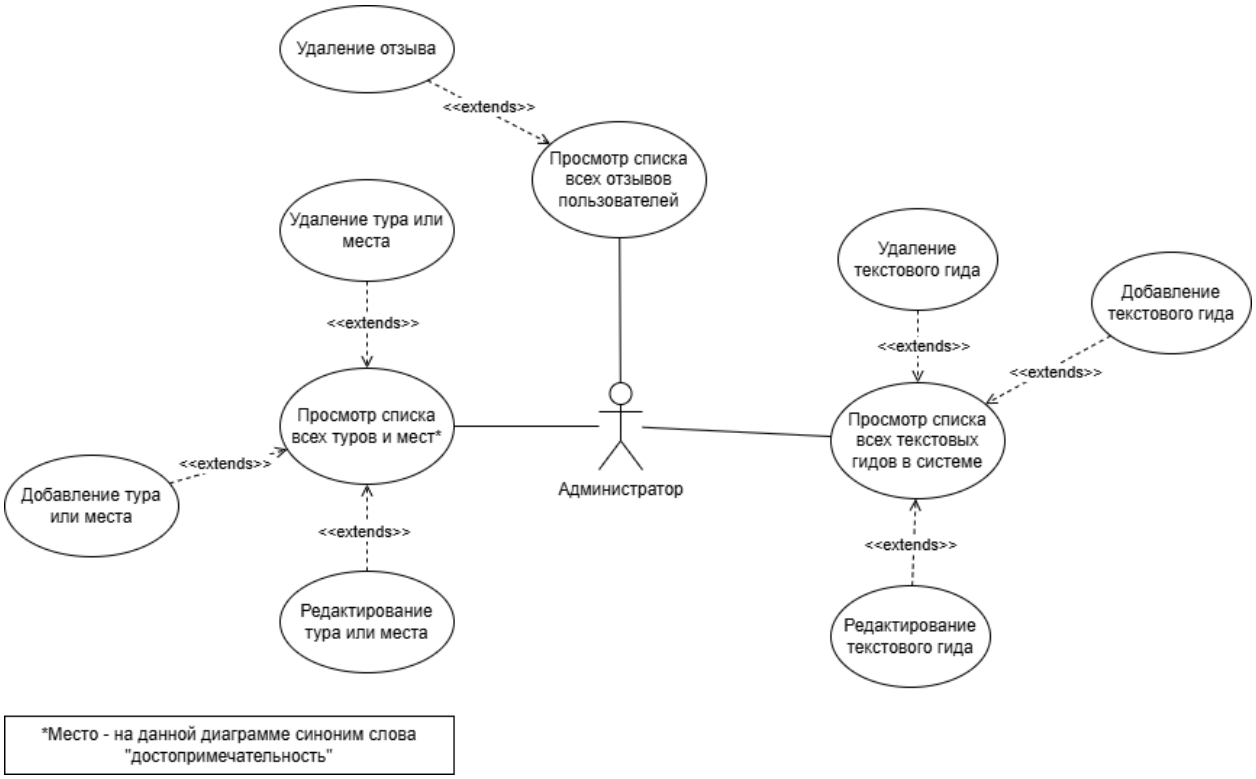

Рисунок 6 - Диаграмма администратора

3.4 Диаграмма активности

На рисунке 7 представлена UML диаграмма активности, иллюстрирующая основные сценарии приложения.

