МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Институт экономики и менеджмента
Факультет экономики и финансов
Кафедра экономики

Допускаю к защите:
И.о. зав. кафедрой ЭК Братухина Е.А.
(название кафедры, ФИО зав. кафедрой)

_____ 23.06. 2020
(подпись) *(дата)*

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

РАЗРАБОТКА РЕКОМЕНДАТЕЛЬНОГО СЕРВИСА ПОСТРОЕНИЯ ПЕРСОНАЛЬНЫХ ТУРИСТИЧЕСКИХ МАРШРУТОВ

Направление подготовки 38.03.05 Бизнес-информатика
Профиль (направленность) «Архитектура предприятия»

Разработала
Обучающийся гр. БИб-4601-01-00 _____ / Бокарева Д.Г. / 19.06.2020
(шифр группы) *(подпись)* *(Ф.И.О.)* *(дата)*

Руководитель
к.ф.-м.н., доцент каф. ЭК _____ Караулов В.М. 20.06.2020
(звание, должность) *(подпись)* *(Ф.И.О.)* *(дата)*

Киров 2020 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт экономики и менеджмента

Факультет экономики и финансов

Кафедра экономики

Утверждаю:
И.о. зав. кафедрой ЭК Братухина Е.А.
(название кафедры, И.О. Фамилия)

_____ 20.04 2020
(подпись) *(дата)*

ЗАДАНИЕ

на выполнение выпускной квалификационной работы обучающегося группы БИб-4601-01-00
(шифр группы)

Бокаревой Дарьи Геннадьевны
(Фамилия Имя Отчество)

1. Тема работы:
Разработка сервиса построения персональных туристических маршрутов

(*Утверждена приказом по университету от* 19.03.2020 № 13-02/35)

2. Срок сдачи обучающимся заключительной работы: "19" июня 2020 г.

3. Исходные данные для выполнения выпускной квалификационной работы:
Образовательная литература, научная литература, законодательная и нормативно-методическая база

4. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов)
- Анализ деятельности и сервисов организации
- Исследование рынка картографических сервисов
- Разработка технического задания на разработку сервиса
- Описание математических основ работы сервисов построения маршрутов
- Проектирование и разработка сервиса
- Экономическое обоснование разработки сервиса

5. Перечень презентационного материала:
таблицы, диаграммы, рисунки, презентация

6. Календарный график выполнения выпускной квалификационной работы:

Наименование этапа работ	*Дата выполнения*
1. Характеристика и анализ проектируемого объекта и его основных технико-экономических показателей	20.04.2020 – 03.05.2020
2. Разработка основных направлений проектной части, способов внедрения	04.05.2020-17.05.2020
3. Описание практической реализации задачи с привязкой к конкретному объекту	18.05.2020-10.06.2020
4. Подготовка доклада и презентационного материала	11.06.2020-18.06.2019
5. Проверка ВКР в системе «Антиплагиат.Вуз»	19.06.2020
6. Предоставление работы на отзыв	21.06.2020
7. Предоставление работы на утверждение заведующему кафедрой	23.06.2020
8. Предоставление работы в ГЭК	03.07.2020

7. Руководитель работы:

к.ф.-м.н., доцент — *(звание, должность)*

Караулов В.М. — *(Фамилия Имя Отчество)*

Дата выдачи задания: **“20” апреля 2020 г.**

Руководитель _____

(подпись)

Задание принято к исполнению **“20” апреля 2020 г.**

Обучающийся _____

(подпись)

Реферат

Бокарева Д.Г. Разработка сервиса построения персональных туристических маршрутов – Выпускная квалификационная работа / ВятГУ, каф. ЭК; рук. к.ф.-м.н., доцент В.М. Караулов. – Киров, 2020. ВКР – 98 с., 5 таб., рис., 45 источников, приложений.

МЕТОДЫ ПОСТРОЕНИЯ МАРШРУТОВ, МУРАВЬИНЫЙ АЛГОРИТМ, ВЕБ-ПРИЛОЖЕНИЕ, PYTHON, DJANGO, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, FLUTTER, РЕКОМЕНДАТЕЛЬНЫЙ АЛГОРИТМ

Цель выпускной квалификационной работы – разработка рекомендательного сервиса построения персональных туристических маршрутов для КОГАУ «Центр развития туризма Кировской области» на основе исследования существующих методов построения оптимальных маршрутов.

Объект исследования – деятельность и сервисы Центра развития туризма Кировской области.

Предмет исследования – методы построения и визуализации маршрутов.

В результате исследования был изучен рынок картографических сервисов, исследованы методы построения маршрутов, предложен новый метод, позволяющий строить туристические маршруты с учётом предпочтений пользователей.

Спроектирован и разработан сервис, решающий проблемы организации, выявленные в ходе анализа её деятельности. Сервис основан на предложенном методе и рекомендательном алгоритме и представляет из себя мобильное приложение, бизнес-логика которого реализована в веб-приложении. Рассчитана экономическая эффективность разработки сервиса.

Abstract

Bokareva D.G. Development of a recommendation service for building personal tourist routes – Final Qualification Work / Vyatka State University, department of Economics; head Ph.D. Physics, Associate Professor V.M. Karaulov. – Kirov, 2020. FQW 98 pp., 5 tab., 36 fig., 48 sources, 4 appendices.

ROUTE-BUILDING METHODS, ANT COLONY OPTIMIZATION, WEB-APPLICATION, PYTHON, DJANGO, MOBILE APP, FLUTTER, RECOMMENDATION ALGORITHM

The purpose of the final qualification work is development of a recommendation service for building personal tourist routes for “Tourism Development Center of the Kirov Region” based on a study of existing route-building methods.

The object of study is the activities and services of the Tourism Development Center of the Kirov Region.

The subject of study is route-building methods and methods of visualization routes.

As a result of the study, the geospatial market was studied, route-building methods were studied, a new method was proposed, that allows building tourist routes taking into account the preferences of users.

A service has been developed that solves the organization's problems identified in the analysis of its activities. The service is based on the proposed method and the recommendation algorithm and is a mobile app, the business logic of which is implemented in the web-application. The economic efficiency of service development is calculated.

Определения, обозначения и сокращения

API – Application Programming Interface – программный интерфейс приложения

ER – Entity Relationship – сущность-связь

HTML – Hypertext Markup Language – язык гипертекстовой разметки

HTTP – Hypertext Transfer Protocol – Протокол передачи гипертекста

IDE – Integrated Development Environment – интегрированная среда разработки

JSON – JavaScript Object Notation – текстовый формат обмена данными

ORM – Object Relational Mapping – объектно-реляционное представление

SDK – Software Development Kit – набор средств разработки

SQL – Structured Query Language – язык структурированных запросов

T-SQL – Transact-SQL – процедурное расширение языка SQL

UML – Unified Modeling Language – унифицированный язык моделирования

АС – автоматизированная система

БД – база данных

ИС – информационная система

ОС – операционная система

ПК – персональный компьютер

ПО – программное обеспечение

ПЭВМ – персональная электронно-вычислительная машина

СУБД – система управления базами данных

ТЗ – Техническое задание

ЭВМ – электронно-вычислительная машина

Содержание

Введение

На сегодняшний день одной из самых актуальных задач в России является реабилитация и развитие туристической отрасли. Особое внимание уделяется развитию внутреннего туризма.

Кировская область является богатым историей регионом, привлекающим туристов. Обеспечение благоприятных условий для доступа к туристским ресурсам и комфортного пребывания туристов в регионе входит в ключевые задачи Центра развития туризма Кировской области.

Качество предоставляемых организацией услуг напрямую влияет на туристическую активность в регионе, объём въездного потока туристов, а также на доходы региона с туристической отрасли. Поэтому организация в первую очередь заинтересована в комфорте и удовлетворённости потребителей. Центр развития туризма проводит персональные консультации, мероприятия, выставки, фестивали и другие активности. Кроме того, организация разрабатывает собственные туристические маршруты по всей области, печатая их на красочных путеводителях-сувенирах. И на текущий момент – это единственный способ предоставить туристу маршрут. Однако если вместить пешеходный городской маршрут на путеводители вполне возможно, то с автомобильными возникает множество проблем.

Туристы не получают развёрнутую информацию о междугороднем маршруте на печатном носителе, возникает потребность в консультировании и растут трудозатраты организации. Центру развития туризма не хватает функционала текущих сервисов для визуализации маршрутов в интерактивном виде и предоставления возможности пользователям строить собственные туристические маршруты, используя информацию и ресурсы организации.

Таким образом, было принято решение о необходимости внедрения дополнительного сервиса в работу организации, который будет способен решить существующие проблемы и направлен на персонализацию предоставляемых услуг.

Целью выпускной квалификационной работы является разработка рекомендательного сервиса построения персональных туристических маршрутов.

Для достижения поставленной цели решаются следующие задачи:

- Анализ деятельности и существующих сервисов Центра развития туризма Кировской области;
- Исследование рынка картографических сервисов и поиск решения;
- Составление технического задания на разработку сервиса;
- Описание математических основ работы сервиса;
- Проектирование и разработка сервиса;
- Экономическое обоснование разработки сервиса.

Объект исследования – деятельность и сервисы Центра развития туризма Кировской области.

Предмет исследования – методы построения и визуализации маршрутов.

Теоретическую и методическую базу составили методы и подходы отечественных и зарубежных авторов к построению маршрутов и разработке программных продуктов.

1. Анализ предметной области и постановка задачи

1.1. Анализ деятельности Центра развития туризма Кировской области

Кировское областное государственное автономное учреждение "Центр развития туризма Кировской области" осуществляет свою деятельность на территории Кировской области с 2004 года.

Основной целью Центра развития туризма Кировской области является создание благоприятных условий для доступа граждан Российской Федерации и иностранных граждан к туристским ресурсам Кировской области, а также развитие внутреннего и въездного туризма на территории Кировской области [1].

Учреждение оказывает следующие услуги [2]:

- предоставление информации о туристических маршрутах по Кировской области;
- организация участия Кировской области в межрегиональных и международных туристских выставках, размещение информации о туристской привлекательности региона на информационных ресурсах;
- организация и проведение туристских событийных мероприятий на территории Кировской области (фестивалей, праздников, акций), семинаров, конференций, форумов, круглых столов по проблемам развития туризма;
- подготовка и распространение информационных материалов о туристской привлекательности Кировской области (туристские карты, буклеты и т.д.);
- индивидуальное консультирование туристов;
- реализация турпакетов туристско-информационного центра (ТИЦ) и операторов внутреннего и въездного туризма, продажа экскурсий, реализация билетов на мероприятия, продажа авиа и ж/д билетов.

Для оказания услуг и распространения информации и продукции, Центр развития туризма использует следующие каналы коммуникации (рис. 1):

- личные встречи;
- распространение печатной продукции на различных мероприятиях;
- сайт;
- социальные сети (VK, Facebook, Instagram);
- средства массовой информации (СМИ).

