Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

РАБОТА ПРОВЕРЕНА
Рецензент
_____
«___»_____ 2020 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____ 2020 г.

Разработка мобильного аудиогида для городских туристических маршрутов

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____Д.В.
Топольский
«___»_____2020 г.

Автор работы,
студент группы КЭ-222
_____В.П. Вертушков
«___»_____2020 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____2020 г.

Челябинск 2020

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____2020 г.

ЗАДАНИЕ

на выпускную квалификационную работу магистра

студенту группы КЭ-222
Вертушкову Владимиру Павловичу
обучающемуся по направлению
09.04.01 «Информатика и вычислительная техника»

1. **Тема работы: «**Разработка мобильного аудиогида для городских туристических маршрутов» утверждена приказом по университету от 24 апреля 2020 г. №627

2. **Срок сдачи студентом законченной работы:** 1 июня 2020 г.

3. **Исходные данные к работе:** техническое задание:
   — язык программирования – Swift;
   — платформа разработки – IOS;
   — в качестве компонента, отвечающего за использование карт, использовать Apple map.

4. **Перечень подлежащих разработке вопросов:**

– анализ предметной области;
– определение требований к системе;
– проектирование приложения;
– реализация приложения;
– тестирование.

5. **Дата выдачи задания:** 1 декабря 2019 г.

Руководитель работы _____/*Д.В.Топольский*/

Студент _____/*В.П.Вертушков* /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и анализ предметной области	01.03.2020	
Определение требований к системе	01.04.2020	
Проектирование приложения	20.04.2020	
Реализация приложения	10.05.2020	
Тестирование	15.05.2020	
Компоновка текста работы и сдача на нормоконтроль	24.05.2020	
Подготовка презентации и доклада	30.05.2020	

Руководитель работы _____/*Д.В. Топольский*/

Студент _____/*В.П.Вертушков*/

АННОТАЦИЯ

Вертушков В.П. Разработка мобильного аудиогида для городских туристических маршрутов – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2020, 57 с., 17 ил., 3 таблицы, библиогр. список – 25 наим.

В данной выпускной квалификационной работе выполнена разработка мобильного аудиогида для городских туристических маршрутов. В ходе выполнения было произведено исследование принципов создания мобильных приложений, был проведен анализ существующих на рынке решений, проведен анализ методов реализации. Разработан алгоритм использования карт телефона, а также голосового помощника. Создано мобильное приложение, содержащее информацию о достопримечательностях города, способное воспроизводить эту информацию при помощи голосового помощника.

Пояснительная записка включает в себя введение, оглавление, основную часть, заключение и библиографический список.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы

В настоящее время мобильные технологии прочно вошли в нашу повседневную жизнь. Мы используем смартфоны и планшеты как для развлечений, так и для решения бизнес-задач. Смартфон тесно связан с человеком в двадцать первом веке. Люди практически не расстаются с мобильными телефонами. В путешествиях туристы часто ориентируются по картам, которые установлены в смартфонах, а информацию о достопримечательностях находят через интернет. Именно поэтому было решено разработать мобильное приложение, которое будет выполнять функции гида: содержать информацию о достопримечательностях, а также воспроизводить эту информацию при помощи голосового помощника.

Описание проблемы

Современные мобильные телефоны представляют собой карманные персональные компьютеры. Смартфоны используются почти в каждой сфере жизни человека. Рынок мобильных приложений может предоставить товар для решения различных задач. Часто мобильные приложения могут заменять целые профессии.

В сфере туризма мобильная разработка представлена очень широко. Существуют приложения для бронирования авиа и ж/д билетов, номеров в отелях. Также можно взять в аренду транспорт используя смартфон. На рынке представлены приложения и для ознакомления с достопримечательностями города. Часто у людей есть необходимость в наличии аудиовоспреизведения туристической информации. Туристам не хочется тратить время и силы на чтение статей об интересных местах города. Для этого на рынке представлены приложения, имеющие функцию аудиогида.

Приложения аудиогиды часто представляют собой карту и набор аудиофайлов, которые необходимо скачать для прослушивания. Основной проблемой данного метода является большой объем занимаемого пространства памяти устройства.

Пользователю необходимо скачивать и удалять аудиофайлы, что бы приложение не занимало излишний объем памяти.

Для решения данной проблемы было решено разработать мобильное приложение, содержащее текстовую информацию о достопримечательностях города, и с функцией воспроизведения при помощи голосового помощника, встроенного в мобильном телефоне. Данное решение позволит решить проблему нехватки памяти, а также сведет к минимуму зависимость приложения от сети интернет.

Для решения описанной проблемы следует разработать мобильное приложение «Аудиогид». Оно улучшит комфорт пользователей в путешествиях, облегчая им планирование их отдыха.

Вопросы, которые нужно решить с точки зрения IT-специалиста:

- выбрать мобильную платформу для приложения;
- выбрать язык программирования;
- выбрать среду разработки;
- исследовать алгоритмы оптимизации построения маршрутов;
- определить группы требований и требования групп.

Правильный выбор инструментов разработки обеспечит компромисс между удобством разработки и лучшей производительностью готового продукта. После выбора инструментов следуют этапы:

- проектирование приложения;
- написание backend кода приложения;
- создание интерфейса приложения;
- тестирование приложения.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор аналогов

После анализа рынка мобильных приложений, не было найдено решений данной проблемы. Существует множество приложений с картами для мобильных устройств. Самым популярным является Google Maps. Данное приложение содержит карту города, информацию об организациях и графике их работы. Но для построения туристического маршрута клиенту необходимо знать местонахождения достопримечательностей. Основным недочетом подобных приложений является отсутствие аудиодорожки с рассказом об интересных местах. Что позволит пользователю с интересом провести время в путешествии.

Существуют так же приложения с аудиогидами. На рынке есть два наиболее популярных: izi.TRAVEL[1] и Guide4Me[2]. Первое содержит информацию о культурных достояниях, таких как музеи, театры, исторические достопримечательности. Пользователь находит интересную для него экскурсию и скачивает необходимую аудиодорожку. После чего, добирается на место и включает аудиогид. Интерфейс данного приложения представлен на рисунке 1.

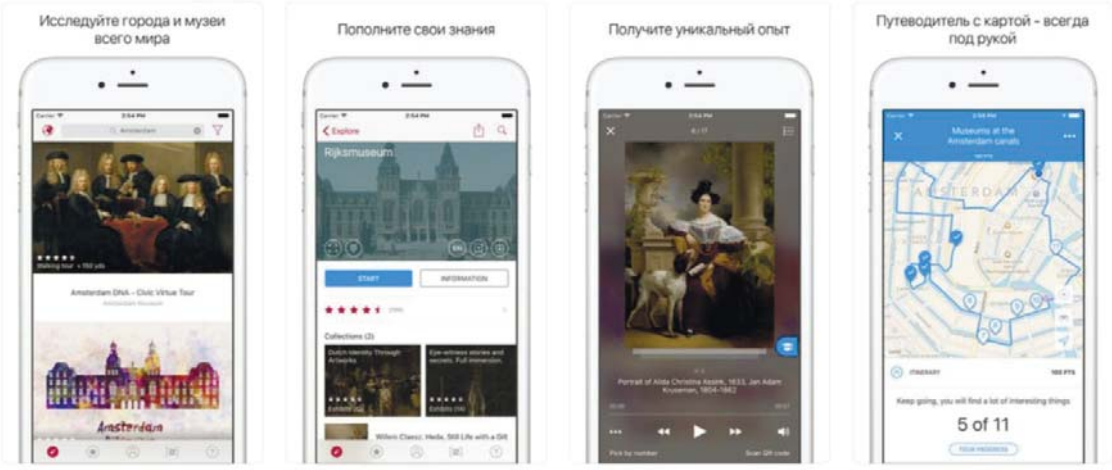

Рисунок 1 – Интерфейс приложения izi.Travel

Guide4Me, в свою очередь, содержит готовые аудиодорожки для различных городов. Пользователь не может построить собственный маршрут, а только выбрать из каталога приложения. Интерфейс приложения представлен на рисунке 2.