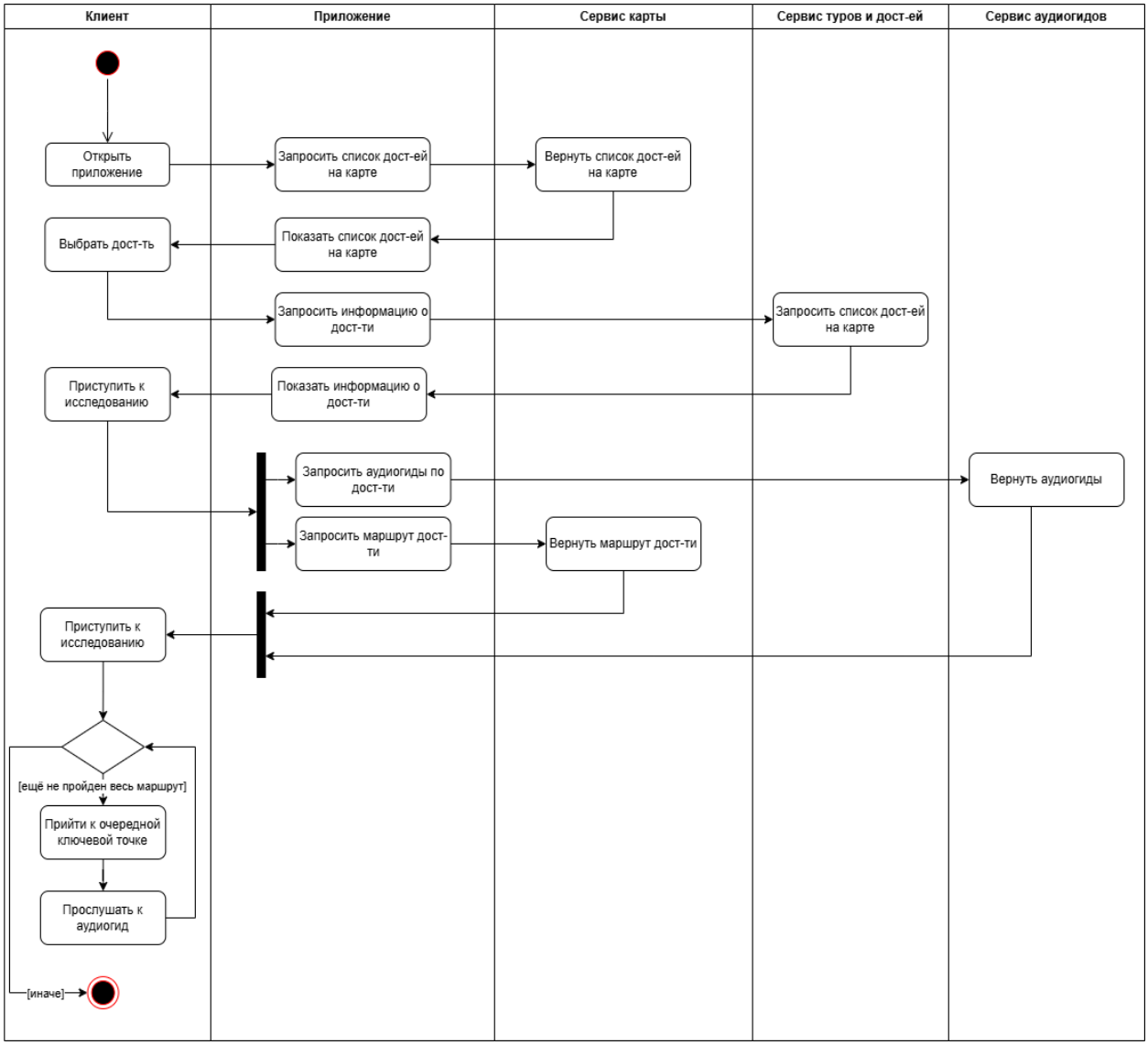

Рисунок 7 – Диаграмма активности

3.5 Диаграмма развертывания

На рисунке 8 представлена UML диаграмма развертывания системы.

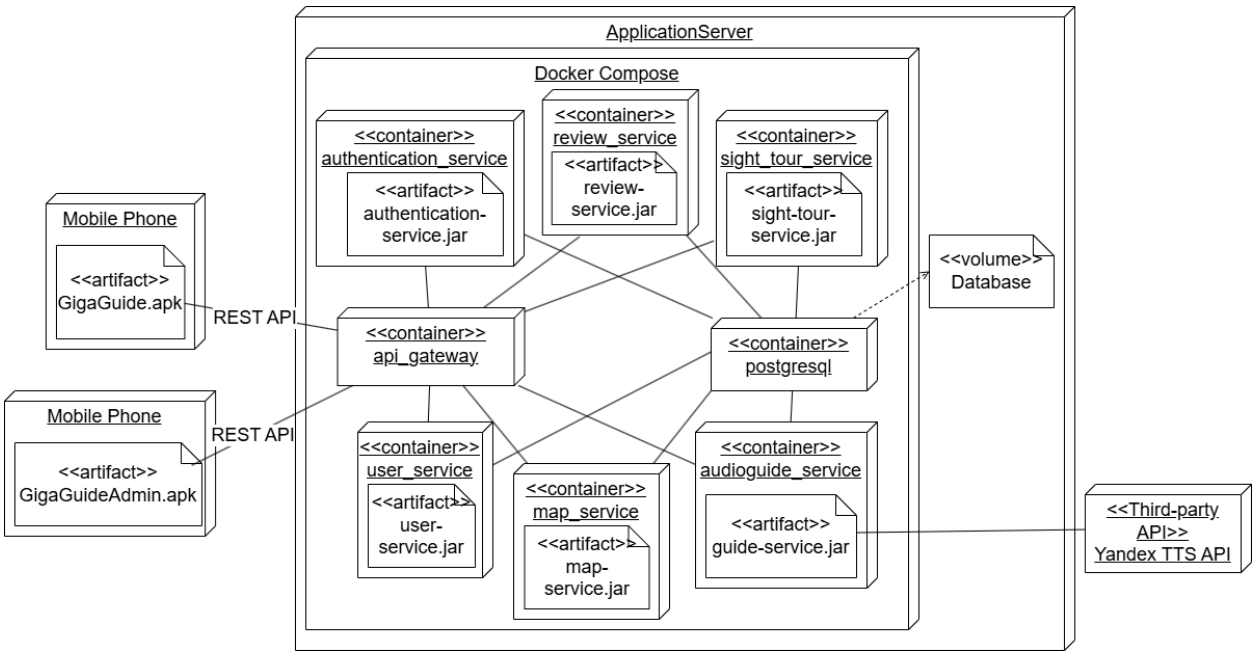

Рисунок 8 – Диаграмма развертывания

3.6 Диаграмма последовательности

Диаграмма последовательности, иллюстрирующая основной пользовательский сценарий, представлена на рисунке 9.