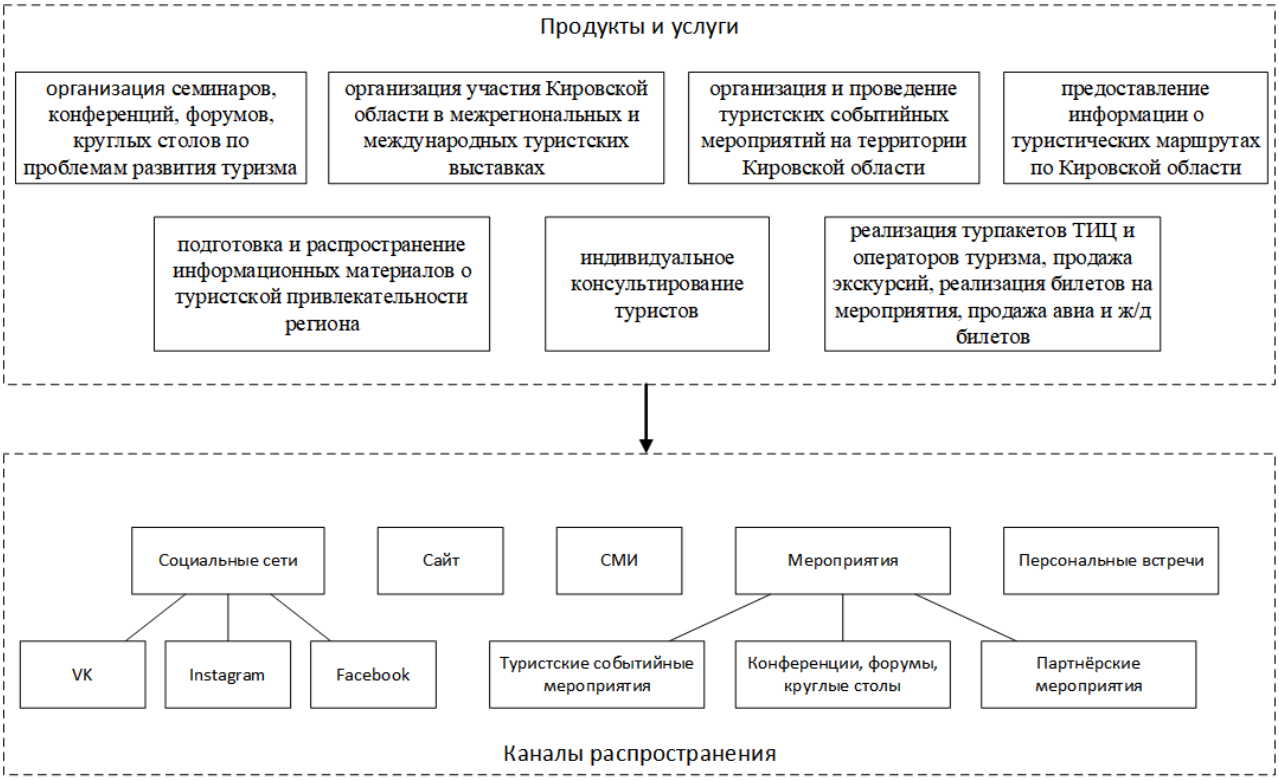
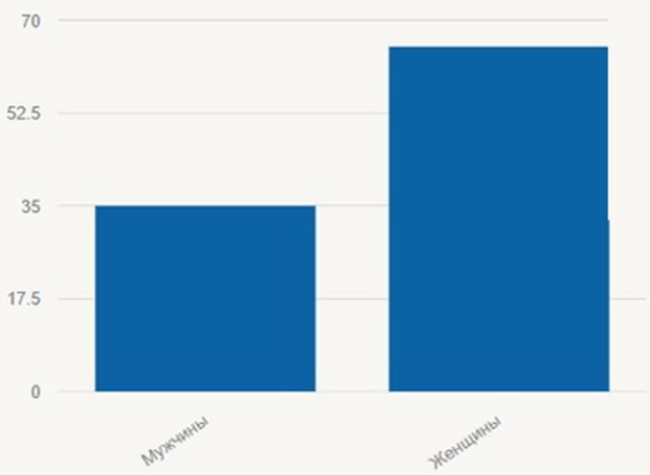
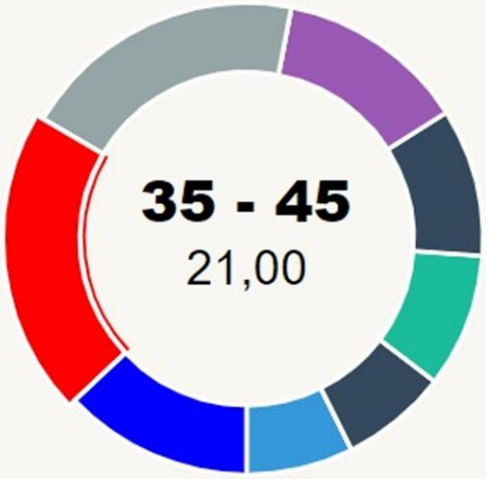


Рисунок 1 – Продукты, услуги и каналы их распространения

В ходе анализа целевой аудитории сервисов составлен портрет потребителя. Потребителями организации являются преимущественно женщины возраста 24-45 лет, проживающие в городе Кирове (рис. 2).



Пол	%
Мужчины	35,0%
Женщины	65,0%



Возраст	%
0-18	7,0
18-21	7,0
21-24	9,0
24-27	10,0
27-30	13,0
30-35	20,0
35-45	21,0
45+	13,0

Город	%
Киров	72,0
Москва	8,0
Санкт-Петербург	8,0
Кирово-Чепецк	3,0
Екатеринбург	2,0
Слободской	2,0
Казань	2,0
Пермь	2,0
Нижний Новгород	1,0
Котельнич	1,0

Рисунок 2 – Анализ целевой аудитории сервисов организации

Организация столкнулась с проблемой обеспечения потребителей развёрнутой информацией о туристических маршрутах. На данный момент, маршруты частично печатаются на бумажных путеводителях. У потребителей возникают проблемы со следованием по маршруту, а также с необходимостью самостоятельного ручного ввода маршрута в электронных картах.

Нынешние сервисы организации, с помощью которых она оказывает услуги информирования – сайт, сообщества и страницы в социальных сетях– не способны решить данную проблему.

Учитывая все факторы, организация делает вывод о необходимости использования нового сервиса, обеспечивающего визуализацию туристических маршрутов на карте, а также возможность построения персональных туристических маршрутов. Сервисы, обеспечивающие возможность использования интерактивных карт, именуются картографическими.

1.2. Исследование рынка картографических сервисов

По оценке аналитического агентства MarketsandMarkets [3], в 2018 году мировой объём геолокационного рынка, включающий картографические сервисы, составил 40,65 миллиардов долларов. Уже к 2023 году прогнозируется двукратный прирост объёма. Исследование данного рынка направлено на поиск решений, удовлетворяющих потребностям организации.

Картографические сервисы позволяют:

- отображать интерактивную карту по нужным географическим координатам местности;
- производить поиск мест по определенным параметрам: адрес, название, категория или определённая позиция объекта относительно иных объектов;
- строить оптимальные маршруты между точками на карте, с сопутствующими дополнительными параметрами: учётом данных о состоянии трафика на дорогах следования, пробках, односторонних движениях и прочих деталей конкретной территории местности.

В России наиболее популярными и универсальными сервисами являются Google Maps, Яндекс.Карты и 2ГИС. Сервисы компаний Google и Яндекс содержат картографическую информацию множества городов и стран по всему миру, тогда как 2ГИС направлен на предоставление подробной карты городов России. Все они позволяют выполнять классические для картографических сервисов функции (построение маршрутов, поиск объектов и др.). Google Maps позволяет пользователям сохранять маршруты и передавать их между устройствами. Яндекс.Карты также способны сохранять маршруты с помощью сервиса Конструктор Карт. 2ГИС такой возможностью не обладает и в большей степени направлен на построение маршрутов внутри населённого пункта. Возможность построения междугороднего маршрута существует только для передвижения на личном транспорте.

Наибольшую актуальность интерактивные карты представляют для логистического бизнеса и сферы туризма. Поэтому, кроме универсальных картографических сервисов, растёт количество решений, направленных на определённую индустрию. В туристской индустрии появляются сервисы, позволяющие планировать путешествия, приложения-путеводители, включающие в себя туристические маршруты и аудиогиды, а также другие решения. В России известны такие сервисы, как Google Trips, izi.Travel, Sight Safari.

Google Trips относится к категории сервисов, направленных на планирование путешествий. Они отличаются дополнительными возможностями подбора авиа- и ЖД-билетов, номеров отелей прямо в приложении, а также содержанием каталогов интересных мест и направлений.

izi.Travel – приложение-аудиогид с каталогом готовых маршрутов и тематических квестов. Данный сервис не позволяет строить собственных маршрутов, но содержит путеводители по множеству городов всего мира.

Sight Safari – приложение, позволяющее строить персональные пешеходные маршруты по городам мира. Оно отличается функционалом, позволяющим учесть пожелания пользователя – выбрать интересующие его

категории объектов, а также отрегулировать продолжительность маршрута (от кратчайшего, до интересного). К сожалению, данный сервис позволяет строить только пешеходные маршруты внутри населённых пунктов.

Все существующие сервисы обладают богатым и ценным функционалом, однако ни один из них в отдельности не решит в полной мере проблемы Центра развития туризма. Организации требуется вести собственный каталог маршрутов, локализованных в Кировской области. Наиболее предпочтительными являются автомобильные маршруты, проходящие между населёнными пунктами. Каждый маршрут содержит описание и фотографии.

Кроме того, согласно информации, предоставленной Центром развития туризма Кировской области, одной из актуальных потребностей целевой аудитории на текущий момент была выявлена потребность в персонализации туристских услуг. В следствие чего, помимо готовых маршрутов, у пользователя должна быть возможность построить и собственные маршруты, учитывающие его предпочтения в категориях объектов и времени путешествия.

Таким образом, организации необходима разработка собственного сервиса для визуализации и построения маршрутов, который, учитывая всё вышесказанное, будет также интегрирован с сайтом компании, для обмена дополнительной информацией.

1.3. Разработка технического задания на создание сервиса

1.3.1. Общие сведения

Настоящее техническое задание распространяется на создание программного продукта «Рекомендательный сервис построения персональных туристических маршрутов», далее «Сервис».

Наименования организаций – Заказчика и Разработчика

Заказчик

Наименование: КОГАУ «Центр развития туризма Кировской области»

Адрес местонахождения/Юридический адрес: 610020, Российская Федерация, Кировская область, г. Киров, ул. Спасская, д. 41а.

Разработчик

Наименование: Бокарева Дарья Геннадьевна

Адрес местонахождения/Почтовый адрес: 610020, Российская Федерация, Кировская область, г. Киров, ул. Свободы, д. 23.

Плановые сроки выполнения работ по реализации Сервиса

Сроки выполнения работ 1 этапа: 20.04.2020 – 02.06.2020.

Сроки выполнения работ 2 этапа: 03.06.2020 – 03.07.2020.

Сроки выполнения работ 3 этапа: 04.07.2020 – 04.08.2020.