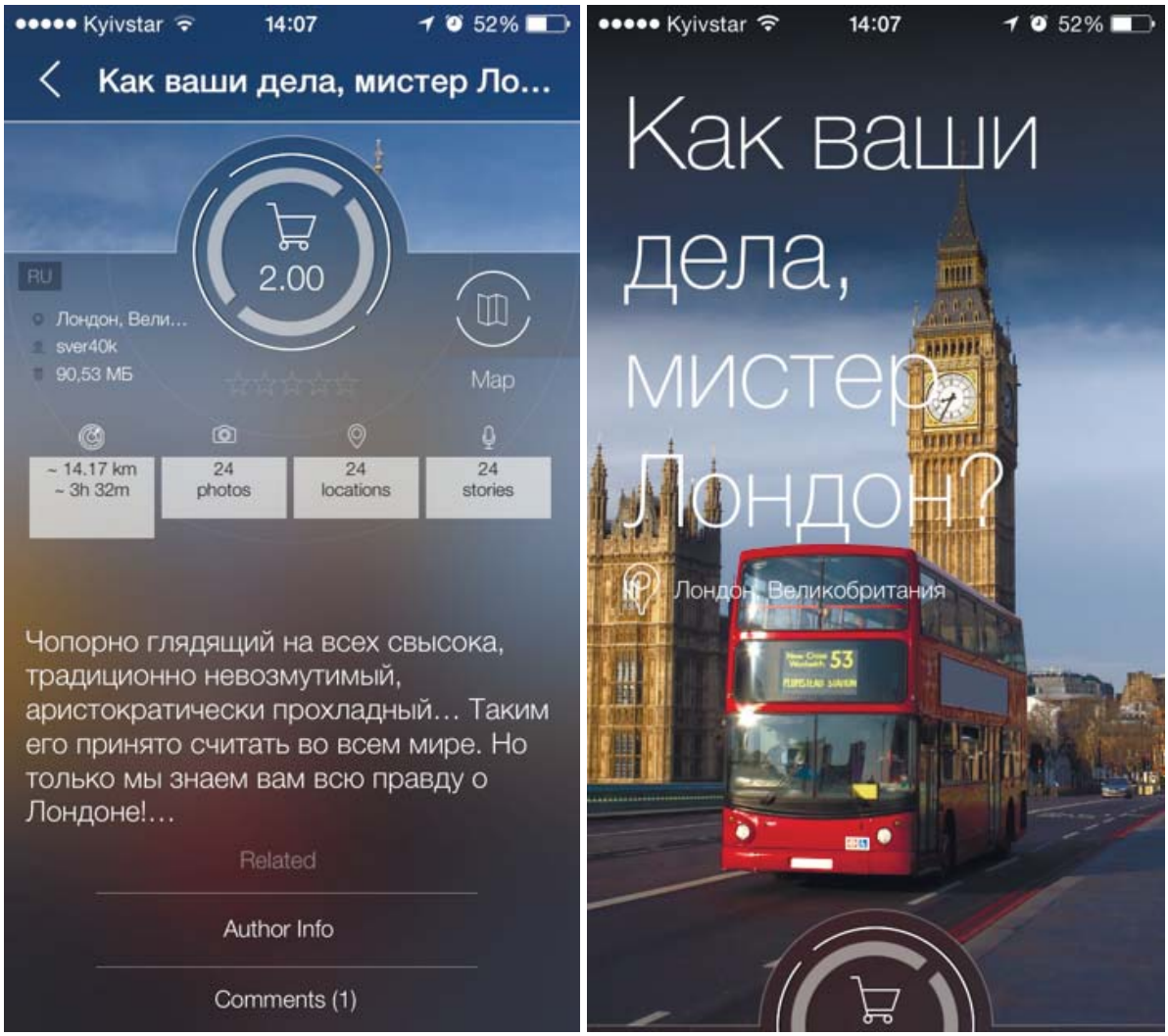

Рисунок 2 – Интерфейс приложения Guide4Me

В таблице 1 приведены популярные существующие решения, их достоинства и недостатки.

Таблица 1 – Преимущества и недостатки имеющихся решений

Наименование приложений	Достоинства	Недостатки
Google maps	• Отображение карты • Построение пеших маршрутов • Посторенние маршрутов на транспорте • Отображение организаций и их графика работы (музеи, театры и т.д.)	• Отсутствие функции постройки туристического маршрута • Отсутствие информации о достопримечательностях • Отсутствие аудиодорожки
izi.TRAVEL	• Наличие аудиодорожки • Наличие гидов по музеям, театрам и т.д. • Наличие гидов по туристическим маршрутам	• Обязательное скачивание аудиодорожки • Большой объем занимаемой памяти • Платные экскурсии
Guide4Me	• Наличие аудиодорожки • Наличие гидов по туристическим маршрутам	• Приложение платное • Обязательное скачивание аудиодорожки • Большой объем занимаемой памяти

1.2 Анализ основных технологический решений

Первым этапом анализа задачи стал выбор операционной системы, для которой будет разрабатываться мобильный аудиогид.

В настоящее время на рынке наиболее распространены две операционные системы:

- Android;
- iOS.

В данной работе было принято решение написания приложения для системы iOS. Выбор был обусловлен растущей популярностью на смартфоны данной операционной системы, а также наличием необходимого функционала.

Для написания программного кода была выбрана среда разработки Xcode. Данная среда обладает необходимым функционалом для реализации данного проекта, а также является бесплатной и распространяется в свободном доступе через Mac App Store.

Веб-сервер служит для принятия данных клиента, хранения отзывов о туристических маршрутах, а также их рейтингов. Веб-сервер будет реализован с помощью фреймворка .NET, на языке программирования C#.

Выводы по разделу один

Задачей данной выпускной квалификационной работы является реализация мобильного приложения аудиогид для городских туристических маршрутов. Приложение должно обладать функцией воспроизведения информации при помощи звукового помощника телефона. Задачу можно разделить на следующие пункты:

- задача написания вэб-сервера;
- задача написания методов использования карт телефона;
- задача написания методов использования функция голосового помощника телефона;
- задачи по написанию интерфейса мобильного приложения.

2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ К СИСТЕМЕ

2.1 Функциональные требования

Для определения функциональных требований необходимо выделить типы пользователей. Для мобильного приложения аудиогид группу пользователей представляют туристы.

Далее необходимо определиться с требованиями для данной группы. Определены следующие функциональные требования для группы пользователей:

1. Интуитивно понятный интерфейс.
2. Наличие информации о достопримечательностях города.
3. Работа приложения без подключения к интернету.
4. Функция воспроизведения информации о достопримечательностях.
5. Функция оставления отзыва о маршруте.
6. Наличие рейтинга маршрутов.
7. Наличие информации о туристических маршрутах.

2.2 Нефункциональные требования

При постановке нефункциональных требований следует выделить следующие пункты:

1. Приложение должно работать на всех iOS уст устройствах (не ниже 12 поколения).
2. Необходимо использовать базы данных для хранения информации о маршрутах и достопримечательностях.
3. Должна быть обеспечена стабильная работа приложения.
4. Должен быть реализован механизм масштабируемости приложения.

Выводы по разделу два

В главе 2 был проведен анализ требований к приложению. Определена целевая группа пользователей, которой являются туристы. Выявлены основные функциональные требования для корректной работы приложения. Также сформированы нефункциональные требования необходимые для реализации приложения, и поддержки необходимого функционала.

3 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

При разработке мобильного приложения необходимо выделить этапы проектирования продукта. Необходимо определить программную среду разработки, операционную систему, для которой будет реализовано приложение, язык программирования. Далее необходимо определить функционал продукта, выбрать методы реализации.

Мобильное приложение должно использовать карты мобильного устройства, хранить информацию о достопримечательностях, а также иметь функцию воспроизведения этой информации при помощи голосового помощника.

Программная часть состоит из:

- интерфейса мобильного приложения;
- backend кода приложения, отвечающего за выполнение необходимого функционала такого как: использование карт устройства, а также голосового помощника;
- вэб-сервера, обрабатывающего запросы клиента к базе данных, хранящей информацию о пользователях, а также отзывы и рейтинги туристических маршрутов.

Первым этапом анализа задачи стал выбор операционной системы, для которой будет разрабатываться мобильный аудиогид.

В данной работе было принято решение написания приложения для системы iOS. Выбор был обусловлен растущей популярностью на смартфоны данной операционной системы, а также наличием необходимого функционала.

В результате разработки приложение должно выполнять следующие функции:

- отображать на карте достопримечательности;
- воспроизводить информацию при помощи голосового помощника;

- хранить информацию о туристических маршрутах.

3.1 Выбор необходимых технологий

3.1.1 Программная среда разработки

Для реализации данной задачи была выбрана программная среда разработки Xcode. Xcode[3] - интегрированная среда разработки программного обеспечения для платформ macOS, iOS, watchOS и tvOS, разработанная корпорацией Apple. Первая версия выпущена в 2003 году. Стабильные версии распространяются бесплатно через Mac App Store.

Данная среда разработки обладает всем необходимым функционалом для создания мобильных приложений под iOS. Также она имеет интуитивно понятный интерфейс. По разработке на Xcode есть литература, находящаяся в открытом доступе.

Исходя из анализа данной программной среды разработки был сделан вывод о том, что Xcode отлично подходит для реализации мобильного аудиогида для туристических маршрутов города.

3.1.2 Язык программирования

Для реализации проекта было рассмотрено два языка программирования: Swift[4] и Objective-C[5]. Для выбора оптимального для реализации проекта языка программирования проведен анализ каждого языка из представленных.

Objective-C

Краткое описание: язык, созданный в начале 80-х годов прошлого века, путём скрещивания C с популярным в то время Smalltalk. Изначально воспринимался, как простая надстройка, модифицирующая некоторые синтаксические конструкции, но после того, как за лицензирование взялась сначала компания NeXT, а потом на правах

преемника и Apple, Objective-C стал одним из наиболее популярных языков для разработки приложений.

Плюсы:

1. Динамическая типизация. В некоторых случаях, это действительно может стать ключевым преимуществом. Например, упрощает создание несложных программ.
2. Документация и сообщества. Более 20 лет успешного применения языка поспособствовали появлению большого количества качественных ресурсов и книг. Сегодня любой, кто желает изучить Objective-C, без труда найдёт ответ на интересующий вопрос на просторах интернета.
3. Относительная простота синтаксиса. Простота заключается в первую очередь в понимании алгоритмов и того, как работает исполняемая машина.

Минусы:

1. Низкая читаемость кода, что затрудняет изучение языка новичками.
2. Динамическая типизация предполагает возможность появления (пропуска) ошибок даже во время компиляции. В частности, банальные опечатки могут серьёзно затянуть процесс.
3. Ограниченная функциональность. Вполне логично, что язык, созданный в прошлом столетии, получил первые изменения в 21 век за 10 минуте, не будет стоять на проходной.
4. Не самая высокая производительность, вызванная динамической типизацией.

Swift

Краткое описание: в 2014 году специалисты Apple представили миру замену привычному Objective-C. Среди преследуемых целей были заявлены повышенная читаемость кода и устойчивость к ошибкам разработчика. Что получилось по факту, давайте посмотрим.

Плюсы:

1. Для написания приложения требуется меньше кода, хотя бы просто потому, что здесь реализован упрощенный принцип работы с повторяющимися строками и заявлениями.
2. Удобен для чтения. Стандартное достоинство любого современного языка.
3. Больше возможностей по сравнению с Objective-C, в частности возможность управлять памятью.
4. Полноценное взаимодействие с кодом, написанным на Objective-C.
5. Повышенная безопасность. Это выражается и в обработке указателей, и в дотошности компилятора, и в том, что в саму компиляцию можно встроить опциональную переменную nil для обеспечения обратной связи.

Минусы

1. Из-за молодости языка и не переведённых на Swift кодов OS X и iOS требуется хотя бы минимальное знание Objective-C;
2. Компилятор выдаёт излишние ошибки, которые разработчику, пришедшему с других языков, покажутся как минимум необычными.

Так же было рассмотрено исследование компании Algoworks[4], которое показало, что Swift намного популярнее прочих конкурентов. Результаты исследования отображены на рисунке 3.