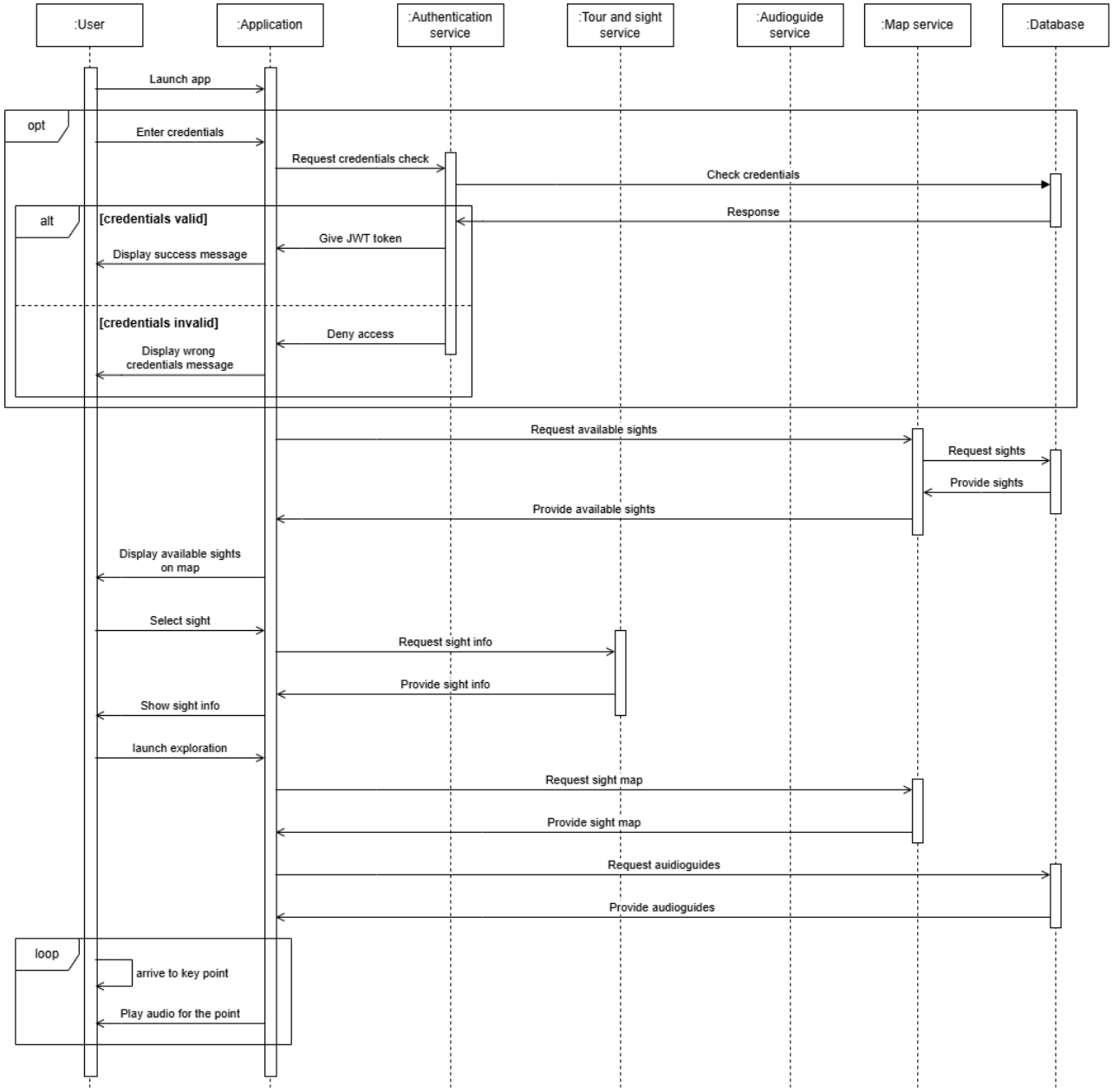

Рисунок 9 – Диаграмма последовательности

3.7 ER-диаграмма базы данных

ER-диаграмма, иллюстрирующая основные сущности базы данных, представлена на рисунке 9.