1.3.2. Назначение и цели создания Сервиса

Назначение Сервиса

Сервис предназначен для визуализации разработанных Центром развития туризма Кировской области туристических маршрутов на интерактивной карте, с целью увеличения каналов информирования туристов и более полного предоставления информации о маршрутах, а также в дополнение к печатной продукции (буклеты, туристские карты). Сервис содержит дополнительные возможности в виде построения персональных автомобильных туристических маршрутов с учётом предпочтений пользователей, а также афишу мероприятий Кировской области. Предусмотрена авторизация пользователей и ведение личного профиля.

Цели создания Сервиса

Сервис направлен на достижение следующих целей:

- увеличение каналов информирования потребителей;
- повышение узнаваемости организации;
- увеличение туристского потенциала региона;
- персонализация предоставляемых туристских услуг;
- обеспечение доступности туристских ресурсов для всех слоёв населения.

В результате внедрения Сервиса ожидаются положительные изменения по следующим показателям эффективности:

- снижение трудозатрат на услуги консультирования;

- повышение туристического потока на афишируемые организацией мероприятия;
- повышение количества упоминаний событий, связанных с организацией, и самой организации в социальных медиа;
- увеличение въездного потока туристов на территорию Кировской области;
- увеличение туристской активности на территории Кировской области.

Объектом автоматизации служит процесс построения и визуализации туристического маршрута.

1.3.3. Требования к Сервису

Требования к структуре и функционированию Сервиса

Разрабатываемый Сервис должен быть организован по трехуровневой клиент-серверной архитектуре.

Сервис представляет собой мобильное приложение, поддерживаемое операционной системой Android версии 4.0.3 и выше. Клиентская часть приложения должна быть реализована на русском языке.

Серверная часть должна быть представлена в виде web-приложения и сервера базы данных.

Сервис предусматривает возможность пользования следующим видам пользователей с разграничением прав доступа:

- неавторизованный пользователь;
- авторизованный пользователь;
- администратор.

Неавторизованным пользователям Сервиса должен быть предоставлен функционал, позволяющий просматривать туристические маршруты на карте, создавать собственные маршруты, указывая начальную и конечную точки, а также промежуточные локации. Кроме того, пользователи могут настраивать следующие параметры (ограничения) при построении маршрута:

- категории предпочитаемых локаций;

- время прохождения маршрута.

Под категориями подразумеваются виды локаций, относящиеся к туристской тематике:

- парки (заповедные зоны, скверы);
- религиозные объекты;
- водоёмы (реки, пруды, озёра, фонтаны);
- исторические здания;
- произведения искусства;
- культурные объекты;
- монументы;
- обзорные площадки.

Авторизованный пользователь Сервиса может сохранять созданные маршруты, добавлять понравившиеся маршруты в «Избранное». Авторизованный пользователь должен получать рекомендации на основе его предыдущих действий в Сервисе, а также данных, предоставленных пользователем Сервису.

Всем пользователям доступен каталог с готовыми маршрутами, а также поиск маршрутов по каталогу. Поиск осуществляется по совпадению запроса с названием или описанием маршрутов.

Пользователи могут просматривать афишу мероприятий, предоставленную организацией. Администратор Сервиса имеет полные права на управление базой данных.

Интерактивная карта подключается при помощи API сервиса «Яндекс.Карты».

Авторизация пользователя осуществляется при помощи социальных сетей: VK, Facebook и Google, с использованием протокола OAuth 2.0 [4].

Источниками данных для разрабатываемого Сервиса являются:

- документы на бумажных носителях;
- печатная продукция Центра развития туризма Кировской области;

- информация, расположенная на действующих сервисах организации;
- брендбук организации;
- информация, расположенная в свободном доступе в сети Интернет.

Сервис должен поддерживать основной режим функционирования, соответствующий следующим требованиям:

- обеспечение работы пользователей 24 часа 7 дней в неделю;
- выполнение обозначенных функций в полном объёме;

При проведении профилактических мероприятий, устранения неполадок, замены программно-аппаратного обеспечения работа Сервиса приостанавливается.

Информация, расположенная в Сервисе, должна регулярно актуализироваться. Сервис должен постоянно регулироваться и сопровождаться техническими специалистами.

Требования к эргономике и технической эстетике

При формировании пользовательского интерфейса должны быть соблюдены следующие требования внешнего оформления, обеспечивающие удобный для конечного пользователя интерфейс:

- интерфейсы должны быть типизированы и соответствовать современным пользовательским сценариям работы с графическими интерфейсами;
- используемые шрифты должны быть хорошо читаемыми, без засечек, размер кегля не менее 8;
- цветовая палитра оформления должна обеспечивать хорошую читаемость информации;

В части диалога с пользователем должны выполняться следующие требования:

- при возникновении ошибок в работе компонента Сервиса на экран устройства должно выводиться сообщение с наименованием ошибки и с рекомендациями по её устранению на русском языке.

- пользователю должна быть доступна информация о приложении и организации, включая контактную информацию.

В части процедур ввода-вывода данных должна быть возможность многомерного анализа данных в табличном и графическом видах.

Требования к информационной безопасности

Обеспечение информационной безопасности Сервиса должно удовлетворять следующим требованиям:

- защита Сервиса должна обеспечиваться комплексом программно-технических средств и поддерживающих их организационных мер;
- защита Системы должна обеспечиваться на всех технологических этапах обработки информации и во всех режимах функционирования, в том числе при проведении ремонтных и регламентных работ;
- Сервис должен обеспечивать сохранность данных пользователей.

Требования по стандартизации и унификации

Документирование процесса проектирования Сервиса производится в соответствии со стандартом UML [5]. Описание модели данных производится в нотации Crow’s Foot [6]. Модель базы данных соответствует требованиям третьей нормальной формы. Для работы с БД должен использоваться язык запросов T-SQL [7].

1.3.4. Объём и содержание работ

1 этап. Разработка настоящего технического задания. Проектирование архитектуры сервиса, базы данных и проектирование пользовательского интерфейса. Разработка веб-приложения. Разработка клиентской части в виде функционала для хранения и вывода информации о готовых туристических маршрутах, предоставленных организацией, а также визуализация маршрутов на карте. Разработка сопроводительной документации.

2 этап. Реализация функционала построения персональных туристических автомобильных маршрутов по Кировской области. Разработка

рекомендательного алгоритма. Реализация дополнительного функционала (авторизация, афиша мероприятий).

3 этап. Запуск первой версии сервиса, публикация клиентской части Сервиса в Google Play. Сбор информации для рекомендательного алгоритма, сбор данных для анализа сервиса, доработка сервиса.

1.3.5. Порядок контроля и приемки Сервиса

Виды и объем тестирования

Тестирование Сервиса подразделяется на следующие виды:

- предварительные испытания эскизного проекта – прототипа Сервиса;
- опытная эксплуатация эскизного проекта – прототипа Сервиса;
- предварительные испытания Сервиса;
- опытная эксплуатация Сервиса;
- приемочные испытания Сервиса.

Состав, объем и методы предварительного тестирования и опытных эксплуатаций, определяются в документах, разрабатываемых на стадии подготовки сопроводительной документации.

Приёмка Сервиса осуществляется при успешном прохождении этапа тестирования.

1.3.6. Источники разработки

Настоящее ТЗ разработано на основе ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы (АС)» [8].

1.4. Выводы

При анализе деятельности Центра развития туризма Кировской области были изучены основные услуги, предоставляемые организацией, а также каналы их распространения, включая сервисы организации. Организацией была озвучена проблема и поставлена задача для её решения.

Проблема, с которой столкнулся Центр развития туризма, связана с недостатками текущих сервисов и каналов коммуникаций, не соответствующих в полной мере направлению деятельности организации.

Исследование рынка подтвердило актуальность и необходимость разработки собственного сервиса построения персональных туристических маршрутов, способного решить проблему организации.

Для решения поставленной задачи было разработано техническое задание, в котором обозначены сроки и этапы работ. Первый этап был реализован в рамках прохождения производственной практики. В Выпускной квалификационной работе реализуется второй этап работ. Третий этап планируется к реализации в рамках дальнейшего сотрудничества с Центром развития туризма Кировской области.

2. Математические основы работы сервиса

Построение персональных туристических маршрутов с математической точки зрения сводится к решению частного случая задачи поиска оптимального маршрута, наиболее известной, как задача коммивояжёра.

Задача коммивояжёра (Traveling Salesman Problem, TSP) – одна из самых известных задач комбинаторной оптимизации, заключающаяся в поиске оптимального маршрута, проходящего через указанные города хотя бы по одному разу с последующим возвратом в исходный город.

Впервые она была рассмотрена в 1930-х годах математиком и экономистом Карлом Менгером в Вене [9]. С тех пор её исследовали многие учёные в разных вариантах постановки и с разными ограничениями, применяя различные способы её решения [10]. Исследования данной задачи продолжаются и на сегодняшний день – разрабатываются новые методы и алгоритмы решения задачи, реализуются программы, которые позволяют работать со всё большим количеством узлов.

Задача построения туристического маршрута отличается наложением дополнительных ограничений при её решении – время, которым располагает турист, значимость достопримечательностей для конкретного туриста, а также время их посещения.

В данной главе будут рассмотрены существующие методы решения задачи коммивояжёра, описан метод решения частного случая задачи – построения оптимального туристического маршрута. Кроме того, Сервис будет включать в себя рекомендательный алгоритм, основанный на истории путешествий пользователя и выставленных им оценок, который также рассмотрен в текущей главе.

2.1. Обзор методов построения маршрутов

Постановка классической задачи коммивояжера

Для возможности применения математического аппарата к решению проблемы, её необходимо представить в виде математической модели.

TSP можно представить в виде модели на графе, используя вершины и ребра между ними. Вершины графа соответствуют городам, а рёбра (i, j) между вершинами i и j – пути сообщения между этими городами. Каждому ребру (i, j) можно сопоставить критерий выгодности маршрута $c_{ij} \geq 0$, под которым обычно понимается расстояние между городами, время, стоимость поездки и т.д.

В целях упрощения задачи и гарантии существования маршрута обычно считается, что модельный граф задачи является полностью связным. Если же между отдельными городами не существует сообщения, этого можно достичь путём ввода рёбер с максимальной длиной. Из-за большой длины такое ребро никогда не попадёт к оптимальному маршруту, если он существует.

Постановка оптимизационной задачи.

Пусть $I = \{1, \cdots, n\}$ – множество городов, матрица (c_{ij}) – попарные расстояния между городами, $1 \leq i, j \leq n$.

Необходимо найти контур минимальной длины, то есть маршрут, проходящий через каждую вершину ровно один раз и имеющий минимальный вес.

Переменные задачи:

$$x_{ij} = \begin{cases} 1, & i \neq j, \text{если из города } i \text{ едем в город } j \\ 0, & i = j, \text{в противном случае,} \end{cases}$$

$$\min \sum_{i \in I} \sum_{j \in I} c_{ij} x_{ij} \tag{1}$$

при ограничениях

$$\sum_{i \in I} x_{ij} = 1, \forall j \in I, \tag{2}$$

$$\sum_{j \in I} x_{ij} = 1, \forall i \in I, \tag{3}$$

$$\sum_{i \in S} \sum_{j \in I \setminus S} x_{ij} \geq 1, \forall S \subset I, S \neq 0,$$

или

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \forall S \subset I, S \neq 0,$$

где S – подмножество городов.