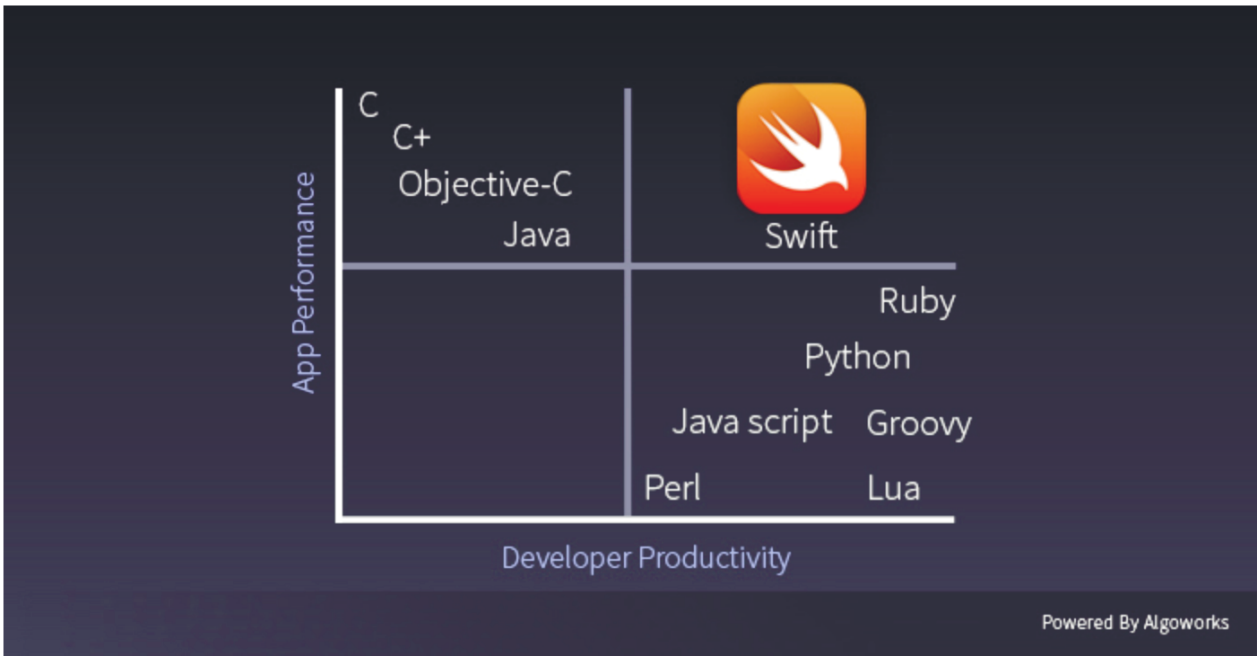

Рисунок 3 – Результаты исследования Algoworks

Исходя из всего вышеперечисленного, языком программирования для реализации мобильного приложения был выбран Swift.

3.2 Архитектура предлагаемого решения

3.2.1 Архитектура мобильного приложения

Структура мобильного приложения состоит из методов использования карт телефона и методов, использующих функции голосового помощника. Также приложение содержит окно с контекстным меню.

При открытии приложения пользователю будет предложен выбор: открыть карту, открыть список туристических маршрутов. При выборе первого пункта меню откроется окно с картой, на которой будет отмечены достопримечательности. При нажатии на вторую кнопку откроется список туристических маршрутов с описанием, при выборе одного из представленных вариантов откроется окно карты с проложенным маршрутом.

Для прослушивания информации о достопримечательности необходимо будет нажать на значок, установленный на данном местоположении на карте.

Для отображения взаимодействия логических частей приложения представлена схема (рисунок 4).

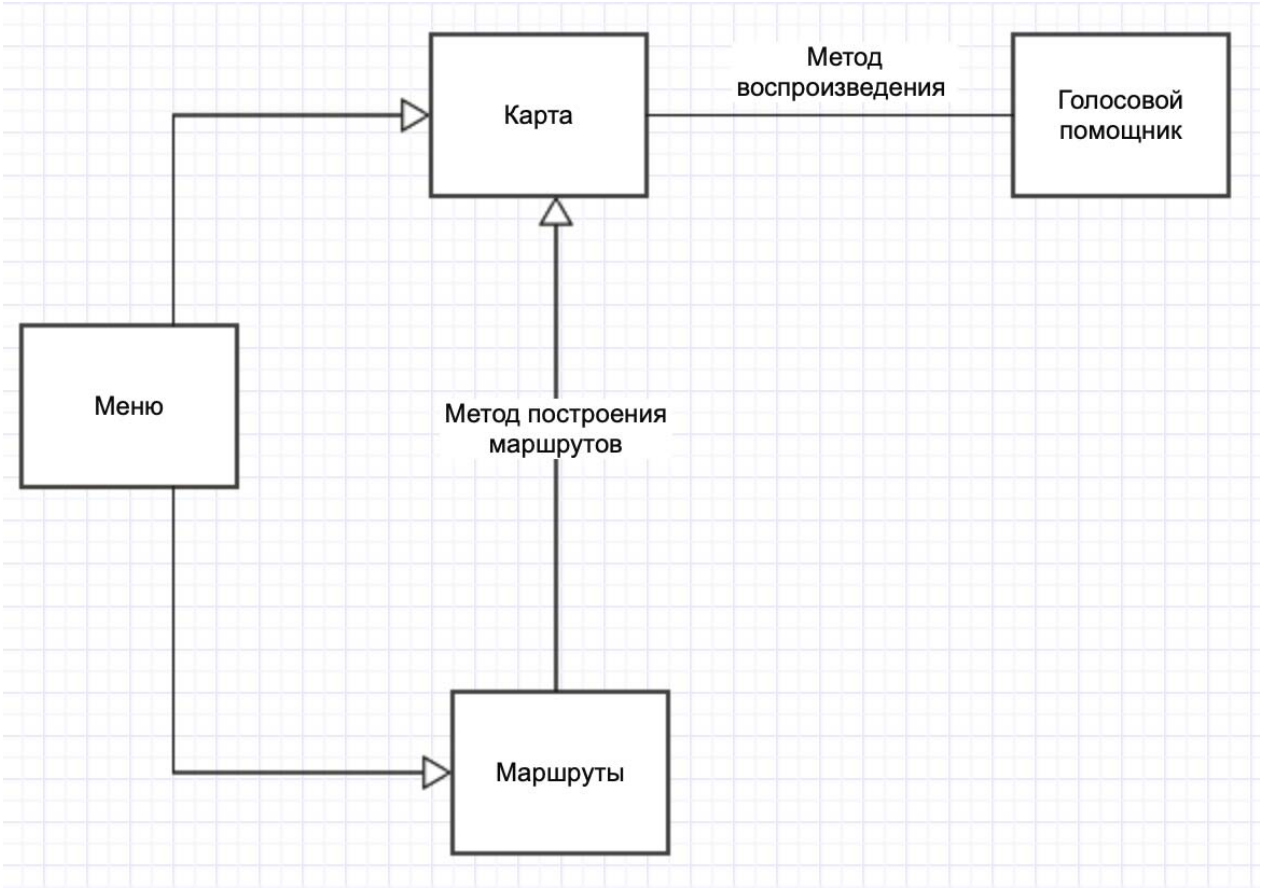

Рисунок 4 – Схема работы приложения

3.2.2 Архитектура веб-сервиса

Методы и средства разработки:

- для разработки серверной части используется платформа .Net язык программирования C# и система WCF;
- для хранения данных будет использоваться PostgreSQL Database;
- для разработки клиентской части мобильного приложения будет

использоваться язык программирования SWIFT.

Логика веб-сервиса включает в себя следующие функции:

- функция добавления нового пользователя;
- функция добавления отзыва;
- функция рейтинга;
- функция получения написанных отзывов и рейтингов.

3.3 Алгоритмы решения задачи

Для решения поставленной задачи было необходимо использовать библиотеки Mapkit[6] и AVFoundation [7].

Mapkit позволяет встраивать карты телефона в окно приложения, использовать методы для построения маршрутов, добавления информации, также позволяет получать доступ к имеющимся публичным отметкам на карте.

Для реализации задачи существует три фундаментальных алгоритма:

- установления маркера на карте;
- определение собственного местоположения;
- построение маршрутов.

Установление маркера на карте проходит по следующему алгоритму:

1. Создается экземпляр класса MKPointAnnotation[8].
2. Устанавливаются координаты широты и долготы типа CLLocationCoordinate2DMake.
3. Устанавливается значения полей title и subtitle.
4. Создается объект типа MKCoordinateRegion для создания региона.
5. Маркер накладывается на карту на верхнем слое.
6. При нажатии на маркер отображается описание местоположения.

Определение местоположения происходит по следующему алгоритму:

1. Создается делегат протокола CLLocationManagerDelegate.
2. Проверятся статус доступа местоположения.
3. Запрашивается объект типа CLLocationManager для получения информации о местоположинии.
4. Рассчитывается регион.
5. На экран выводится маркер с местоположением.

Построение маршрута происходит по следующему алгоритму:

1. Создается делегат протокола MKMapViewDelegate[9].
2. Устанавливается широта и долгота местоположений.
3. Создаются объекты - метки, содержащие метки координаты местоположения.
4. MKMapitems используются для маршрутизации. Этот класс инкапсулирует информацию о конкретной точке на карте.
5. Добавляются аннотации, которые отображают названия меток.
6. Аннотации отображаются на карте.
7. Класс MKDirectionsRequest используется для вычисления маршрута.
8. Маршрут будет нарисован с использованием полилинии по наложенному на карту верхнему слою. Область установлена, поэтому будут видны обе локации;

Для воспроизведения информации голосовым помощником используется библиотека AVFoundation.

Класс AVSpeechSynthesizer производит синтезированную речь из текста на устройстве IOS, и предоставляет методы для управления или контроля речи.

Чтобы произнести некоторое количество текста, необходимо сначала создать экземпляр AVSpeechUtterance[10], содержащий текст. Затем передать его методу speak(_:) на экземпляре синтезатора речи, чтобы произнести это высказывание.

3.4 Алгоритм построения оптимального маршрута

Задача коммивояжера

Задача коммивояжера (Travelling salesman problem, TSP) – классическая задача комбинаторики. Чтобы нагляднее показать ее значимость, ниже приведена краткая история решения поставленной задачи.

Математические проблемы, связанные с задачей коммивояжера, были изучены в 1800-ых годах ирландским математиком сэром Уильямом Роуэном Гамильтоном и британским математиком Томасом Пенингтоном Киркменом. В 1857 Гамильтон создал игру Икосиан, в правилах которой сказано, что участники должны соединить 20 точек додекаэдра так, чтобы каждая точка использовалась не более одного раза, также конечная точка пути должна совпадать с исходной. Эта игра легла в основу появления гамильтонова графа [11].

Задача поиска оптимально пути была впервые рассмотрена с математической точки зрения в 1930-ых годах математиком и экономистом Карлом Менджером в Вене. В 1940-ых статистики Мэхаланобис, Джессен, Гош и Маркс пытались найти применение задаче коммивояжера в сельскохозяйственном секторе, что привело к ее популяризации – начиная с середины 1950ых годах методы решения задачи стали публиковаться в научных журналах.