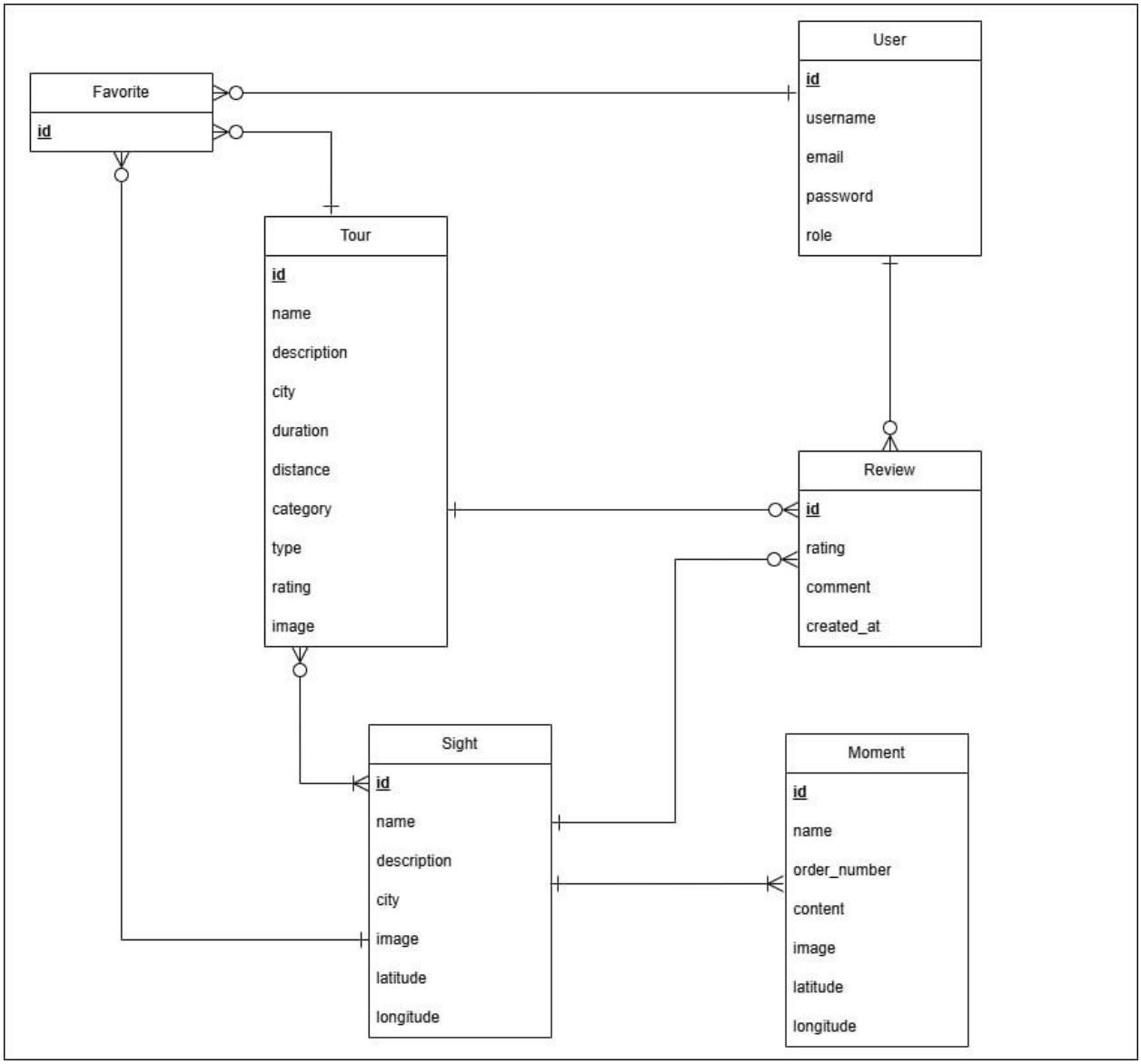

Рисунок 9 – ER диаграмма базы данных

3.8 Серверная часть

Для реализации серверной части приложения использовалась микросервисная архитектура. Все сервисы являются независимыми компонентами, которые взаимодействуют между собой через API, что обеспечивает масштабируемость, упрощает сопровождение и ускоряет внедрение новых функций. Несмотря на это, все микросервисы работают с единой общей базой данных PostgreSQL, что облегчает синхронизацию данных.

Каждый сервис реализован на основе паттерна MVC (Model-View-Controller), что позволяет логически разделить приложение на три слоя: модель, представление и контроллер. Модель отвечает за работу с данными, представление реализует бизнес логику, осуществляет промежуточную обработку данных между контроллером и моделью, а контроллер обрабатывает HTTP-запросы. Такое разделение упрощает разработку, тестирование и поддержку приложения.

3.8.1 Сервис авторизации и аутентификации

Сервис авторизации и аутентификации отвечает за регистрацию новых пользователей, проверку учетных данных при входе в систему и генерацию JWT-токенов, которые затем используются для доступа к защищённым ресурсам. В процессе регистрации сервис проверяет уникальность адреса электронной почты, чтобы исключить возможность создания нескольких аккаунтов с одинаковыми данными. При входе в систему проверяется корректность введённых данных, и в случае успеха пользователю выдаётся токен.

3.8.2 Пользовательский сервис

Пользовательский сервис предоставляет функциональность для работы с пользовательским профилем и избранным. Через данный сервис можно получить данные о текущем пользователе, отредактировать имя,

электронную почту и пароль. При изменении email система проверяет, что такой адрес не используется другим аккаунтом. При изменении пароля требуется указать текущий пароль. Также сервис реализует возможность добавления и удаления туров и достопримечательностей в списке избранного, а также просмотра всех добавленных объектов.

3.8.3 Сервис туров и достопримечательностей

Сервис туров и достопримечательностей реализует основную бизнес-логику приложения. Он обеспечивает добавление, удаление, обновление и поиск туров и достопримечательностей. Поиск осуществляется по городу и названию с поддержкой частичных совпадений. Также реализованы фильтрация по категориям, протяжённости и длительности, а также сортировка по рейтингу и близости к пользователю. Каждый тур содержит название, описание, длительность, протяжённость, категорию, тип и рейтинг, а также связан с набором достопримечательностей. Каждая достопримечательность, в свою очередь, содержит ключевые моменты с текстовыми гидами. Изображения туров, достопримечательностей и моментов хранятся в базе данных в виде ссылок на облачное хранилище.

3.8.4 Сервис аудиогидов

Сервис отвечает за генерацию аудиогидов из текстовых описаний с помощью нейросети YandexSpeechKit. Также поддерживается перевод аудиогидов на английский язык.

3.8.5 Сервис карты

Сервис карты обеспечивает отображение маршрута тура и достопримечательностей на карте, а также расчёт протяжённости и длительности маршрута на основе координат с использованием внешнего сервиса GraphHopper. Расчёт осуществляется между всеми достопримечательностями, входящими в тур, в порядке их следования. Поддерживается использование различных типов маршрутов, таких как

пешие, автомобильные и велосипедные. Полученные данные о расстоянии и длительности маршрута используются как в логике отображения, так и при создании туров.