Дополнительные переменные:

$u_i \geq 0,$ – номер шага, на котором посетили город i.

Дополнительное ограничение для исключения замкнутых путей:

$$u_i - u_j + n x_{ij} \leq n - 1, i, j \in I \backslash i_1, i \neq j. \quad (4)$$

Таким образом, решение задачи коммивояжёра – это нахождение гамильтонова цикла (цикла, включающего ровно по одному разу каждую вершину графа) минимального веса в полном взвешенном графе.

Поставленная в (1) – (4) задача коммивояжёра будет использована при рассмотрении существующих методов решения TSP.

Алгоритмическая сложность задачи коммивояжера

Поскольку коммивояжёр в каждом из городов встает перед выбором следующего города из тех, что он ещё не посетил, в задаче коммивояжера для формирования оптимального маршрута из n городов существует $(n-1)!$ маршрутов для асимметричной и $(n-1)!/2$ маршрутов для симметричной задачи коммивояжёра – размер пространства поиска факториально зависит от количества городов.

Задача коммивояжёра относится к классу NP-полных задач [11]. Класс NP – это класс задач распознавания, в которых можно проверить ответ "да" за полиномиальное время. NP-полные задачи – самые трудные задачи в NP, то есть если существует точный полиномиальный алгоритм для решения одной из них, то существует точный полиномиальный алгоритм для решения всех задач из класса NP.

Кроме того, задача коммивояжёра относится к числу трансвычислительных (Transcomputational problem): уже при относительно небольшом числе городов (66 и более) она не может быть решена методом

перебора всех вариантов никакими теоретически мыслимыми компьютерами за время, меньшее нескольких миллиардов лет.

2.1.1. Точные алгоритмы

Алгоритмы для решения задачи коммивояжёра можно разделить на точные (Exact algorithm) и неточные (Non-exact algorithm). Точные алгоритмы включают в себя перебор всех возможных вариантов и дают, соответственно, точное решение задачи. Их особенность заключается в том, что в частных случаях решения могут быть быстро найдены, однако в общих случаях осуществляется перебор $n!$ циклов.

Полный перебор

Метод полного перебора или метод «грубой силы» (Bruto Force) – это один из самых очевидных способов решения задачи коммивояжера. Он заключается в переборе всех возможных вариантов путей и выборе из них оптимального.

Алгоритм полного перебора:

Шаг 1. Поиск общего числа всех возможных гамильтоновых циклов (контуров).

Шаг 2. Поиск веса каждого контура, путём сложения весов всех его рёбер.

Шаг 3. Выбор гамильтонова контура с минимальным весом, который и является решением задачи.

Преимуществом данного метода является гарантия нахождения точного решения задачи коммивояжера при его простоте.

Основным недостатком алгоритма является его неэффективность на больших объёмах данных из-за временной сложности. Так, для нахождения оптимального маршрута, состоящего из n пунктов, необходимо найти веса $(n-1)!$ гамильтоновых циклов.

Метод ветвей и границ

Одним из методов оптимизации полного перебора является метод ветвей и границ (Branch and Bound), впервые предложенный учёными А. Лендом и А. Дойгом в 1960 году [12], а в 1963 году модифицированный Дж. Литтлом, К. Мурти и другими учёными для решения TSP [13]. Он заключается в разбиении множества на два непересекающихся подмножества на каждом шаге ветвления и отсеве заведомо неоптимальных решений.

Алгоритм метода ветвей и границ [20]:

Шаг 1. Составляется таблица расстояний (весов) между заданными городами (матрица c_{ij}). Независимые (несмежные) вершины отмечаются как $c_{ii} = \infty$, это обозначение не позволит выбрать уже использованный маршрут.

Шаг 2. Таблица уменьшается путём вычитания из каждой строки, затем каждого столбца наименьшего элемента. Данное действие основано на том, что при вычитании константы из любого столбца или строки стоимость оптимального маршрута уменьшается на её величину, но маршрут остаётся тем же.

c_i – наименьший элемент строки:

$$c_i = min_j c_{ij}, \forall j \in I, \tag{5}$$

наименьший элемент столбца:

$$c_j = min_i(c_{ij} - c_i) c_j = \min(c_{ij} - c_i), \tag{6}$$

Уменьшение таблицы проводится в соответствии с формулой:

$$c'_{ij} = c_{ij} - c_i - c_j. \tag{7}$$

В результате получается приведенная матрица, в которой в каждой строке и в каждом столбце имеется хотя бы один нулевой элемент.

Сумма констант является нижней границей (оценкой) для всех вариантов маршрутов:

$$l = \sum c_i + \sum c_j . \tag{8}$$

Шаг 3. Производится выбор некоторого ребра графа, при котором все возможные варианты маршрута делятся на два подмножества: одни включают выбранное ребро, вторые – нет.

Для обоих подмножеств создается отдельная матрица расстояний и производится вычисление нижних границ.

Окончательно выбирается такое ребро с нулевым весом, для которого вес второго минимального ребра в строке или столбце максимальный. Выбранное ребро включается в маршрут для вариантов маршрута первого подмножества и исключается из всех вариантов маршрута второго подмножества. В результате нижняя оценка для всех вариантов маршрута второго подмножества увеличивается на вес второго минимального ребра в строке и столбце.

Шаг 4. Вычисляется оценка узла, смежного с краем, который имеет вес (i, j). В таблице расчетов вводится обозначение ограничения для коммивояжёра, которые не позволяют ему вернуться из города i в город j. Далее удаляется столбец i и j из этой таблицы и повторяются шаги 2 и 3.

Шаг 5. Строится дерево ветвления. В ветвящемся дереве начальному узлу назначается граница l. Края, выходящие из этого узла, разделяются на содержащие (i, j) и не содержащие (i, j).

Алгоритм завершается, когда остаётся матрица размерности 2, из которой добавление рёбер в маршрут происходит тривиальным способом.

Алгоритм является надёжным вариантом решения целочисленных задач, но на больших объёмах данных метод ветвей и границ также является неоправданно трудоёмким.

Метод динамического программирования

Применение метода динамического программирования впервые предложено Р. Беллманом [22]. Данный метод является вычислительно рациональным для числа городов не более 17. Суть метода заключается в итерационном нахождении функции Беллмана, отражающей длину пути

минимальной длины, соединяющего два города и проходящего один и только один раз через каждый из n городов.

Алгоритм метода динамического программирования Беллмана [23]:

Шаг 1. Составляется таблица расстояний между заданными n городами (матрица c_{ij}). Несмежные вершины описываются как $c_{ii} = \infty$.

Шаг 2. Записывается функция Беллмана для конечного решения:

$$f(i; j_1, j_2, j_3, j_2 \dots, j_n) = \min_{1 \le m \le n} \{c_{ij_m} + f(i; j_1, j_2, j_3, \dots j_{m-1}, j_{m+1}, \dots, j_n)\}. \quad (9)$$

Решение разбивается на этапы – количество этапов равно количеству городов обхода.

Шаг 3. Создается таблица множества решений, в которой по строкам строки интерпретируются как номера шагов, а колонки определяют значения функций Беллмана.

Шаг 4. Для каждого этапа рассчитывается функция Беллмана и заносится в таблицу. Итерационный процесс начинается с вычисления функции:

$$f(i, j) = c_{ij} + c_{j0}, \quad (10)$$

из которой получается:

$$f(i; j_1, j_2).$$

На основании данных первой строки по рекуррентной формуле получаются значения функций Беллмана для второй строки

$$f(i; j_1, j_2, j_3),$$

которые используются для вычислений значения функций третьей строки и т.д.:

$$f(i; j_1, j_2, j_3, j_2 \dots, j_{n-1}).$$

Процесс вычислений и заполнения строк таблицы заканчивается получением значения функции Беллмана для -го шага:

$$f(0; j_1, j_2, j_3, j_2 \dots, j_n).$$

Шаг 5. Для определения требуемого маршрута в выражении на последнем шаге находят элемент, минимизирующий его. Таким образом, путь получения данного элемента – последовательность значений m, которая минимизирует

значение в скобках правой части выражения (9) дает искомый минимальный путь.

Преимущество данного метода заключается в повышении эффективности вычислительных повторений, так как метод позволяет хранить промежуточные результаты и при необходимости обращаться к ним снова.

2.1.2. Неточные алгоритмы

Неточные алгоритмы дают приближенный к оптимальному результат и применяются для задач такой сложности, что существующие точные решения требуют значительных и неоправданных временных затрат, или как часть более сложного алгоритма, который позволяет решить задачу точно [14].

Нижняя грань Хелд-Карпа

Для измерения эффективности эвристических алгоритмов при решении задачи коммивояжёра используется распространённый способ, заключающийся в сравнении результатов с нижней гранью Хелд-Карпа.

Нижняя грань Хелд-Карпа (The Held-Karp Lower Bound) – это решение TSP за полиноминальное время с использованием симплекс-метода. Она приблизительно на 0,8% ниже оптимальной длины маршрута, при этом гарантированно не превышает более, чем на 2/3 оптимальное время [15].

Жадный алгоритм

Жадный алгоритм (Greedy algorithm) заключается в принятии локально оптимальных решений, допуская, что конечное решение также окажется оптимальным. Решение задачи коммивояжера с использованием жадного алгоритма заключается в исследовании всех n рёбер, выходящих из города-узла, и выбора самых коротких. Если эти рёбра формируют гамильтонов цикл, тогда оптимальное решение считается найденным [16].

Трудоемкость решения задачи жадным алгоритмом равна $O(n^2)$. Нижняя граница стоимости оптимального маршрута выше нижней границы Хелд-Карп на 15-20% [17].

Метод ближайшего соседа

Метод ближайшего соседа (NearestNeighbour, NN) относится к жадным алгоритмам и основывается на простом эвристическом правиле – всегда выбирать близлежащий город.

Алгоритм ближайшего соседа:

Шаг 1. Выбирается начальный город.

Шаг 2. Осуществляется поиск ближайшего города, ещё не включённого в маршрут, и переход в него.

Шаг 3. Осуществляется проверка наличия городов, не включённых в маршрут и, при нахождении таких городов, повторяется второй шаг.

Шаг 4. Маршрут завершается путём добавления ребра из последнего города в начальный.

Трудоемкость решения задачи в общем случае равна $O(n^2)$. Нижняя граница стоимости оптимального маршрута выше нижней границы Хелд-Карп на 10% [18].

Генетический алгоритм

Основы теории генетических алгоритмов (Genetic algorithm, GA) сформулированы Дж. Г. Холландом и в дальнейшем были развиты рядом других исследователей. Наиболее известной и часто цитируемой в настоящее время является монография Д. Голдберга, где систематически изложены основные результаты и области практического применения генетических алгоритмов. Являясь эвристическими оптимизационными алгоритмами, GA применяются для решения задачи коммивояжера. Для генетических алгоритмов задача коммивояжера чувствительна к выбору начальных условий [27], поэтому принимается, что начальной и конечной точкой алгоритма будет являться первый элемент в списке городов c_1.