Хотя задача коммивояжера проста для понимания, ее крайне сложно решить. [12] В 1972 Ричард М. Карп показал, что задача гамильтонова цикла принадлежит к классу NP-полных задач (Nondeterministic Polynomial time), которые подразумевает NP-сложность задачи TSP. [13] Это открытие дало научное объяснение очевидной вычислительной трудности нахождения оптимальных маршрутов.

Методы решения TSP становились более сложными, количество городов возрастало. Ниже представлена краткая история этапов решения задачи коммивояжера.

В 1954 Данциг, Фалкерсон и Джонсон издали описание метода для решения TSP и практически проиллюстрировали его, решив задачу с 49 городами, соединив таким образом города каждого штата Америки. В 1971 году Хельд и Карп решили задачу поиска оптимального пути между 64 городами. Позже в 1977 году, Мартин Гротчел построил путь между 120 немецкими городами. В 1973 Лин и Керниган нашли применение задачи TSP в инженерии – они соединили 318 пунктов, полученных в результате резки лазером. [14]

В 1987 году Голландом и Гроешел была решена задача с 666 городами, позже в этом же году Падберг и Ринальди нашли оптимальный тур, связывающий уже 2392 города. В 1994 году количество городов увеличилось до 7397, спустя 10 лет количество городов достигло до 24978. В 2006 было получено решение задачи коммивояжера с 85900 городами.

Алгоритмы для решения задачи коммивояжера можно разделить на точные (exact algorithm) и неточные (non-exact algorithm). Точные алгоритмы включают в себя перебор всех возможных вариантов, в частных случаях решения могут быть быстро найдены, но в целом осуществляется перебор $n!$ циклов. Вторые в общих случаях применяются для задач, которые невозможно решить точно (вычисление определенных интегралов, решение нелинейных уравнений, извлечение квадратного корня...), если существующие точные решения требуют значительных и неоправданных временных затрат при высокой сложности задачи, и как часть более сложного алгоритма, с помощью которого задача решается точно [15].

3.4.1 Точные алгоритмы

В свою очередь существует две группы точных алгоритмов — одна из них использует методы релаксации линейного программирования TSP: алгоритм Гомори, метод внутренней точки, метод ветвей и границ; вторая, меньшая группа, использует

методы динамического программирования. Характерная особенность методов обеих групп — гарантия нахождения оптимальных решений при общей трудоемкости процесса [16].

Рассмотрим точный алгоритм на примере полного перебора.

Один из самых очевидных методов решения задачи коммивояжера – метод полного перебора или грубой силы. Его суть заключается в переборе всех возможных вариантов путей, алгоритм решения можно записать как:

1. Найти общее число возможных гамильтоновых контуров.
2. Найти вес каждого гамильтонова контура, сложив вес всех его ребер.
3. Выбрать гамильтонов контур с минимальным весом, который и будет оптимальным.

Метод полного перебора обладает рядом преимуществ – он гарантирует

нахождение решения задачи TSP, при этом он прямолинеен и прост в исполнении. В то же время, алгоритм считается неэффективным при работе с большим объемом данных, так как для нахождения оптимального маршрута требует найти вес $(n-1)!$ гамильтоновых контуров.

3.4.2 Неточные алгоритмы

В целом алгоритмы данной группы предлагают потенциально неоптимальные, но быстрые решения. В свою очередь приближенные алгоритмы можно разделить на две категории: приближенные (Approximation Algorithms) и эвристические (Heuristic Algorithms).

Жадный алгоритм (Greedy)

Чтобы решить TSP использование жадный алгоритм, мы исследуем все ребра, выходящие из города-узла, и выбираем n самых коротких дуг. Если те n самых

коротких дуг формируют гамильтонов цикл, тогда мы нашли оптимальное решение. [17]

Трудоемкость решения задачи жадным алгоритмом равна O(n^2). Нижняя граница стоимости оптимального маршрута выше нижней границы Хелд-Карп на 15-20%. [18]

Алгоритм Кернигана – Лина (Lin-Kernighan)

Алгоритм Кернигана – Лина считается один из самых эффективных методов поиска оптимальных или почти оптимальных решений задачи коммивояжера. Однако разработка и реализация алгоритма не проста, так как алгоритм состоит из множества шагов, большинство из которых сильно влияет на работу алгоритма [19]. Создание алгоритма Кернигана было вдохновлено наблюдением, что статическое K в оптимальном методе не дает наилучшее решение. Появилась идея использовать различные стадии -оптимального метода в выполнении эвристического алгоритма. На практике было показано, что практически невозможно заранее предугадать какое K следует использовать, чтобы достигнуть лучшего компромисса между трудоемкостью и качеством решения. Лин и Керниган убрали этот недостаток, введя оптимальную переменную, таким образом значение K меняется во время выполнения алгоритма [20]. Трудоемкость при этом равна O($n^{2.2}$).

Алгоритм поиска с запретами (TabuSearch)

Главная проблема алгоритма ближайшего соседа состоит в частом застревании в точке локального оптимума. Этого можно избежать, применив алгоритм поиска с запретами, в 1977 году предложенный Ф. Гловером. Данный метод позволяет переходить от одного локального оптимума к другому в поиске глобального оптимума, после перехода ребро попадает в список запретов и повторно не используется, кроме случаев, когда оно может улучшить построенный оптимальный путь. На практическом уровне запрещенный набор сохраняется как комбинация ранее посещаемых шагов, который позволяет построить дальнейший путь относительно текущего решения и соседних узлов [21].

Главным недостатком этого метода служит его время выполнения – трудоемкость алгоритма оценивается как $O(n^3)$.

Муравьиный алгоритм (Ant Colony Optimization)

Муравьиный алгоритм – эффективный полиномиальных алгоритм, вдохновленный вдохновлена поведением настоящих муравьев. Впервые его принципы были описаны в 1991 Марко Дориго. Муравьям свойственно сотрудничать в поисках пищевые ресурсы, поэтому они оставляют след химического вещества, феромонов, на их пути от гнезда до источника пищи. [22] Этот тип невербальной коммуникации называют стигмергия – стимуляция, основанная на опыте предыдущих муравьев и направленная на повышение производительности. [23]

Для решения задачи коммивояжера как правило используют около 20 муравьев. Их размещают в случайные города и отправляют в другие города. Им не позволяют дважды посещать один и тот же город, только если они не завершают маршрут. Тот муравей, который выбрал самый короткий тур, будет оставлять след феромонов обратно пропорциональный длине маршрута. Этот след феромонов будет считываться следующим муравьем при выборе города, и с большой вероятность он пойдет тем же путем, еще сильнее укрепив след. Этот процесс будет многократно повторен пока не будет найдет маршрут, достаточно короткий, чтобы быть оптимальным.

Среди недостатков алгоритма хочется выделить, что первое полученное решение может оказаться одним из худших в плане оптимизации, однако при повторном решении метод выдает достаточно точный результат. [24]

3.4.3 Генетический алгоритм

Генетические алгоритмы принадлежат к методам оптимизации, в основу которых легли биологические процессы, протекающие в природе. Чарльз Дарвин в своей эволюционной теории ввел определение естественного отбора, согласно которой особи, более присоленные к условиям окружающей среды, имеют больше шансов на выживание и продолжение рода, и наоборот – неприспособленные осыпи подвергаются избирательному уничтожению. Основой отбора являются мутации генов и их комбинации, формирующиеся при размножении и передающиеся потомству. В ходе естественного отбора выживают экземпляры с наибольшей функцией приспособленности. Это численная характеристика, которая может изменяться в зависимости от условий конкретной задачи. Приспособленные особи скрещиваются (кроссовер) и производят потомство. Случайные мутации также могут оказывать влияние на развитие популяции. На рисунке 5 представлена общая схема генетического алгоритма:

Можно обозначить задачу оптимизации как задачу нахождения функции $f(x_5,x_6,...,x_7)$, носящей название функция приспособленности, которая позволяет выделить наиболее приспособленных особей популяции для размножения и наименее приспособленных, которые вычеркиваются, повышая тем самым приспособленность нового поколения. Требуется, чтобы на области определения выполнялось неравенство $f(x\_1,x\_2,...,x\_n) \geq 0$, область является ограниченной. Параметры функции записывается как строки, состоящие из битов. Строка, полученная конкатенацией строк, называется особью:

1010 10110 101 . . . 10101

$| x1 | x2 | x3 | . . . | xn |$

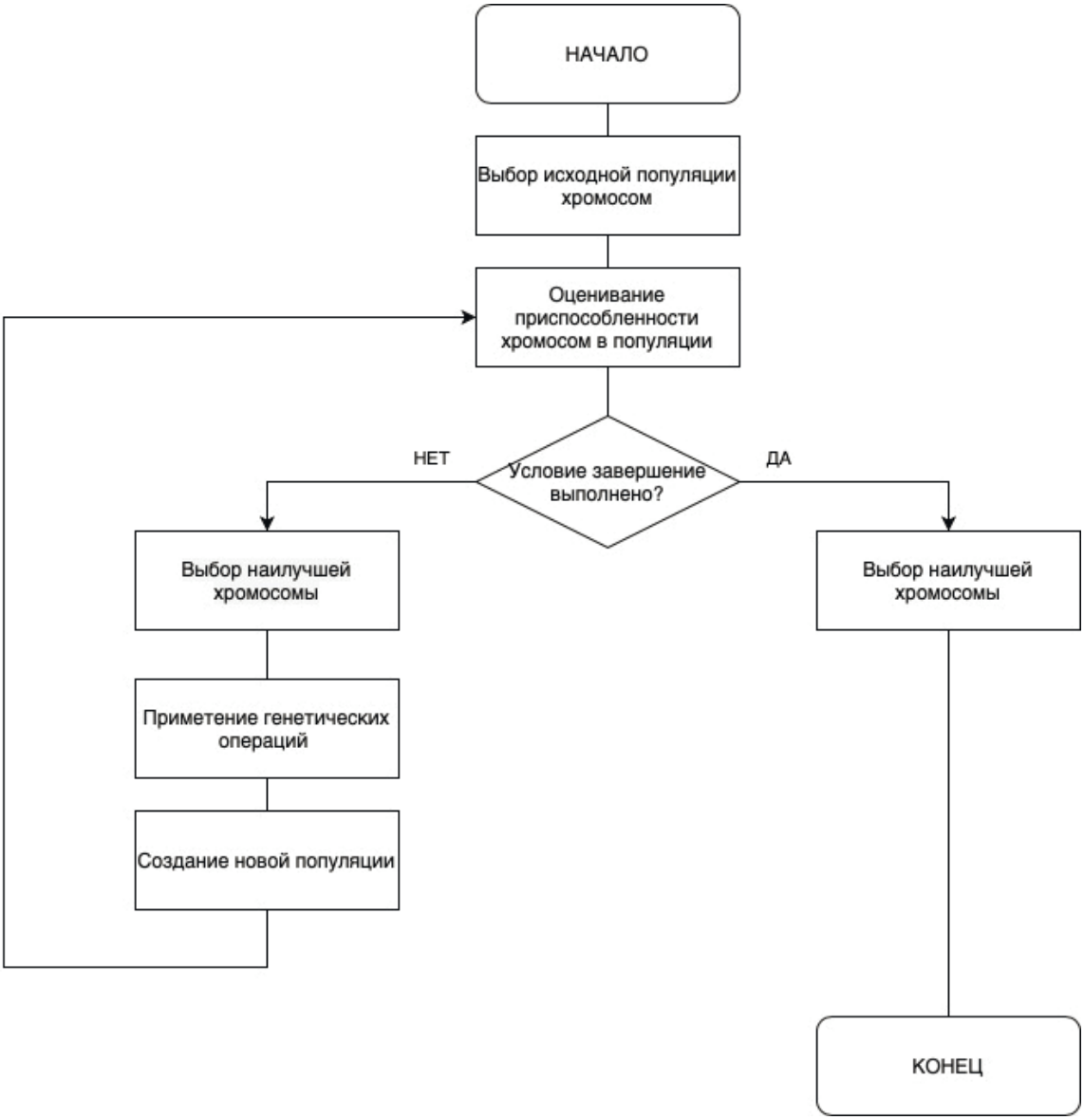

Рисунок 5 – Общая схема генетического алгоритма

Генетический алгоритм универсален, так как только функция приспособленности и кодирование решений зависят от условий поставленной задачи. При выполнении алгоритма учитываются следующие правила: начальная популяция выбирается случайно, количество ее особей остается неизменным, каждая из них записывается как строка с определенной длиной кодировки.

Каждый шаг алгоритма можно разбить на три этапа:

1) пропорциональный в зависимости от приспособленности отбор особей текущего поколения, имеющих право производить потомство, и формирование из них промежуточной популяции;

2) промежуточную популяцию делят на пару и с какой-то вероятностью скрещивают, в итоге в новое поколение попадает сама пара или ее потомки при их присутствии. Потомки формируются из отсеченных частей родительских строк, разделенной некой точкой.

3) происходит мутация полученного поколения, не допускающая преждевременной сходимости. Каждый бит особи с вероятностью не более 1% записывается как противоположный исходному.

В качестве заранее заданных критериев получения оптимального решениях могут быть определенное число смены поколений или схождения популяции – условие выполняется, если все строки являются почти идентичными и находятся в области экстремума. Решением алгоритма будет особь с наибольшим значением функции приспособленности.

При использовании генетического алгоритма для решения задачи поиска оптимального пути выполняются все вышеперечисленные шаги, приспособленность особи фактически служит мерой длины маршрута. Существуют несколько различных реализаций классического генетического алгоритма в зависимости от подхода к этапам скрещивания и мутации. Некоторые из них дали хорошие показатели при тестировании, сильно превзойдя алгоритм Кернигана-Лина. Но также, как и у алгоритма поиска с запретами трудоемкость процесса создает проблему при достаточно большом количестве узлов [19]. Исследование, представленное в статье “Comparison of Algorithms for Solving Traveling Salesman Problem” [25] показывает результаты сравнения трех методов: алгоритма ближайшего соседа, генетического и жадного алгоритмов. По итогам теста генетический алгоритм показал себя как самый

эффективный метод решения TSP с небольшим объемом данных, но в то же время как самый трудоемкий.

Суммируя сказанное выше, можно сказать, что главным недостатком генетического алгоритма является отсутствие гарантии, что полученное решение является оптимальный, как и само его нахождение за приемлемое оптимально время.

В то же время генетические алгоритмы показывает высокую эффективность в задачах с небольшим количеством городов. В ситуациях, когда в задачах большой размерности отсутствует упорядоченность вводных данных, генетический алгоритм является единственной альтернативой алгоритму полного перебора. Главное его преимущество – его можно использовать для решения сложных неформализованных проблем, для которых не существует частных методов решения, что позволяет ему эффективно решать нестандартные задачи.

По результатам данного сравнения было выявлено, что для задач с небольшим количеством меток, к которым относится практическая задача дипломной работы, одним из наиболее эффективных является генетический алгоритм, который и был выбран для дальнейшей работы.

Ниже представлен краткий алгоритм работы Swift кода для непосредственного поиска оптимального маршрута на карте, представленной во второй секции:

1. Инициализируем Apple Maps с определенными параметрами.
2. В результате взаимодействия считываем координаты точек и инициализируем начальную популяцию из заданных узлов.
3. Оцениваем каждую особь данной популяции с учетом приспособленности хромосом (длины маршрута).
4. Выбираем особей с наилучшей хромосомой.
5. Скрещиваем выбранные особи.
6. Добавляем полученную особь в популяцию.
7. Повторяем предыдущие шаги, пока количество поколений не превысит 50.

8. Готовый маршрут (особь с наилучшей хромосомой) добавляем на карту.

3.5 Описание данных

Для функционирования приложения необходимо создание базы данных. Она отвечает за хранение учетных записей, а также информацию о маршрутах и метках.

В таблице Route хранится информация о гостевых маршрутах города. Таблица User хранит информацию о пользователях, такую как логин и хешированный пароль. Таблица Massage предназначена для хранения сообщений пользователей о достопримечательностях. В таблице Location находится информация о маркерах на карте. Таблица Photo хранит фотографии достопримечательностей. Reviews необходима для хранения отзывов пользователей о маршрутах.

Скриншот схемы базы данных представлен на рисунке 6.

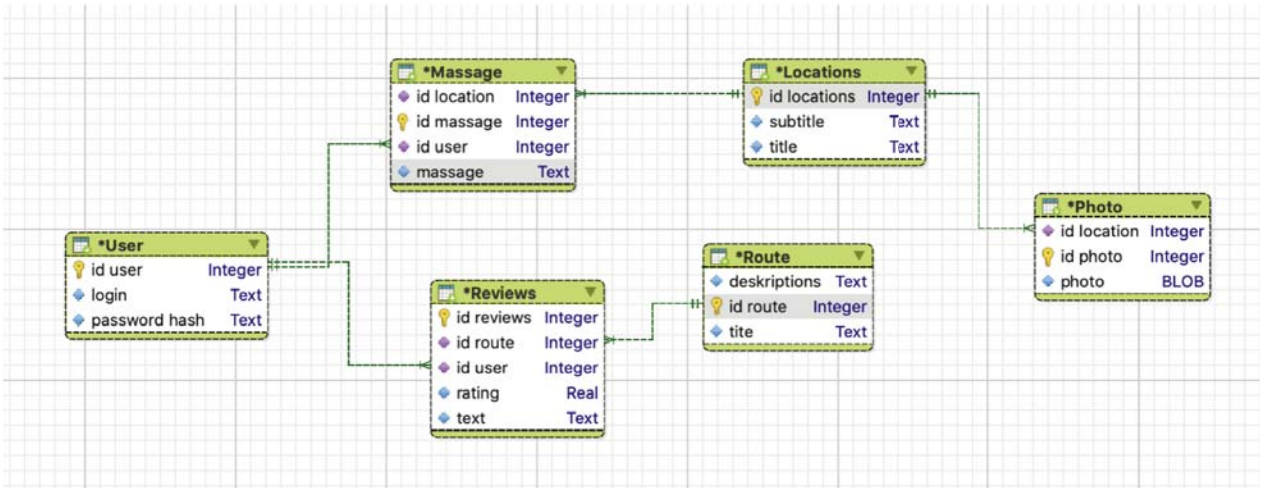

Рисунок 6 – Схема базы данных

Выводы по разделу три

В данной главе было произведено проектирование мобильного приложения. Подобраны необходимые технологии, такие как операционная система, среда разработки и язык программирования.

Следующим шагом было проектирование архитектуры приложения. Были описаны основные функции и методы. Представлена схема работы приложения.

После чего были представлены описания основных алгоритмов для реализации работы приложения. Были рассмотрены вопросы работы с картами. В частности: отображения местоположения, отображение маркера достопримечательности, построения маршрута. Также были разобраны алгоритмы использования голосового помощника для функции аудиогида.

Для улучшения построенных маршрутов был проведен анализ алгоритмов оптимизации. В ходе анализа были рассмотрены точные, неточные, а также генетический алгоритмы. В итоге для реализации задачи, рассмотренной в выпускной квалификационной работе, был выбран генетический алгоритм.

Последним шагом проектирования было описание данных необходимых для работы аудиогида.

4 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

4.1 Реализация интерфейса

При запуске приложения, открывается стартовое окно (рисунок 7). На данном окне расположены поля, которые необходимо заполнить для входы в приложения: логин и пароль. Если пользователь еще не зарегистрирован, то ему предлагается создать учетную запись.

Рисунок 7 – Стартовое окно приложения

При выборе пункта «Зарегистрироваться», вызывается ViewController, содержащий форму для регистрации (рисунок 8).

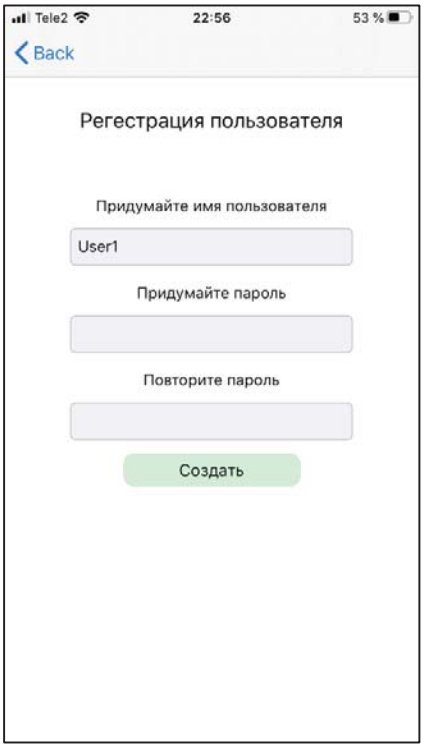

Рисунок 8 – Окно регистрации

В открытом окне пользователю предлагается ввести данные для создания учетной записи. Для ввода пароля у поля включено свойство «Secure text entry», что позволяет скрыть вводимые символы, на скриншоте данные поля отображаются как пустые.