3.8.6 Сервис отзывов

Сервис отзывов предоставляет функциональность для хранения, отображения, добавления и удаления отзывов о турах и достопримечательностях. Каждый пользователь может оставить отзыв, содержащий текст и рейтинг, и в случае необходимости удалить его. Также предусмотрена возможность удаления любого отзыва администратором.

3.9 Клиентская часть

Для реализации мобильного приложения использовался паттерн проектирования MVVM (Model-View-ViewModel). Это архитектурный подход, который обеспечивает четкое разделение между пользовательским интерфейсом, бизнес-логикой и данными приложения. В рамках данного паттерна View отвечает исключительно за отображение UI и обработку пользовательских действий, ViewModel содержит всю бизнес-логику и преобразует данные из Model в удобный для View формат, а Model представляет собой слой данных и бизнес-правил. Для реализации паттерна BLoC использовался встроенный в Jetpack Compose пакет ViewModel.

Для работы с данными использовался паттерн Repository. Он выступает промежуточным слоем между ViewModel и источниками данных, такими как API, предоставляя единый интерфейс доступа к данным. Это позволяет легко подменять реальные данные на искусственные для тестирования, не затрагивая бизнес-логику приложения.

Общая структура приложения показана на рисунке 10.

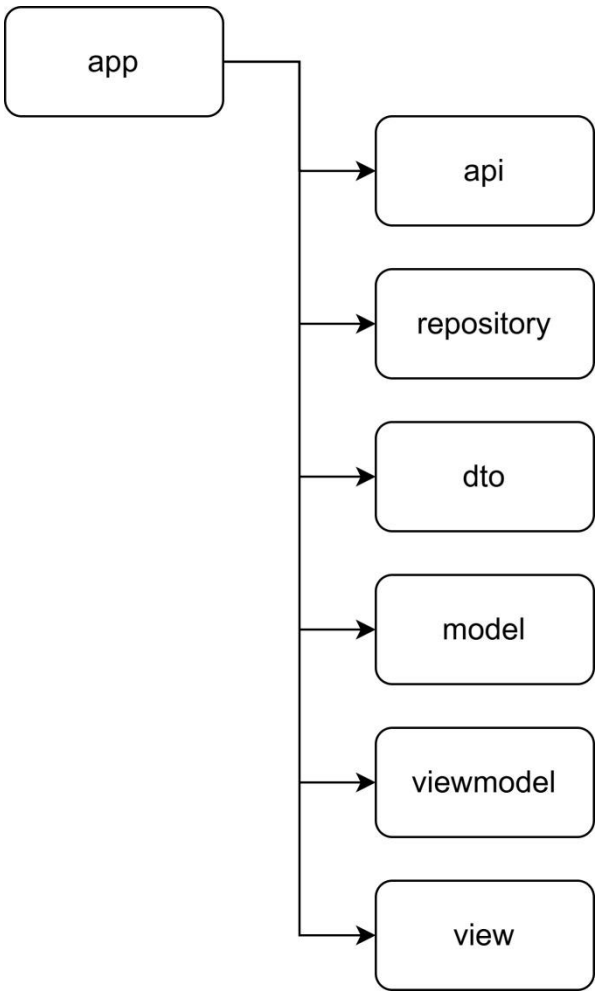

Рисунок 10 – Структура мобильного приложения

Пакет api содержит API-интерфейсы для библиотеки Retrofit, которые задают схему API, конечные точки, формат параметров запросов и ответов.

Пакет repository содержит интерфейсы Repository для основных сущностей приложения, отвечающие за операции по сохранению, полученииию и удалению данных с сервера. Реализующие эти интерфейсы классы используют Retrofit для обращения к серверу.

Пакет dto содержит классы, использующиеся для передачи данных между сервером и клиентом. Их основная цель - упаковать данные в удобную структуру для эффективной передачи. Они содержат только поля данных и не содержат поля, специфичные для бизнес-логики.

Пакет model содержит классы, хранящие данные, необходимые для бизнес-логики и использующиеся для обмена данными между слоями ViewModel и View.

Пакет viewmodel содержит классы, решающие задачи хранения и управления состоянием пользовательского интерфейса, изоляции бизнес-логики от View и автоматического обновления пользовательского интерфейса при изменении данных.

Пакет view содержит графические элементы: экраны и виджеты, необходимые для построения пользовательского интерфейса.

При открытии мобильного приложения пользователя встречает главный экран, на котором расположен список ближайших к пользователю достопримечательностей, а также тех достопримечательностей, которые имеют самый высокий рейтинг в городе. Пользователь может нажать на карточку заинтересовавшей его достопримечательности и перейти на страницу её обзора. Страница обзора достопримечательности содержит такую информацию, как наименование, описание, список названий ключевых точек достопримечательности, а также её рейтинг и количество отзывов. Главный экран и экран обзора достопримечательности показаны на рисунке 11.