В основе работы генетического алгоритма лежит нахождение локального экстремума целевой функции $F(x_1, x_2, \dots x_n)$, в которой переменные называются параметрами функции. Параметры задачи являются генетическим материалом –

генами. Совокупность генов составляет хромосому. Каждая особь обладает своей хромосомой, и, следовательно, своим набором параметров. Подставив параметры в целевую функцию, можно получить определённое значение. То, насколько это значение удовлетворяет поставленным условиям, определяет характеристику особи, которая называется приспособленностью $Fitness(\varphi)$. Для задачи коммивояжера она равна:

$$Fitness(\varphi) \equiv c\big(\varphi(1), \varphi(n)\big) + \sum \big\{c\big(\varphi(i), \varphi(i+1)\big)\big\} \to min \qquad (11)$$

Функция, определяющая приспособленность должна удовлетворять следующему условию: чем «лучше» особь, тем выше приспособленность.

Генетические алгоритмы работают с популяцией как правило фиксированного размера. Особи «скрещиваются» между собой с помощью генетических операторов, формируя новые наборы генов. Выбор особей для скрещивания проводится согласно селективной стратегии (selection strategy). Процесс формирования особей является итерационным и заканчивается при значениях целевой функции близких к экстремуму, либо равному ему.

Оператор кроссинговера (crossover operator), является основным генетическим оператором, за счет которого производится обмен генетическим материалом между особями. Процесс скрещивания особей для решения задачи коммивояжера выглядит следующим образом:

Шаг 1. Для каждой пары хромосом случайным образом выбирается точка жадного алгоритма и в качестве номера стартовой вершины графа берётся номер отмеченного гена в хромосоме.

Шаг 2. Сравнивается частичная стоимость путей, ведущих из текущих вершин в хромосомах родителях, и выбирается кратчайший путь.

Шаг 3. Если выбранная таким образом вершина графа уже была включена в частичный путь, то берётся случайная величина из числа ещё не отмеченных вершин и значение текущей присваивается полученной вершине.

Шаг 4. При преждевременном образовании циклов выбирается другой кратчайший путь.

Шаг 5. Повторяются шаги 2 и 3 до тех пор, пока не будет построен гамильтонов цикл с минимальной суммарной стоимостью ребер.

Шаг 6. Завершение алгоритма. При этом решение-потомок в данном алгоритме формируется как последовательность вершин графа в том порядке, в котором они становились текущими.

Генетические алгоритмы показывают высокую эффективность в поиске оптимального маршрута при небольшом числе городов. Однако, при количестве городов более 20, алгоритм уже не гарантирует нахождения оптимального маршрута за приемлемое время и с увеличением числа городов начинает существенно проигрывать, к примеру, жадному алгоритму.

Муравьиный алгоритм

Муравьиный алгоритм (Ant Colony Optimization, ACO) – эффективный полиноминальный алгоритм, на идею создания которого учёного Марко Дориго вдохновило поведение муравьёв [19]. Так, идея муравьиных алгоритмов основывается на принципах самоорганизации муравьиной колонии и их способности находить кратчайший путь от муравейника к источнику пищи, при этом адаптируясь к изменяющимся условиям и находя всё новые кратчайшие пути. Эта способность реализуется с помощью феромона, которым при движении метит свой путь муравей, а другие муравьи используют данную информацию и следуют по маршруту, наиболее обогащённому данным веществом.

Муравьи, выбравшие кратчайший путь, соответственно, проходят его быстрее и всё больше обогащают феромоном. В то же время, вещество постепенно испаряется и это гарантирует, что найденное локально-оптимальное решение не будет единственным и муравьи будут продолжать искать и другие пути.

Решение задачи коммивояжёра с использованием муравьиного алгоритма выглядит следующим образом [21]:

Для муравья, находящегося в городе i, переход в город j зависит от трех составляющих: памяти муравья, видимости и виртуального следа феромона.

Память муравья – это список тех городов, в которых муравей уже побывал и в которые заходить еще раз ему нельзя.

За $J_{i,k}$ принимается список городов, которые муравей k, находящийся в городе i еще не посетил – это дополнение памяти.

Видимость – величина, обратная расстоянию: $\eta_{ij} = 1/c_{ij}$, выражающая желание муравья посетить город j из города i. Желание зависит от близости города: чем он ближе, тем больше желание посетить его. Но для нахождения кратчайшего маршрута видимости недостаточно.

Виртуальный след феромона на пути между городами i и j представляет собой подтвержденное опытом муравьев желание посетить город j из города i. В отличие от видимости, эта величина изменяется после каждой из итераций алгоритма. Количество виртуального феромона на пути из города i в город j на итерации t обозначается как $\tau_{ij}(t)$.

Немаловажную роль в алгоритме также играет вероятностно-пропорциональное правило, которое определяет вероятность перехода -го муравья из города i в город j на t-й итерации:

$$P_{ij,k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in J_{i,k}}[\tau_{il}(t)]^{\alpha}[\eta_{il}]^{\beta}}, \text{если } j \in J_{i,k}, \\ 0, \text{если } j \notin J_{i,k} \end{cases} \quad (12)$$

где α и β – два регулируемых параметра, задающие веса следа феромона и видимости при выборе маршрута, соответственно.

При $\alpha = 0$ выбирается ближайший город, что соответствует основному правилу жадного алгоритма.

При $\beta = 0$ работает только феромонное усиление, что приводит к тому, что маршруты вырождаются к одному субоптимальному решению.

Значения $P_{ij,k}(t)$ не изменяются во время итерации, но для разных муравьев в одном и том же городе, в общем случае, различны, т. к. $P_{ij,k}(t)$ зависит от $J_{i,k}$.

После завершения маршрута каждый муравей k откладывает на ребре (i,j) такое количество феромона:

$$\Delta\tau_{ij,k}(t) = \begin{cases} \frac{Q}{L_k(t)}, \text{если } (i,j) \in T_k(t) \\ 0, \text{если } (i,j) \notin T_k(t) \end{cases}, \quad (13)$$

где $T_k(t)$ – маршрут, пройденный муравьем k на итерации t;

$L_k(t)$ – длина этого маршрута;

Q – регулируемый параметр, значение которого одного порядка с длиной оптимального маршрута.

Кроме того, для нахождения решения необходимо обеспечить испарение феромона – уменьшение его количества, накопившееся во время предыдущих итераций. С этой целью вводится коэффициент испарения феромона $p \in [0,1]$. Тогда за правило обновления феромона принимается следующее:

$$\tau_{ij}(t+1) = (1-p)\tau_{ij}(t) + \Delta\tau_{ij}(t), \quad (14)$$

где

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij,k}(t),$$

а m – количество муравьёв в колонии.

Таким образом, по сравнению с точными методами (например, динамическим программированием или методом ветвей и границ), муравьиный алгоритм находит близкие к оптимальному решения за значительно меньшее время даже для задач с n более 20. Время оптимизации муравьиным алгоритмом является полиномиальной функцией от размерности $O(t * n^2 * m)$.

Его основным недостатком считается, что первое полученное решение может оказаться крайне неоптимальным, но при повторном прохождении алгоритм способен дать достаточно точное решение [19].

Существует достаточно много методов решения задачи коммивояжёра, однако в последних работах [24–26] чаще используются генетические или муравьиные алгоритмы, доказывающие свою эффективность над другими методами.

В таблице 1 представлено сравнение генетического и муравьиного алгоритмов в нахождении длин маршрутов с различным числом городов [21].

Таблица 1 – Сравнение эвристических алгоритмов оптимизации маршрута коммивояжёра

	TSP для 50 городов	TSP для 75 городов	TSP для 100 городов
Генетические алгоритмы	428	545	22761
Муравьиные алгоритмы	425	535	21282

Из таблицы следует, что муравьиный алгоритм способен найти более оптимальный маршрут во всех описанных случаях. Поэтому, для реализации метода построения персональных туристических маршрутов, за основу будет взят муравьиный алгоритм.

2.2. Разработка метода построения туристических маршрутов

Задача построения туристического маршрута отличается от классической задачи коммивояжёра, в связи с чем появляется необходимость выделить критерии, определяющие эффективность построения туристического маршрута, а также модифицировать существующий алгоритм Ant Colony Optimization, чтобы достичь поставленной цели.

Данный раздел посвящён постановке задачи построения туристического маршрута и разработке алгоритма в соответствии с поставленной задачей и выделенными показателями эффективности.

Постановка задачи для разработки алгоритма

Задача построения туристического маршрута заключается в построении тура между достопримечательностями и иными местами, интересующими туриста. Данная задача от классической задачи коммивояжёра (1) – (4) отличается в первую очередь тем, что туристу нет необходимости завершать

маршрут, возвращаясь в начальную точку, т.е. конечная точка задаётся самим туристом и может быть любой.

Также существует дополнительное ограничение в виде времени, которым располагает турист для прохождения маршрута. В связи с этим, отпадает необходимость посещения всех имеющихся пунктов. Ограничением длины маршрута будет параметр S_{max}, задающийся туристом. Под длиной в задаче понимается время прохождения, тогда матрица (c_{ij}) – матрица, содержащая попарные значения времени прохождения между городами.

Кроме того, в задаче появляется необходимость выделения дополнительного времени на осмотр достопримечательностей, перерыва на отдых и развлечения. В представлении в виде графа, это заключалось бы в добавлении к существующим вершинам дополнительных петель, вес которых будет равен продолжительности посещения данного пункта.

Необходимо также учитывать интересы туриста к конкретным местам, т. е. уровень желания посетить тот или иной объект. Таким образом, все места группируются по видам, для которых можно задать одинаковое значение релевантности r_i, однако турист может изменять значение данного параметра в соответствии с собственными предпочтениями.

Показатели эффективности построения туристического маршрута

В соответствии с поставленной задачей, оптимальным можно считать маршрут, удовлетворяющий следующим условиям:

1) Время построенного маршрута не превышает времени, которым располагает турист:

$$L \leq S_{max}. \tag{15}$$

2) Маршрут имеет наибольшую релевантность – численное выражение, соответствующее интересам туриста:

$$F = \sum r_i \rightarrow max, \tag{16}$$

3) Маршрут является наиболее оптимальным из возможных маршрутов, выполняющих первые два условия.

Модификация алгоритма ACO

Необходимо модифицировать классический муравьиный алгоритм, используемый для решения TSP (12) – (14).

Изначальное количество виртуального феромона примем как $\tau_0 = 0{,}5$.

Тогда в классическом алгоритме на выбор следующей для посещения вершины влияет расстояние до неё. Для решения задачи построения оптимального туристического маршрута, кроме расстояния следует учитывать и значение релевантности r_i, при этом принимается, что оба параметра влияют на выбор равным образом, для этого необходимо нормировать релевантность:

$$rn_i = \frac{r_i}{S_r}, i = \overline{1,n}, \tag{17}$$

где n – число имеющихся вершин, S_r – коэффициент нормирования, вычисляющийся по формуле:

$$S_r = \sum_{j=1}^{n} rj\,. \tag{18}$$

Память муравья обозначается как $M_k(t)$. Таким образом, второе условие оптимальности маршрута будет выглядеть следующим образом:

$$F = \sum\nolimits_{i \in M_k(t)} rn_i \to max. \tag{19}$$

Далее необходимо определить число муравьёв в колонии. Обычно принято запускать порядка 20 муравьёв.