После создания учетной записи открывается окно с меню приложения (рисунок 9).

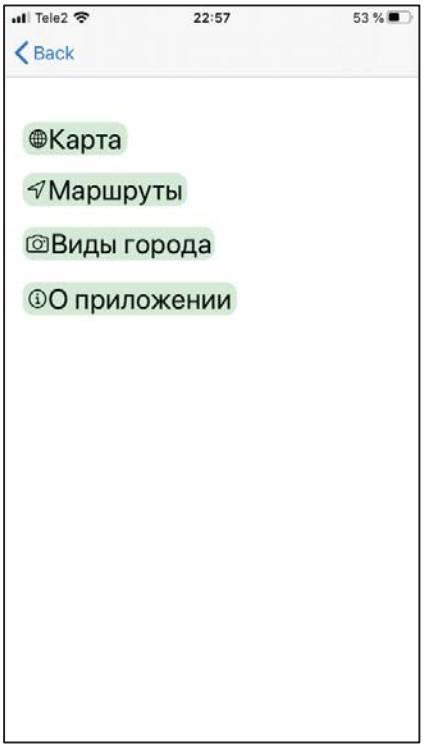

Рисунок 9 – Окно «Меню»

При выборе каждого из пунктов откроется функциональное окно приложения.

На рисунке 10 представлен ViewController отображающий карту мобильного устройства.

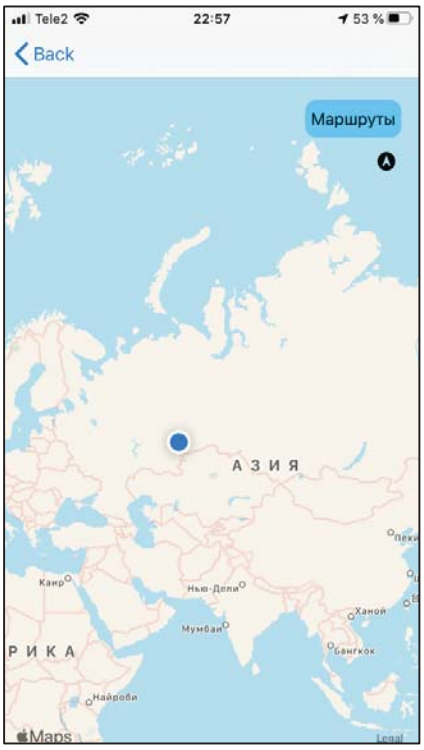

Рисунок 10 – Карта

В данном окне реализована кнопка перехода в раздел «Маршруты», расположенная в верхнем правом углу. На рисунке 11 представлен интерфейс данного окна.

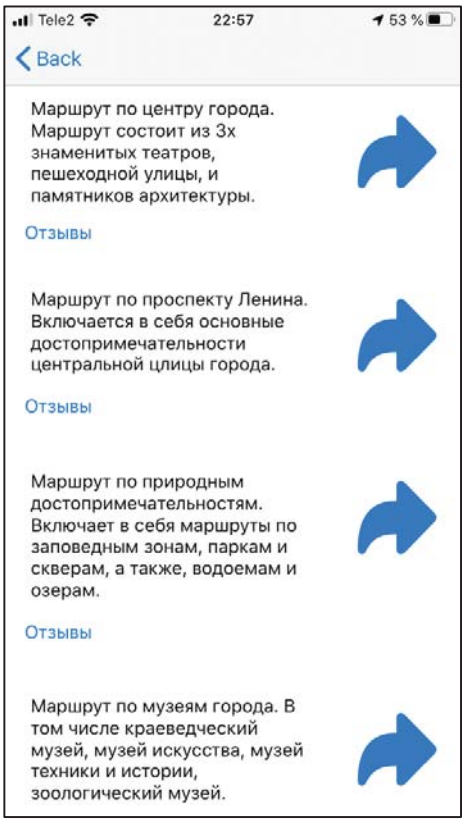

Рисунок 11 – «Маршруты»

У каждого из представленных маршрутов есть кнопка «Отзывы», нажав на которую можно прочесть мнения пользователей о данном маршруте, а также оставить собственный отзыв. Интерфейс данного функционала представлен на рисунке 12.

При нажатии на кнопку, расположенную справа от описания маршрута, пользователь откроет окно карты с построенным маршрутом (рисунок 13). На экране реализована кнопка «Start», нажав на которую запустится экскурсия.

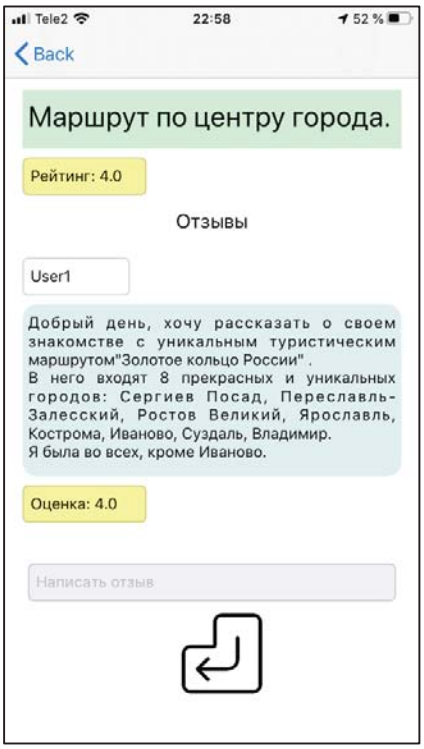

Рисунок 12 – «Отзывы»

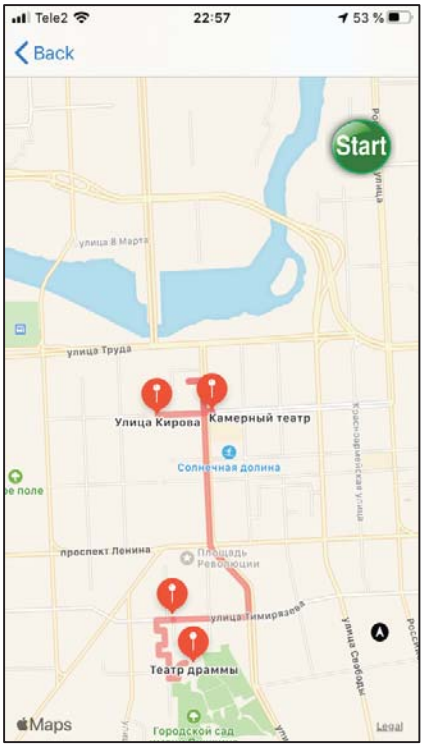

Рисунок 13 – «Маршрут»

При выборе пункта главного меню «Виды города», вызывается ViewController отображающий галерею фото, с подписью названия места. При нажатии на него открывается карта с установленной меткой. Интерфейс данного окна представлен на рисунке 14.

Рисунок 14 – «Виды города»

При нажатии на кнопку информации рядом с маркером, открывается окно с описанием места (рисунок 15). На данном экране можно ознакомиться с информацией. А также на нем есть две кнопки. Первая кнопка – галерея, куда пользователи могут загружать фото достопримечательности. Вторая – сообщения, где пользователи делятся эмоциями и общаются (рисунок 16).

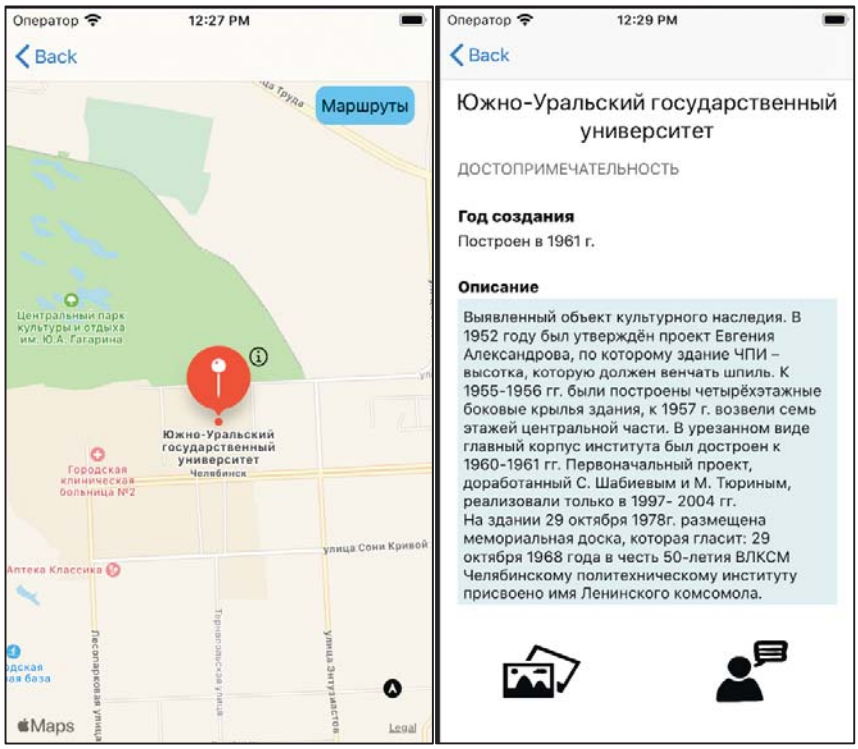

Рисунок 15 – «Информация о месте»

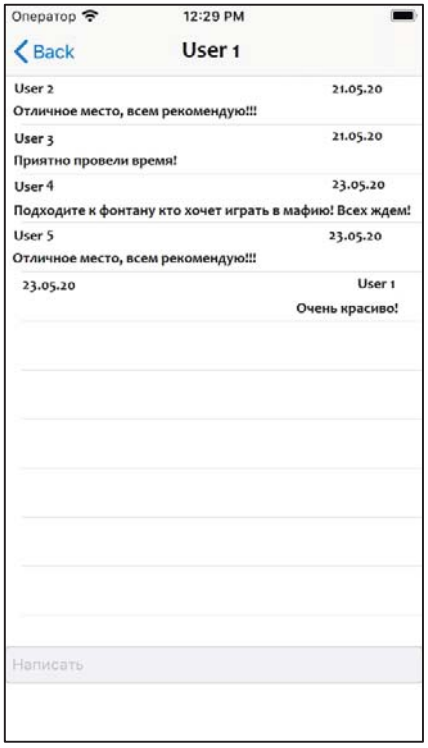

Рисунок 16 – «Сообщения»

4.2 Реализация логики приложения

Логика приложения написана на языке программирования Swift.