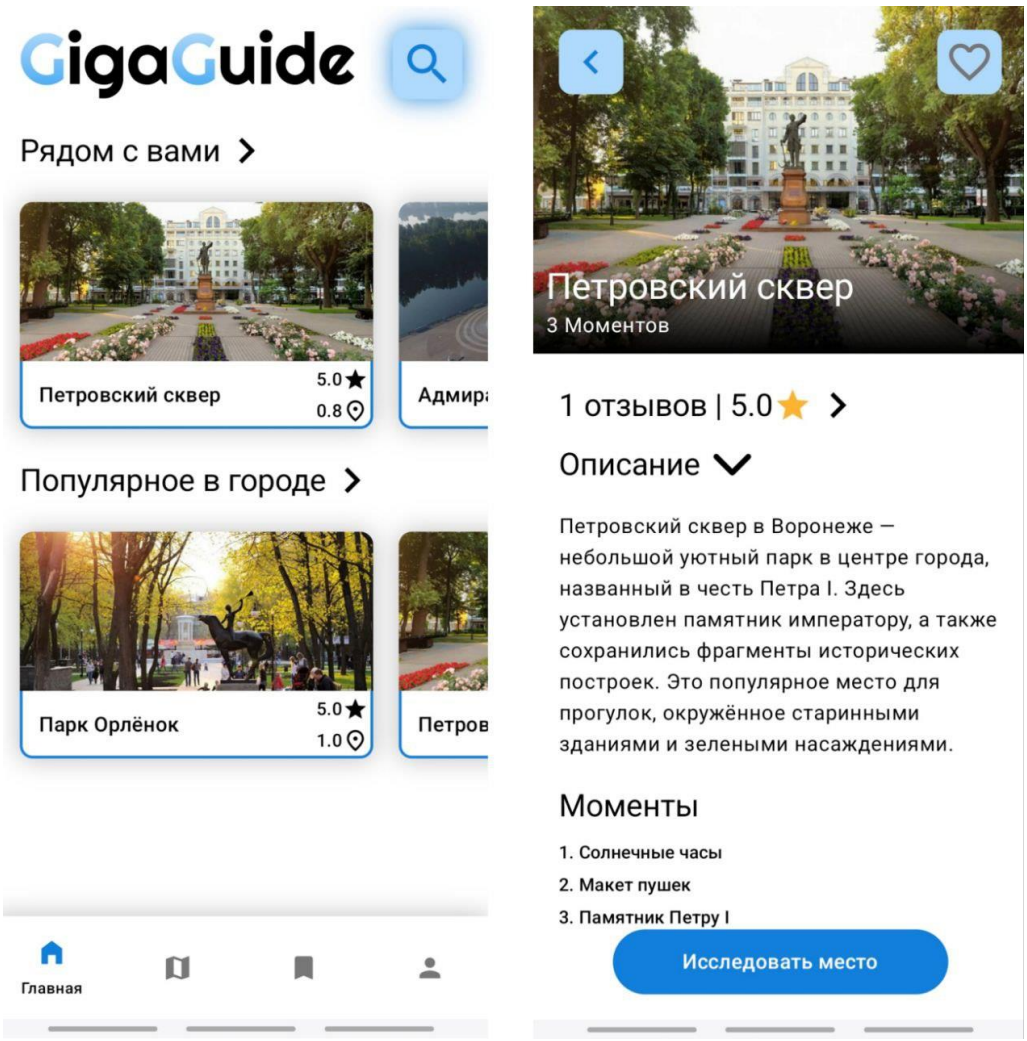

Рисунок 11 – Главный экран и страница достопримечательности

При нажатии на кнопку «Исследовать место» на странице достопримечательности пользователь попадает на страницу её исследования. На карте отображаются ключевые моменты данной достопримечательности, пользователь последовательно проходит все моменты, прослушивая прилагающиеся к ним аудиогиды. Экран исследования достопримечательности показан на рисунке 12.