Формула (12) нахождения вероятности выбора остаётся прежней.

Теперь необходимо осуществить отбор маршрутов по продолжительности не превышающих поставленного туристом ограничения. Кроме того, расчёт должен содержать длину до выставленной конечной точки. Также в каждой вершине, имеющей петлю, необходимо включать продолжительность прохождения этой петли.

После совершенной итерации на ребре (i, j) откладывается следующее количество виртуального феромона:

$$\Delta\tau_{ij,k}(t) = \begin{cases} \frac{S_{max}\sqrt[p]{f_{k,t}}}{L_k(t) * F}, \text{если } (i,j) \in T_k(t) \\ 0, \text{если } (i,j) \notin T_k(t) \end{cases}, \quad (20)$$

где $f_{k,t} = \sum_{i \in T_k(t)} rn_i$; $F = \max_{k,t} f_{k,t}$.

Поиск оптимального туристического маршрута завершается, когда все муравьи в рамках одной итерации пройдут по одинаковому пути, либо когда пройдёт g итераций. Кроме того, необходимо предотвратить возможность случайного выбора всеми муравьями одинаковых маршрутов на начальном этапе работы алгоритма, запустив для этого d итераций.

В ходе экспериментов с начальными количеством виртуального феромона τ_0, количеством запускаемых муравьёв m, начальным числом итераций d, и значением конечной итерации g, подбираются значения для нахождения наиболее оптимального решения за полиноминальное время. При этом, основа алгоритма в виде Ant Colony Optimization и внесение в неё несущественных изменений для частного случая, даёт гарантию нахождения маршрута, близкого к оптимуму.

2.3. Описание рекомендательного алгоритма

Персонализация предоставляемых услуг заключается не только в возможности пользователем настроить маршрут по его предпочтениям. Важную роль здесь играют полезные рекомендации, предоставленные Сервисом пользователю. Если рекомендации совпадают с желанием пользователя – он гораздо меньше времени тратит на подбор услуги, а уровень его удовлетворённости Сервисом повышается.

Для построения системы рекомендаций используются рекомендательные алгоритмы, предсказывающие, какие объекты (фильмы, музыка, книги, новости, веб-сайты и др.) будут интересны пользователю, используя определенные данные, полученные от него.

Данные о пользователях собираются с помощью использования сочетания явных и неявных методов [47]. Примеры явного сбора данных:

- запрос у пользователя оценки объекта по дифференцированной шкале;
- запрос ранжирования группы объектов от наилучшего к наихудшему;
- вопрос о том, какой из двух представленных объектов лучше.

Примеры неявного сбора данных:

- наблюдение за тем, что просматривает пользователь в интернет-магазинах или базах данных другого типа;
- ведение записей о поведении пользователя в сети Интернет и приложениях.

Рекомендательные системы сравнивают однотипные данные от разных людей и вычисляют список рекомендаций для конкретного пользователя. В случае с разрабатываемым сервисом, важно порекомендовать пользователю потенциально интересные для него готовые туры. Для этого понадобятся данные о выбранных и оценённых раннее маршрутах. Для построения прогноза используется метод коллаборативной фильтрации.

Коллаборативная фильтрация (Collaborative filtering) – метод прогнозирования (рекомендаций), использующий известные оценки группы пользователей для прогнозирования неизвестных предпочтений другого пользователя [28]. Его основное допущение состоит в следующем: те, кто одинаково оценивал какие-либо объекты в прошлом, склонны давать похожие оценки другим объектам в будущем.

Алгоритм коллаборативной фильтрации:

Шаг 1. Рассчитывается коэффициент подобия пользователей. В качестве него используется коэффициент корреляции Пирсона:

$$r_{xy} = \frac{\sum_{i=1}^{i=N}\left(x - \overline{X}\right) \cdot (y - \bar{Y})}{\sqrt{\sum_i \left(x - \overline{X}\right)^2 \cdot \sum_i \left(y - \overline{Y}\right)^2}} \quad (21)$$

Параметрами служат маршруты пользователя с поставленными оценками по пятибалльной шкале.

Шаг 2. После поиска подобных пользователей осуществляется ранжирование и выбор наиболее подходящих по коэффициенту пользователей.

Шаг 3. Составляется таблица с подходящими пользователями, содержащая маршруты из их списков, которых нет у исходного пользователя, а также их оценки.

Шаг 4. По каждому маршруту производится умножение коэффициента подобия пользователя на оценку, поставленную им.

Шаг 5. Вычисляется сумма коэффициентов подобия всех пользователей, а также сумма показателей 4 шага.

Шаг 6. Вычисляется прогнозируемая оценка пользователя: итоговая сумма показателей делится на сумму коэффициентов подобия по каждому маршруту.

Шаг 7. Маршруты ранжируются по наиболее высокой прогнозируемой оценке для показа рекомендации.

В результате пользователю выводится список рекомендованных маршрутов из общего каталога, которые он с большой вероятностью оценит высоко, как это сделали подобные ему пользователи.

2.4. Выводы

В данной главе была рассмотрена классическая задача коммивояжёра, описана её математическая модель, а также проведено исследование существующих методов её решения и определены наиболее эффективные алгоритмы решения TSP.

На основе TSP была поставлена частная задача поиска оптимального туристического маршрута с некоторыми отличиями от классической задачи коммивояжёра. Для этого были введены дополнительные параметры, а также выделены условия эффективности построенных туристических маршрутов.

Для решения поставленной задачи была предложена модификация классического муравьиного алгоритма. Внесённые изменения позволяют учесть ограничения, выставляемые туристом для построения оптимального тура.

Описан алгоритм работы рекомендательной подсистемы Сервиса, который позволяет предлагать пользователю наиболее подходящие для него готовые туры. Рекомендации выводятся на основе поиска подобных пользователей и их оценок, за счёт которых вычисляется прогнозируемая оценка пользователя.

3. Проектирование и разработка сервиса

Разработка Сервиса включает в себя в формализацию требований технического задания, проектирование его структурных элементов: архитектуры, базы данных, сервера приложений, мобильного клиента, и непосредственно их реализацию. Кроме того, необходимо выбрать средства и платформы разработки серверной и клиентской части, а также наиболее подходящую СУБД.

3.1. Проектирование сервиса и выбор средств реализации

3.1.1. Проектирование архитектуры сервиса

Формализация требований к функционалу Сервиса представлена в виде диаграммы вариантов использования (Use Case Diagram) на рис. 4.

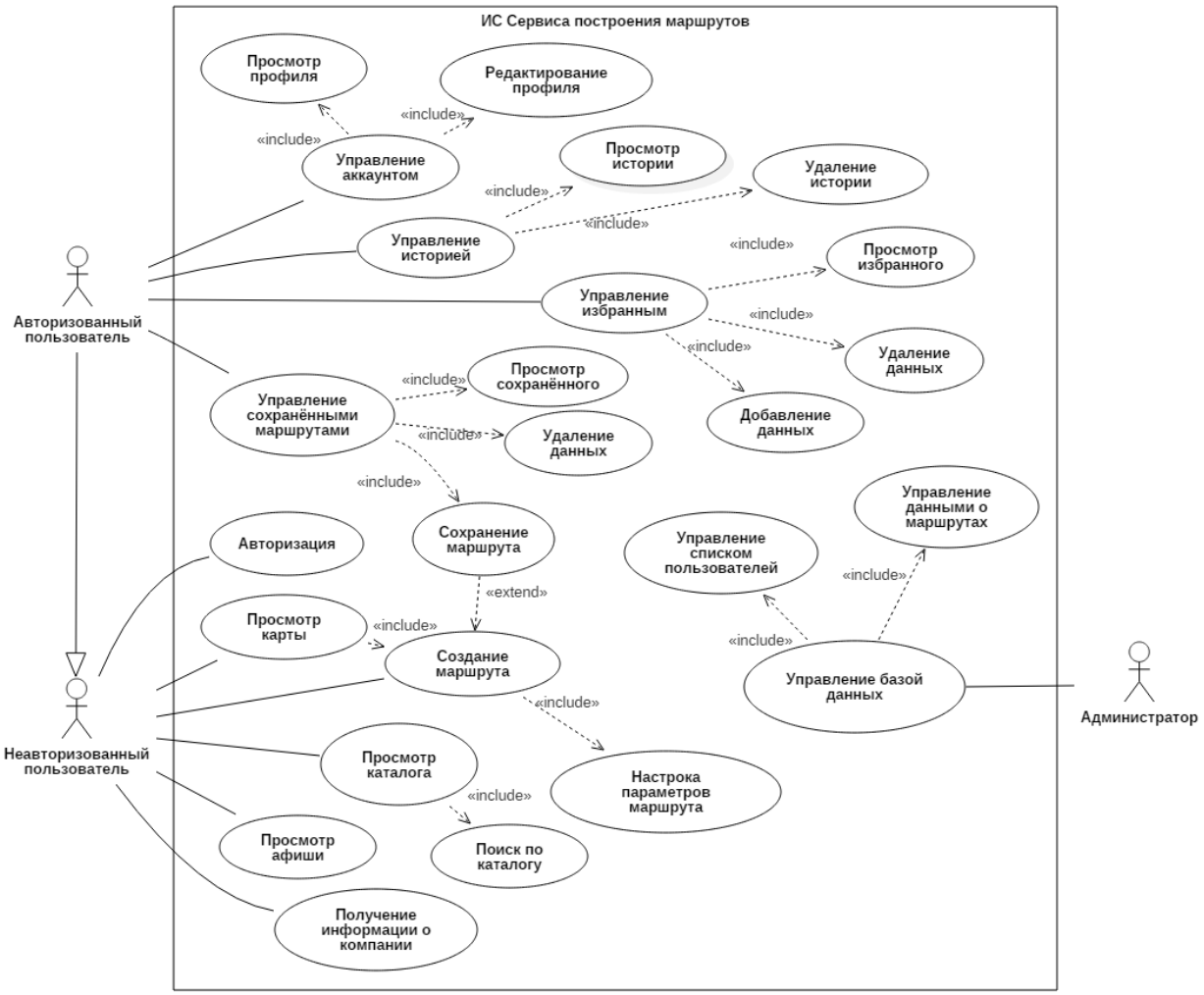


Рисунок 4 – Use Case диаграмма

Пользователи системы разделены на 3 роли:

1) неавторизованный пользователь;
2) авторизованный пользователь;
3) администратор.

Неавторизованный пользователь имеет беспрепятственный доступ к основному функционалу системы:

- просмотр каталога маршрутов;
- поиск по каталогу;
- просмотр афиши мероприятий;
- просмотр карты;
- создание и настройка параметров маршрута;
- получение информации о компании;
- настройка приложения;
- авторизация.

Такие функции, как хранение персональных маршрутов, добавление маршрутов в избранное и история путешествий доступны только авторизованному пользователю, т. к. будут привязаны к его идентификатору. Таким образом, авторизованный пользователь может выполнять функции неавторизованного пользователя, а также:

- управление аккаунтом (просмотр и редактирование профиля);
- управление сохранёнными маршрутами (добавление, просмотр и удаление);
- управление избранным (добавление, просмотр и удаление);
- управление историей путешествий (просмотр и очистка данных).