Приложение состоит из семи рабочих окон: вход, регистрация, меню, карта, маршруты, виды города, отзывы. Для каждого окна написан код на языке Swift. Каждое окно представляет собой новый класс, содержащий программный код.

1. Реализация входа и регистрации.

Для входа в приложение пользователю необходимо ввести логин и пароль. При вводе данных в поля на экране, данные считываются в классе EnterViewController. После чего отправляется Get запрос на сервер для получения ответа о существовании пользователя. При положительном ответе открывается окно меню. Если пользователя с такими данными не существует, то клиенту сообщится об этом, а также продолжится завести учетную запись.

При регистрации пользователь вводит собственные данные и на сервер отправляется Put запрос, для добавления пользователя. Логин пользователя проверяется на уникальность. Если логин уникален, то пользователю открывается окно меню, в противном случает пользователю будет предложено поменять логин.

2. Реализация методов работы с картой.

Для реализации работы карты была использована библиотека MapKit. В классе окна создается элемент MapView, который необходим для отображения карты. Для отображения местоположения пользователя реализована функция setupManager(). Для реализации данной функции был создан делегат класса Map CLLocationManagerDelegate. Данный делегат предоставляет функционал для работы с обновляющейся геопозицией. В теле делегата вызывается функция locationManager, которая позволяет получать обновляемые данные. После чего создается регион отображения карты с маркером местоположения. Реализация представлена в листинге 1.

Листинг1 – Получение местоположения

```
   func       locationManager(_       manager:       CLLocationManager,
didUpdateLocations locations: [CLLocation]) {

          if let location = locations.last?.coordinate{
              let   region   =   MKCoordinateRegion(center:   location,
latitudinalMeters: 1000, longitudinalMeters: 1000)
               map.setRegion(region, animated: true)
        }
           manager.stopUpdatingLocation()
      }
```

Для проверки на доступ к геопозии реализована функция checkAutorization(). В данной функции происходит проверка на наличие прав на доступ к местоположению. При разрешенном доступе местоположение отображается автоматически. При закрытом доступе пользователю предлагается перейти в настройки и разрешить доступ. Если в доступе отказано, то функция определения местоположения не будет работать. Реализация функции представлена в листинге 2.

Листинг 2 – Проверка на доступ к геопозиции

```
func checkAutorization(){
        switch  CLLocationManager.authorizationStatus() {
        case .authorizedAlways:
            break
        case .authorizedWhenInUse:
            map.showsUserLocation = true
            locationManager.startUpdatingLocation()
            break
        case .denied:
            break
        case .restricted:
            break
        case .notDetermined:
            locationManager.requestWhenInUseAuthorization()
            break
        }
```

Для построения маршрутов была использована библиотека MapKit и CoreLocation. Для построения маршрутов реализована функция rout(). Для

реализации данной функции был создан делегат класса Map MKMapViewDelegate. Данный делегат предоставляет функционал рисования маршрутов на верхнем слое карты. Для создания маршрута в функцию rout подаются входные значения типа MKPointAnnotation. Входные данные имеют поля координаты и описания объекта. С помощью функции MKPlacemark получаем координаты точек. После чего создается запрос на создание маршрута. Задается тип транспорта и калькулируется оптимальный маршрут. После чего создается регион данного маршрута для отображения на экране пользователя. Реализация представлена в листинге 3.

Листинг 3 – Построение маршрута

```
   func rout (firstAnotation: MKPointAnnotation, secondAnotation: MKPointAnnotation){

            mapView.showAnnotations([firstAnotation, secondAnotation], animated: true)

            let firstCoordinate = firstAnotation.coordinate
            let secondCoordinate = secondAnotation.coordinate

            let firstItem = MKMapItem(placemark: MKPlacemark(coordinate: firstCoordinate))
            let secondItem = MKMapItem(placemark: MKPlacemark(coordinate: secondCoordinate))

            let request = MKDirections.Request()
            request.source = firstItem
            request.destination = secondItem
            request.transportType = .walking

            let directions = MKDirections(request: request)
            directions.calculate { (response, error) in
                guard let response = response else {
                    NSLog("Error of response : \(error.debugDescription)")
                    return
            }
```

Используя встроенную функцию делегата MKMapViewDelegate, изменяем цвет толщину и уровень линии маршрута. Функция представлена в листинге 4.

Листинг 4 – Настройка линий маршрута

```
   func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) -
> MKOverlayRenderer {
           let renderer = MKPolylineRenderer(overlay: overlay)
           renderer.lineWidth = 6
           renderer.strokeColor   =   UIColor(red:   0xff/0xff,   green:
0x77/0xff, blue: 0x77/0xff, alpha: 0.8)
           return renderer  }
```

Для оптимизации маршрута используется генетический алгоритм. Нахождение наилучшего маршрута происходит следующим образом:

1. Загрузить в список markList координаты всех точек маршрута Номер достопримечательности – это индекс списка markList.

2. Найти и запомнить в списке Distanc для каждой точки расстояния до всех прочих точек.

3. Найти и запомнить в списке Neighborдля каждой метки номера ее меток соседей, то есть наиболее близко расположенных к ней меток.

4. Сформировать начальную популяцию – список маршрутов, создаваемых на основе жадного алгоритма.

5. Рассчитать для каждого маршрута его длину.

6. Упорядочить маршруты по возрастанию длины пути.

7. Взять два лучших маршрута в качестве родительских и сформировать, используя скрещивание и мутацию, дочерний маршрут - childroute.

8. Заменить в популяции худший маршрут на маршрут childroute.

9. Вычислить длину маршрута childroute и если она меньше длины лучшего маршрута, то признать childroute в качестве лучшего маршрута.

10. Перейти к пункту 7 если условие завершения не выполнено, иначе завершить вычисления.

3. Реализация методов использования голосового помощника.

Для использования функций голосового помощника была использована библиотека AVFoundation. Для воспроизведения текста создается объект класса AVSpeechUtterance. В данный объект передаем текст, который необходимо воспроизвести, а также необходимые параметры. Параметры содержат язык воспроизведения, скорость, интервал времени и тд. После чего создается объект класса AVSpeechSynthesizer. Данный класс необходим для воспроизведения синтезированной речи. На вход подается объект класса AVSpeechUtterance, содержащий необходимые параметры. Реализация представлена в листинге 5.

Листинг 5 – Воспроизведение синтезированной речи

```
@IBAction func Voice(_ sender: Any) {
        let utterance = AVSpeechUtterance(string: text1)
        utterance.voice = AVSpeechSynthesisVoice(language: "ru-RU")
        utterance.rate = 0.4
   let synthesire = AVSpeechSynthesizer()
        synthesire.speak(utterance)

    }
```

Выводы по разделу четыре

В главе 4 была описана реализация мобильного приложения. Были приведены листинги методов для реализации функционала приложения. Были описаны библиотеки, которые использовались в проекте. Также был представлен интерфейс приложения, с описанием всех функциональных окон.

5 ТЕСТИРОВАНИЕ

5.1 Тестирование входа и регистрации

При вводе неверных данных клиентом появляется оповещение с ошибкой. На корректную работу данного аспекта были проведены тесты. В таблице 2 описано тестирование входа и регистрации.

Таблица 2 – Тестирование входа и регистрации

Название теста	Входные данные	Ожидаемый результат	Полученный результат	Итоги теста
Неверные данные	Клиент ввел неверный логин или пароль	Появится окно с ошибкой «Неверный логин или пароль»	Появилось окно с ошибкой «Неверный логин или пароль»	Тест пройден успешно
Уникальное имя	При регистрации пользователь ввел уже существующее имя	Появится окно с ошибкой «Пользователь с таким именем уже существует»	Появилось окно с ошибкой «Пользователь с таким именем уже существует»	Тест пройден успешно
Подтверждение пароля	При регистрации пользователь ввел неверный пароль при подтверждении	Появится окно с ошибкой «Введённые пароли не совпадают»	Появилось окно с ошибкой «Введённые пароли не совпадают»	Тест пройден успешно

Результаты тестов представлены на рисунках. «Неверные данные» – рисунок 14, «Уникальное имя» – рисунок 15, «Подтверждение пароля» – рисунок 16.

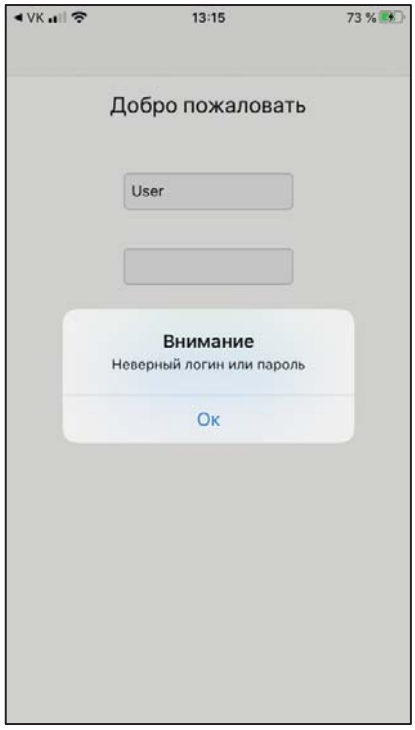

Рисунок 14 – Тест «Неверные данные»

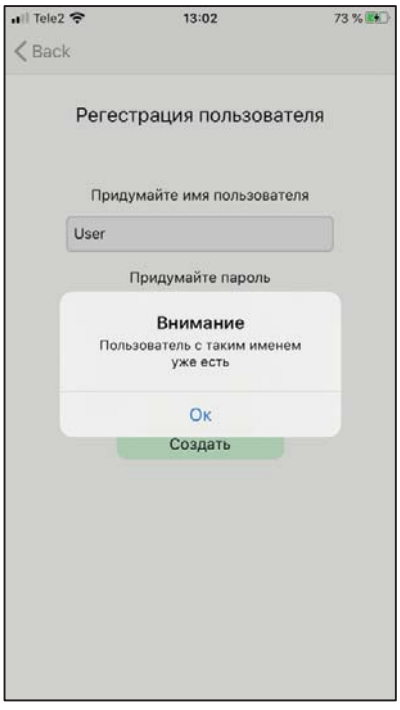

Рисунок 15 – Тест «Уникальное имя»