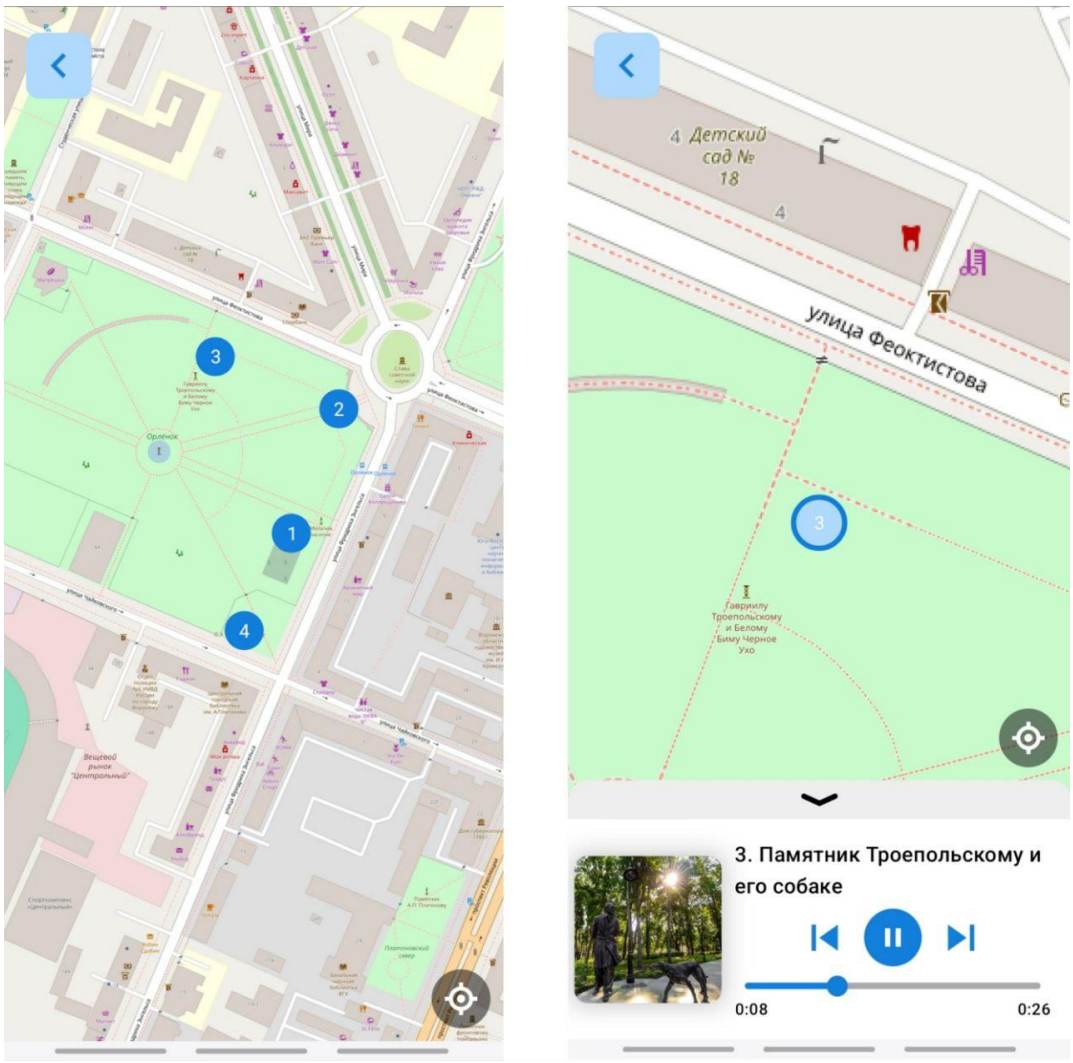

Рисунок 12 – Экран исследования достопримечательности

На странице достопримечательности авторизованный пользователь может нажать на иконку с сердцем в верхней правой части экрана. Таким образом он может добавить достопримечательность в избранное. Экран «Избранное» содержит список карточек добавленных пользователем достопримечательностей. Экран списка избранных достопримечательностей изображён на рисунке 13.

Рисунок 13 – Экран списка избранных достопримечательностей

Пользователю также доступен экран поиска, где он может найти достопримечательности по названию и городу. Также он может отсортировать их по рейтингу и удалённости от своего текущего местоположения. Экран поиска изображён на рисунке 14.

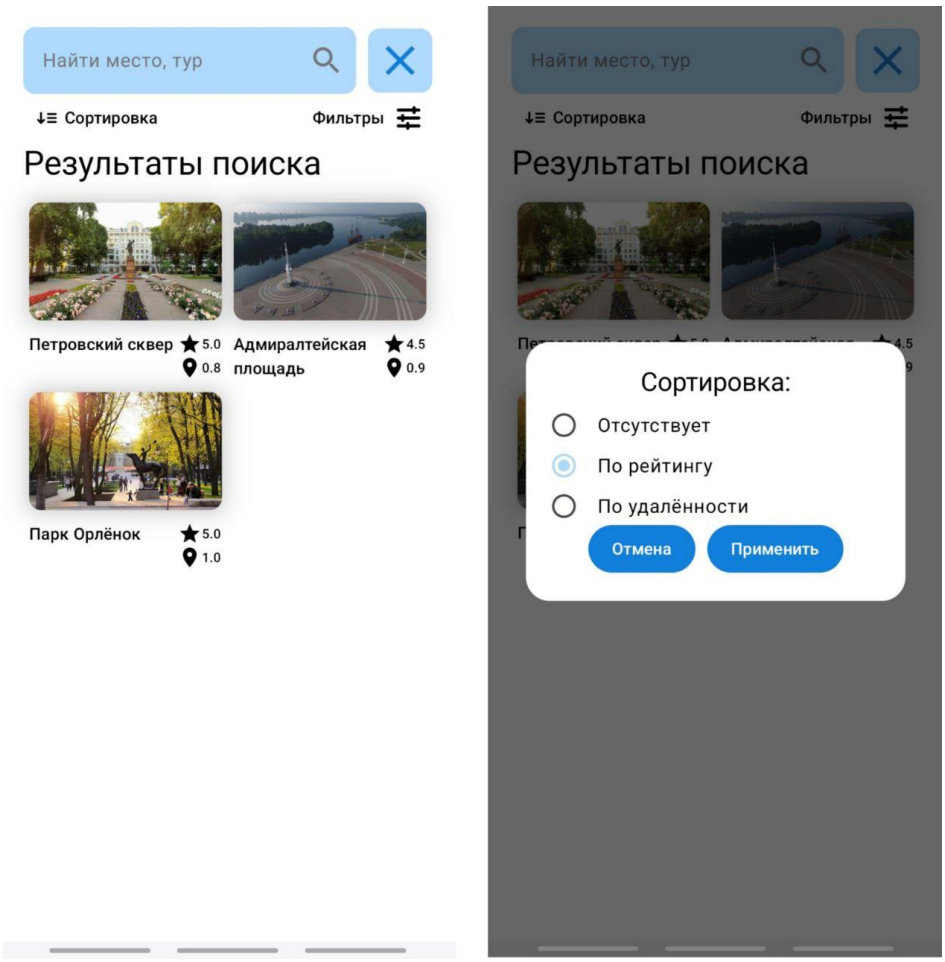

Рисунок 14 – экран поиска

Для администрирование достопримечательностей разработана удобная панель администратор, доступ которой может получить только пользователь с правами администратора. Экраны создания и редактирования достопримечательностей представлены на рисунке 15.