Администратор системы наделён всеми правами над управлением базой данных (управление списком пользователей и данными о маршрутах, редактирование всех таблиц базы данных).

Сервис построения персональных туристических маршрутов организован в виде трёхуровневой клиент-серверной архитектуры [29], где клиентской

частью приложения является мобильное приложение, серверной – сервер приложений и сервер базы данных.

Клиент-серверная архитектура обладает рядом преимуществ. В первую очередь, она способна обеспечить сохранность информации и её защищенность от несанкционированного доступа. Ведение базы данных осуществляется на сервере базы данных, что обеспечивает независимость обработки данных в базе от программ пользователя. Защитить информацию на сервере базы данных проще, благодаря гибкому администрированию прав доступа. При необходимости, прямой доступ может быть ограничен до определенного поля таблицы или совсем запрещен. Такая архитектура также является устойчивой к сбоям – сбой при работе клиента не сказывается на целостности данных и их доступности для других клиентов. Такой тип архитектуры также снижает нагрузку сети, чем обеспечивает её большую пропускную способность и возможность обслуживать большее число пользователей. Кроме того, клиент-серверная архитектура способна к расширению, обеспечивая возможность системы адаптироваться к росту количества пользователей и увеличению объема базы данных без замены программного обеспечения. В основном, масштабирование производится за счет наращивания аппаратных средств.

Схема данной архитектуры, разработанная с учётом требований технического задания, представлена в виде диаграммы развёртывания (Deployment Diagram) на рисунке 3.

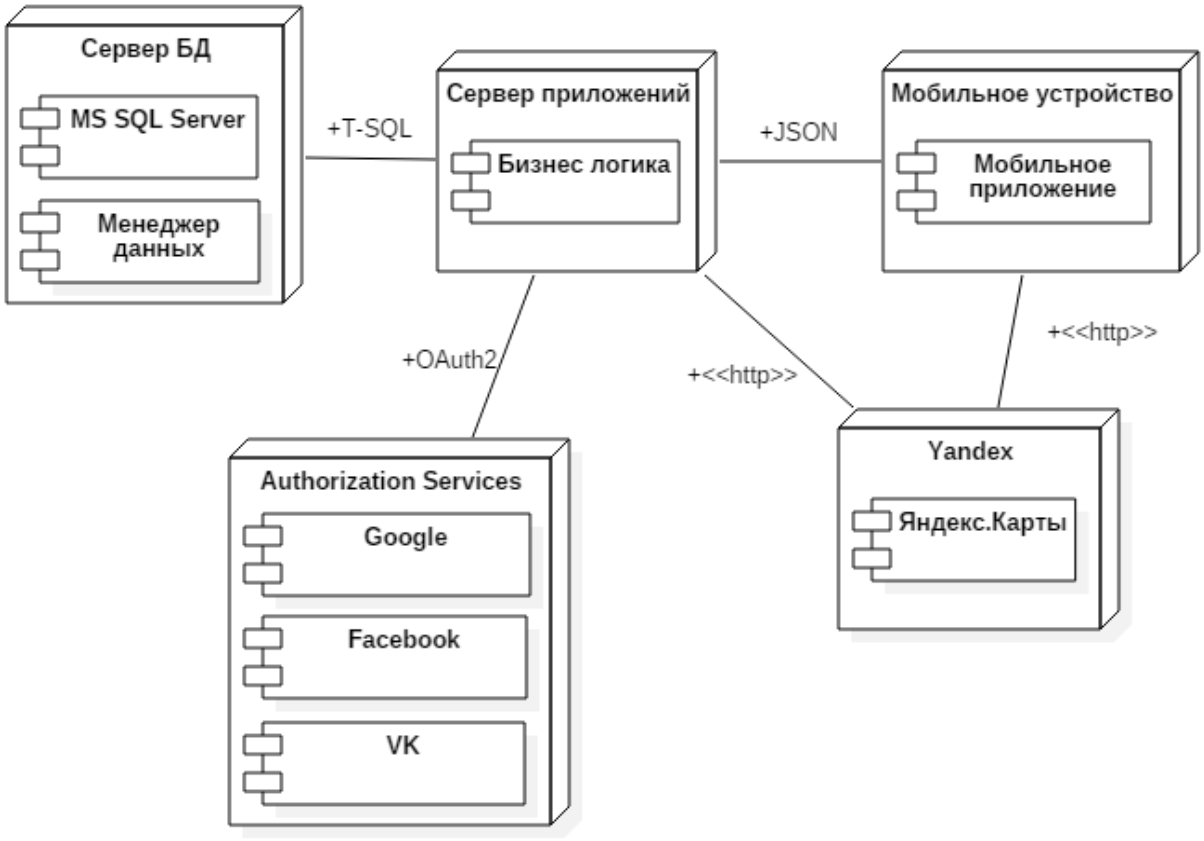


Рисунок 3 – Диаграмма развёртывания

Слои клиент-серверной архитектуры обозначены в виде трёх узлов: сервер базы данных, сервер приложений и мобильное устройство. Кроме архитектурных слоёв обозначены такие узлы, как Authorization Services и Yandex.

Authorization Services – это сервисы, с помощью которых, согласно ТЗ, пользователь сможет авторизоваться в приложении. Протоколом авторизации служит OAuth 2.0. Данный протокол позволяет выдать какому-либо сервису права на доступ к ресурсам пользователя, расположенных на другом сервисе.

Это обеспечивает дополнительную безопасность данным пользователя. Разрабатываемый Сервис не будет хранить логинов и паролей пользователей, а сервисы, с помощью которых будет реализована авторизация (VK, Facebook, Google) – выдадут лишь ограниченные права доступа к конкретным данным и действиям.

Yandex – узел, обозначающий связь разрабатываемого Сервиса с сервером компании Яндекс. Данное соединение необходимо для получения доступа к сервису Яндекс.Карты и интеграцию интерактивной карты в мобильное приложение при помощи Yandex MapKit API. Кроме мобильного клиента, соединение с сервером Яндекса будет осуществляться и веб-сервисом.

3.1.2. Проектирование базы данных

Модель данных, предназначенная для схематичного представления базы данных представлена в виде диаграммы сущность-связь, или ER-диаграммы (рис. 5). Схема базы данных реализована в нотации Crow’s Foot и удовлетворяет третьей нормальной форме [30].

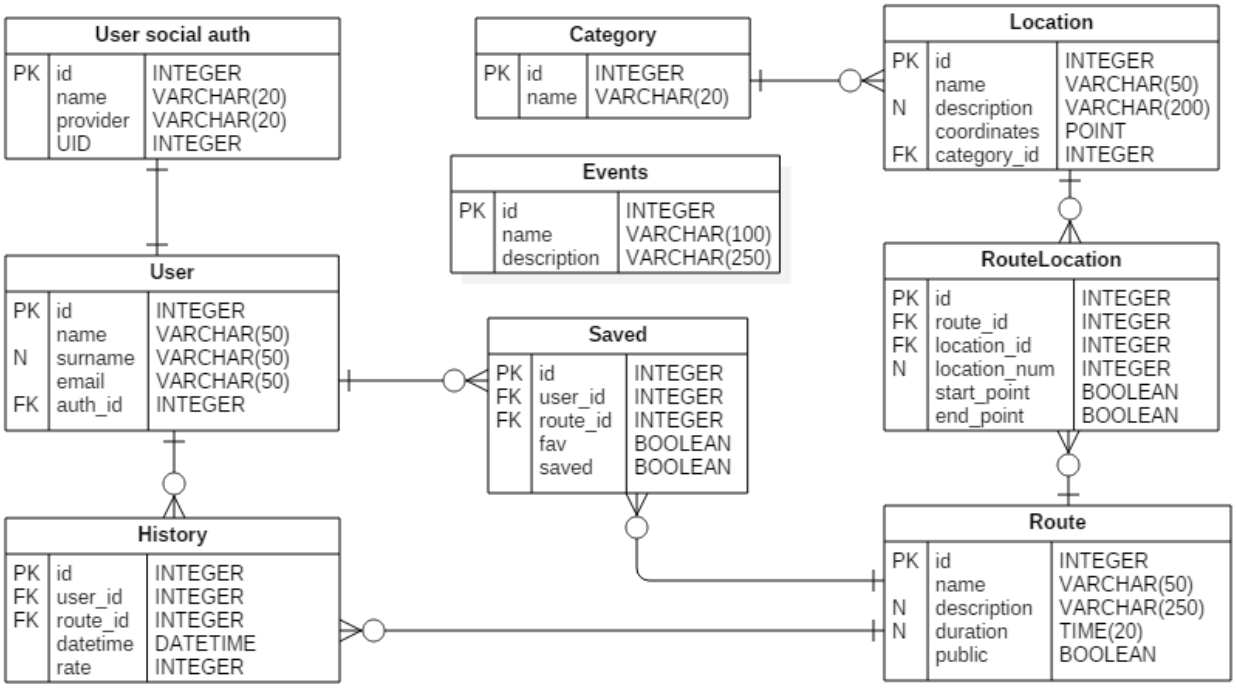


Рисунок 5 – ER-диаграмма

Для реализации сервиса, выделены следующие сущности:

- User social auth – таблица, предназначенная для хранения данных при авторизации через социальные сети;
- User – таблица, хранящая данные пользователей;

- Location – таблица, в которой хранится информация о локациях Кировской области, необходимая для построения маршрутов (например, координаты);
- Category – виды категорий, к которым относятся локации (например, парки, достопримечательности, музеи);
- Route – таблица, предназначенная для хранения маршрутов;
- RouteLocation хранит данные о локациях, составляющих определённый маршрут;
- Events – таблица для хранения афиши мероприятий.
- Saved – таблица, в которой хранятся избранные и сохранённые пользователем маршруты;
- History – таблица, предназначенная для хранения истории путешествий пользователя.

3.1.3. Проектирование пользовательского интерфейса

Пользовательский интерфейс представляют экранные формы мобильного приложения. Для проектирования пользовательского интерфейса необходимо их обозначить и иерархически структурировать. Экраны приложения выделены в соответствии с необходимым для неавторизованного и авторизованного пользователя функционалом Сервиса (рис. 4). Структура экранных форм представлена в Приложении А.

При проектировании пользовательского интерфейса были учтены требования и тенденции UX/UI дизайна. UX-дизайн представляет собой проектирование интерфейса на основе исследования пользовательского опыта и поведения. UI-дизайн включает в себя работу над графической частью интерфейса: анимацией, иллюстрациями, кнопками, меню, шрифтами и др.

Интерфейс должен соответствовать следующим требованиям [31]:

- Ясность. В интерфейсе не должно быть двусмысленности, текст и структура должны направлять к цели.