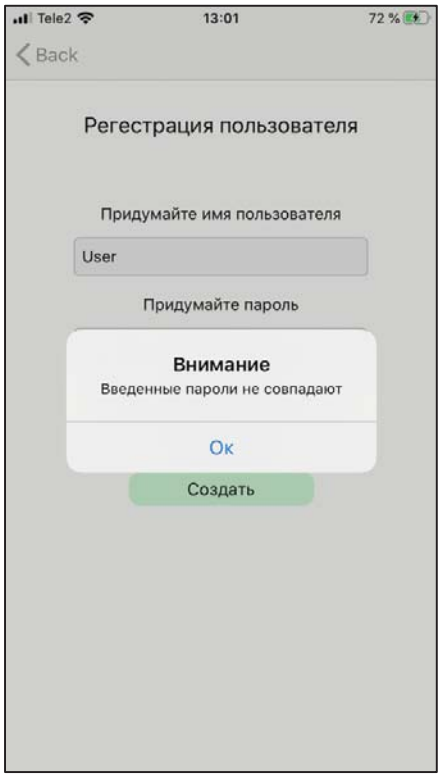

Рисунок 16 – Тест «Подтверждение пароля»

5.2 Тестирование работы карты

При выборе одного из маршрутов пользователю должна открыться карта с метками, и построенным по ним маршрутом. При нажатии на кнопку определения местоположения, пользователю должен открыться регион карты, на котором будет отображен маркер местоположения клиента.

Тестирование работы карты представлено в таблице 3.

Таблица 3 – Тестирование работы карты

Название теста	Входные данные	Ожидаемый результат	Полученный результат	Итоги теста
Определение местоположения	Пользователь нажал на кнопку определения местоположения	Покажется регион карты с маркером местоположения пользователя.	Покажется регион карты с маркером местоположения пользователя.	Тест пройден успешно
Построение маршрута	Пользователь выбрал маршрут	Откроется карта с регионом на котором будет построен маршрут	Открылась карта с регионом на котором построен маршрут	Тест пройден успешно

Результаты тестов представлены на рисунках. «Определение местоположения» – рисунок 17, «Построение маршрута» - рисунок 18.

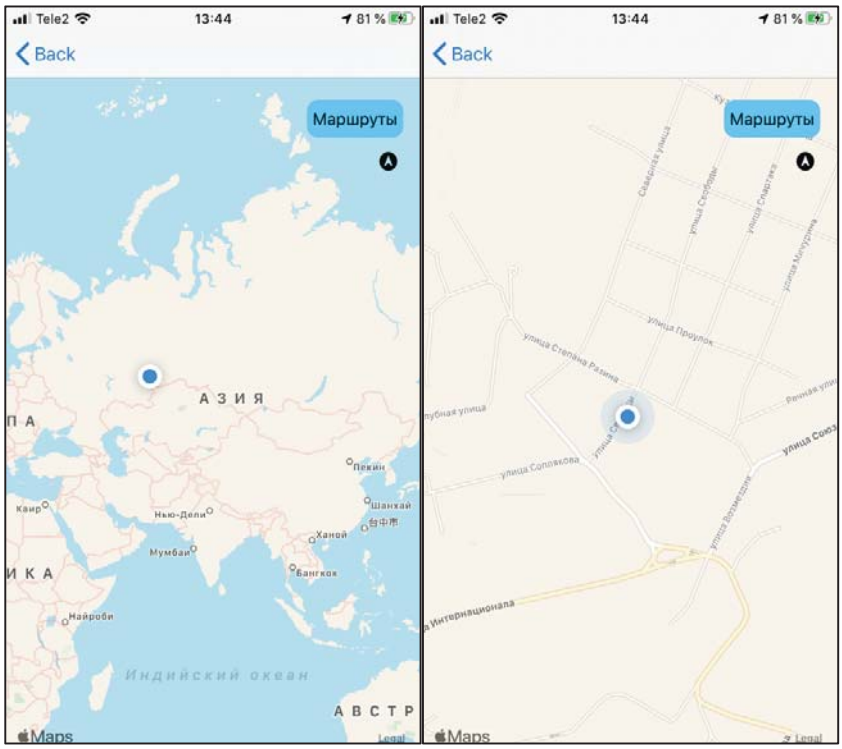

Рисунок 17 – Тест «Определение местоположения»

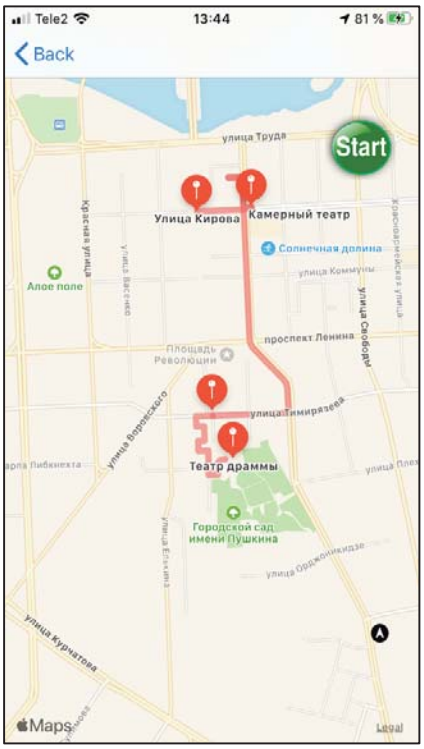

Рисунок 18 – Тест «Построение маршрута»

Выводы по разделу пять

В данной главе были проведены тесты приложения на обнаружение ошибок. Были проведены итоговые таблицы тестов, а также продемонстрированы скриншоты итоговых результатов.

ЗАКЛЮЧЕНИЕ

– В представленной работе были разобраны все основные темы, связанные с реализацией мобильного приложения. Проведен анализ предметной области, в том числе: проведен обзор аналогов, представлена актуальность данной разработки. Определены основные требования к системе. Подобраны необходимые технологии для реализации приложения, в том числе: операционная система, среда разработки и язык программирования. Разработано мобильное приложение, в котором реализован метод работы с картами. При помощи генетического алгоритма была проведена оптимизация маршрутов. Так же реализован метод использования голосового помощника, который необходим для воспроизведения экскурсий на городских туристических маршрутах. Проведены тесты на работоспособность приложения.

В ходе выполнения данной работы были решены следующие задачи:

- задача анализа предметной области;
- задача определения требований к системе;
- задача проектирования приложения;
- задача реализации приложения;
- задача тестирования.

– .

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Мобильный аудиогид. – https://izi.travel/ru. Дата обращения: 22.02.2020

2. Информационный сайт о технологиях. – https://keddr.com. Дата обращения: 22.02.2020

3.Developer Apple – https://developer.apple.com/documentation/mapkit/mkmapview. Дата обращения: 22.02.2020

4. iOS & Swift Tutorials – https://www.raywenderlich.com/548-mapkit-tutorial-getting-started&. Дата обращения: 24.02.2020

5. SwitfBook – https://swiftbook.ru/post/tutorials/risuem-marshrut-pri-pomoshchi-mapkit. Дата обращения: 25.02.2020

6. Грей, Э. Swift. Карманный справочник / Э. Грэй. – М.: Изд-во Вильямс, 2016. – 224 с.

7. Усов, В.Н. Swift. Основы разработки приложений под iOS или macOS / В.Н. Усов. – СПб.: Питер, 2018. – 448 с.

8. Маскри, М. Разработка приложений в среде Xcode для iPhone или iPad с использованием iOS SDK / М. Маскри. – М.: Диалектика, 2018. – 896 с.

9. Филер, Д. Swift For Dummies / Д. Филер, С.Ламайти . – СПб.: Питер, 2018, – 655 с.

10. Swiftbook. - https://swiftbook.ru. Дата обращения 22.02.2020

11. Applegate D.L., Bixby R.E., Chvátal V., Cook W.J. The Traveling Salesman Problem. Princeton University Press, 2007. P. 44–52.

12. Левитин А. В. Алгоритмы. Введение в разработку и анализ / А.В. Левитин. – М.: Вильямс, 2006. – 160 с.

13. Гэри М. Вычислительные машины и труднорешаемые задачи / М. Гэри, Д. Джонсон. – М.: Мир, 1982. – 228 с.

14. Okano H Study of Practical Solutions for Combinatorial Optimization Problems. Tohoku University, 2009. P. 14–17.

15. Захарова, Е.М. Обзор методов многомерной оптимизации / Е.М. Захарова. – М.: Вильямс, 2014. – 266 с.

16. Kona H., Burde A., Dr. Zanwar D. R. A Review of Traveling Salesman Problem with Time Window Constraint // IJIRST – International Journal for Innovative Research in Science & Technology, 2015. Vol. 2, Issue 1. P. 253– 254.

17. Gutin G.,Yeo A. The Greedy Algorithm for the Symmetric TSP. University of London, 2002. P. 1–2.

18. Nilsson C. Heuristics for the Traveling Salesman Problem. Linkoping University, 2011. P. 1–6.

19. Gutin G., Karapetyan D. Lin-Kernighan Heuristic Adaptations for the Generalized Traveling Salesman Problem. Royal Holloway London University, 2010. P. 1–5.

20. Gupta R., Chauhan C., Pathak K. Survey of Methods of Solving TSP along with its Implementation using Dynamic Programming Approach. // International Journal of Computer Applications, 2012. Vol. 52, No.4. P. 1–6.

21. Basu S. Tabu Search Implementation on Traveling Salesman Problem and Its Variations: A Literature Survey. Indian Institute of Management Calcutta, 2012. P. 1–8.

22. Gupta R., Chauhan C., Pathak K. Survey of Methods of Solving TSP along with its Implementation using Dynamic Programming Approach. // International Journal of Computer Applications, 2012. Vol. 52, No.4. P. 1–6.

23. Dorigo M., Caro G. D. Ant Algorithms for Discrete Optimization // Artificial Life, 1999. Vol. 5, No 2. P. 139–140.

24. Dorigo M., Gambardella L. M. Ant colonies for the traveling salesman problem. Université Libre de Bruxelles, 1996. P. 1–4.

25. Abdulkarim H.A., Alshammari I. F. Comparison of Algorithms for Solving Traveling Salesman Problem. // International Journal of Engineering and Advanced Technology, 2015. Vol.4, Issue 6. P. 76–79.