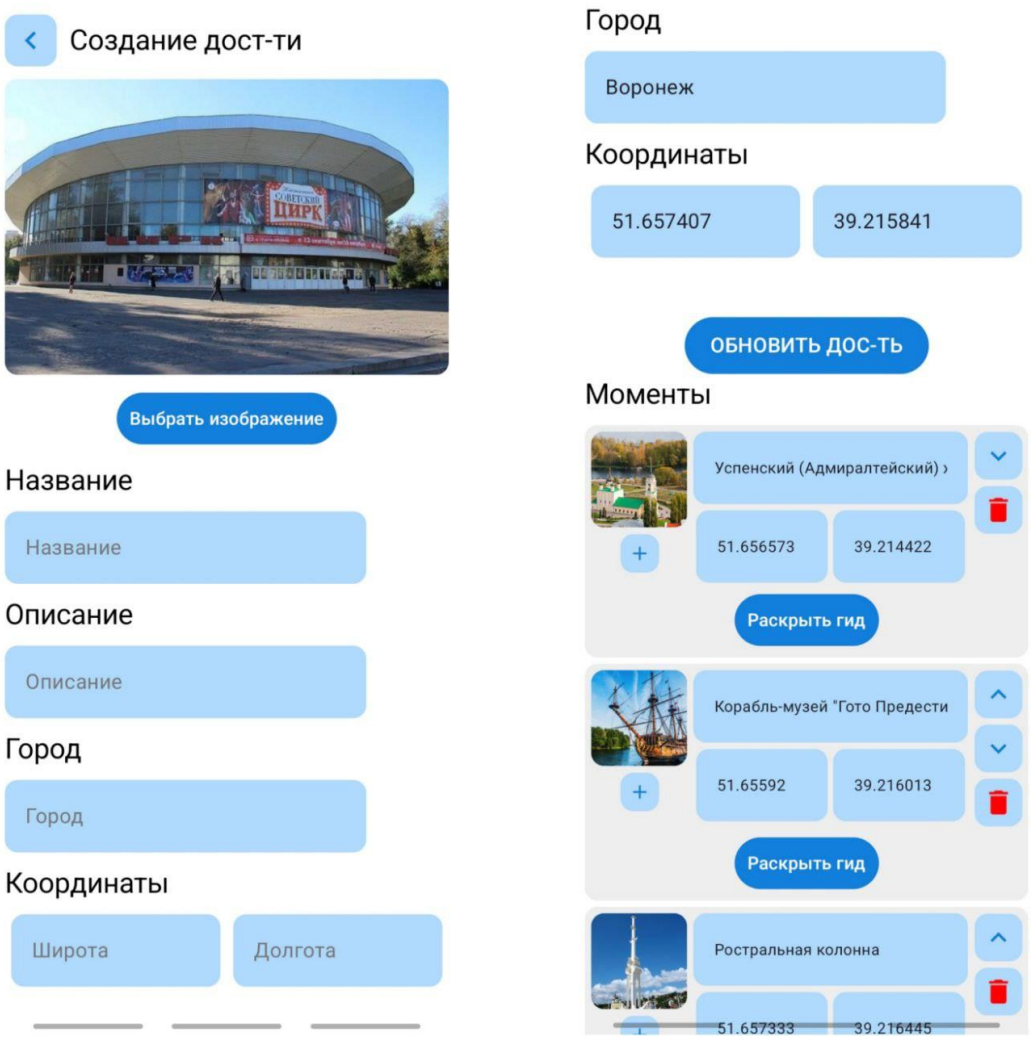

Рисунок 15 – Экраны создания и редактирования достопримечательности в панели администратора

Кроме функции обновления достопримечательностей и аудиогидов, экран предоставляет возможность их удалить.

Заключение

В ходе выполнения курсовой работы были выполнены следующие задачи:

— анализ предметной области;

— исследование аналогов;

— создание UML диаграмм;

— проектирование API;

— проектирование структуры базы данных;

— создание макетов графического дизайна мобильного приложения;

— написание технического задания;

— разработка приложения в соответствии с техническим заданием.

Список использованных источников

1. Документация по Kotlin. – [Электронный ресурс]. – URL: https://kotlinlang.org/docs/home.html (дата обращения: 15.05.2025).
2. Jetpack Compose. Официальный сайт. – [Электронный ресурс]. – URL: https://developer.android.com/jetpack/compose (дата обращения: 15.05.2025).
3. Документация по Retrofit. – [Электронный ресурс]. – URL: https://square.github.io/retrofit/ (дата обращения: 15.05.2025).
4. Spring Boot. Документация. – [Электронный ресурс]. – URL: https://spring.io/projects/spring-boot (дата обращения: 16.05.2025).
5. PostgreSQL. Официальный сайт. – [Электронный ресурс]. – URL: https://www.postgresql.org/ (дата обращения: 16.05.2025).
6. Liquibase – инструмент управления миграциями БД. – [Электронный ресурс]. – URL: https://www.liquibase.org/ (дата обращения: 16.05.2025).
7. Spring Security. – [Электронный ресурс]. – URL: https://spring.io/projects/spring-security (дата обращения: 17.05.2025).
8. JJWT: Java JWT library. – [Электронный ресурс]. – URL: https://github.com/jwtk/jjwt (дата обращения: 17.05.2025).
9. Документация по REST API. Архитектурный стиль взаимодействия клиент–сервер. – [Электронный ресурс]. – URL: https://restfulapi.net/ (дата обращения: 18.05.2025).