- Лаконичность. Интерфейс не должен быть перегружен подсказками, всплывающими окнами и анимацией.
- Узнаваемость. Интерфейс должен быть интуитивно понятным, а элементы узнаваемыми.
- Отзывчивость. Хороший интерфейс реагирует на действия пользователя мгновенно. Пользователь всегда должен получать ответ на свои действия. За это отвечают понятный текст, анимация и иконки.
- Постоянство. Необходимо соблюдать постоянство для всех разделов продукта. Элементы интерфейса должны вести себя одинаково на любой странице.
- Эстетика. Визуально интерфейс должен быть привлекательным. Хороший интерфейс, в котором пользователю приятно работать, ничто не раздражает и не отвлекает его от решения задачи.
- Эффективность. Помимо внешней привлекательности, хороший интерфейс экономит время пользователя и доставляет его в нужную точку с минимальными усилиями.
- Снисходительность. Даже при самом продуманном интерфейсе ни один пользователь не застрахован от ошибки. Необходимо продумать заботливые сообщения на случай, если что-то пошло не так. Это поможет сохранить деньги, время и лояльность клиентов в случае сбоя.

Проектирование дизайна экранных форм выполнено в программном продукте Figma – кроссплатформенном бесплатном онлайн-сервисе для проектирования дизайнов и прототипов [32]. Макеты экранных форм представлены на рис. 6 – 12.

Рисунок 6 – Экран авторизации

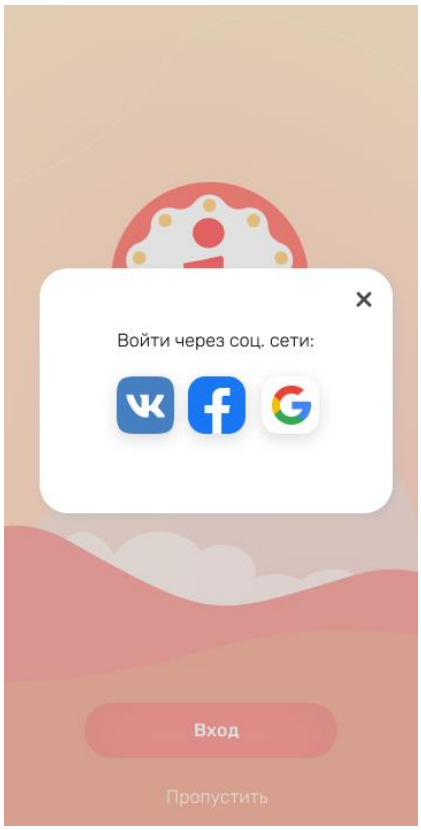


Рисунок 7 – Экран авторизации через социальные сети

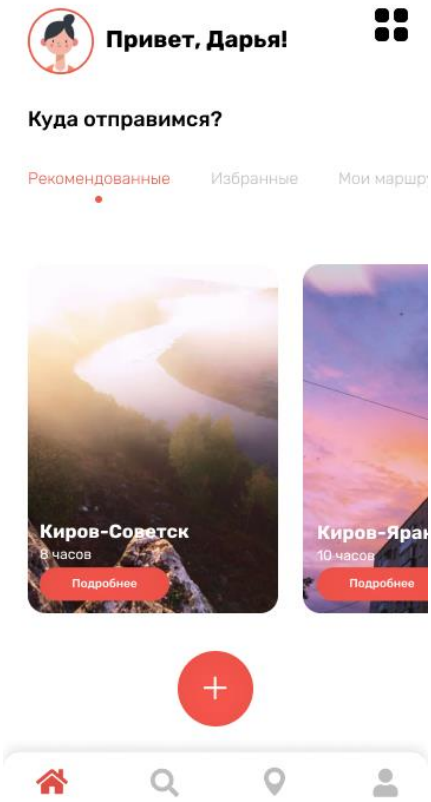


Рисунок 8 – Домашний экран

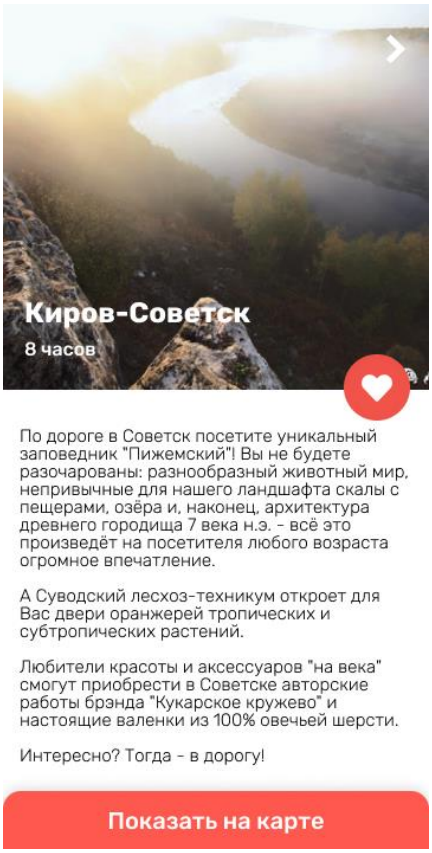


Рисунок 9 – Экран с описанием туров

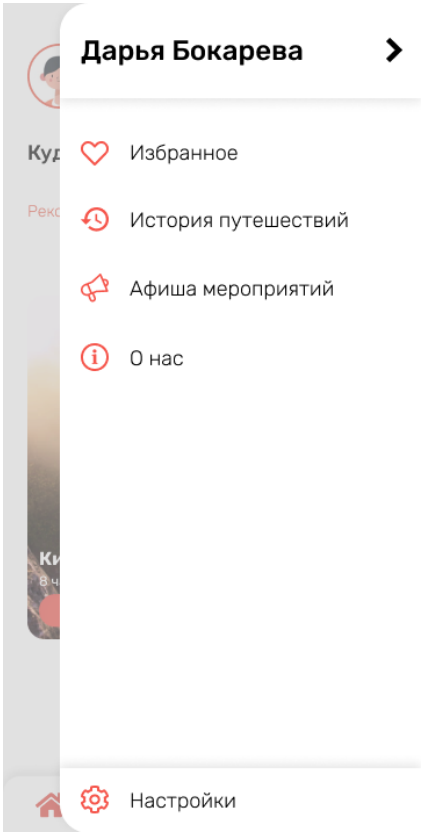


Рисунок 10 – Всплывающее меню

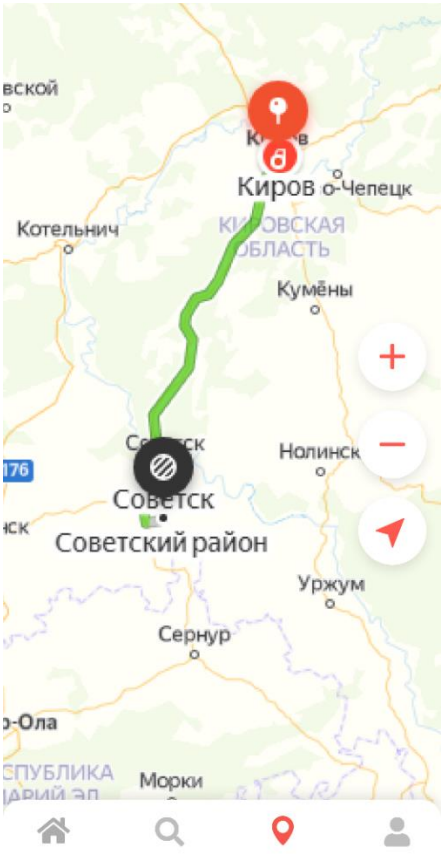


Рисунок 11 – Экран с визуализацией маршрута

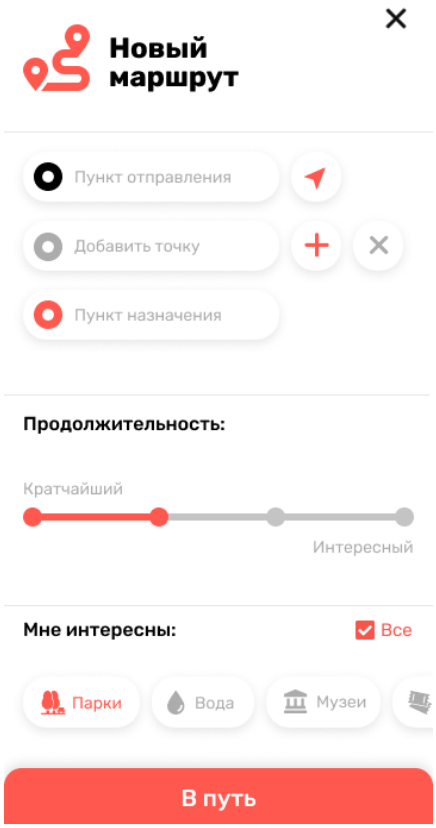


Рисунок 12 – Экран создания персонального маршрута

3.1.4. Выбор средств реализации

В первую очередь, необходимо выбрать наиболее подходящую систему управления базами данных. Для оценки требований к СУБД используется следующая шкала оценок:

1 – неважно;

2 – маловажно;

3 – желательно;

4 – требуется;

5 – обязательно.

В таблице 2 представлены критерии к СУБД с оценкой важности каждого из критериев.

Таблица 2 – Оценка важности требований к СУБД

Критерий	Оценка важности
Моделирование данных:	
Используемая модель данных	5
Возможность применения триггеров и хранимых процедур	3
Средства поиска	2
Предусмотренные типы данных	5
Реализация языка запросов	5
Особенности архитектуры и функциональные возможности:	
Переносимость	4
Масштабируемость	4
Распределенность	2
Сетевые возможности	4
Контроль работы системы:	
Ведение журналов производительности и контроль использования ресурсов системы	2
Имеющийся инструментарий управления и настройки	5
Особенности разработки приложений:	
Средства разработки приложений для своих систем	4
Многоязыковая поддержка	3
Поддерживаемые языки программирования	3
Производительность:	
Рейтинг TPC	3
Возможности оптимизирования запросов	2
Надежность:	
Восстановление после сбоев	5
Резервное копирование	3
Наличие средств управления и контроля доступа пользователей к данным	3
Требования к рабочей среде:	
Поддерживаемые аппаратные платформы	3
Минимальные требования к оборудованию	4
Операционные системы, под управлением которых способна работать СУБД	4
Смешанные критерии:	
Качество и полнота документации	3
Локализованность	3
Стоимость	3
Стабильность производителя	4
Распространенность СУБД	3

Для сравнения выбраны такие СУБД, как Microsoft SQL Server, PostgreSQL, Firebird (табл. 3). Итоговый балл рассчитывается путём умножения оценки важности критерия на трёхбалльную оценку конкретной СУБД.

Таблица 3 – Сравнение СУБД

	MS SQL Server	PostgreSQL	Firebird
Моделирование данных:			
Используемая модель данных	3	3	3
Возможность применения триггеров и процедур	3	2	2
Средства поиска	2	2	1
Предусмотренные типы данных	2	2	2
Реализация языка запросов	3	3	3
Особенности архитектуры и функциональные возможности			
Переносимость	3	2	2
Масштабируемость	2	1	1
Распределенность	2	1	1
Сетевые возможности	3	2	2
Контроль работы системы			
Контроль использования ресурсов системы	3	3	3
Имеющийся инструментарий управления и настройки	3	2	2
Особенности разработки приложений			
Средства разработки приложений для своих систем	3	1	1
Многоязыковая поддержка	2	2	1
Поддерживаемые языки программирования	3	2	2
Производительность			
Рейтинг TPC	2	2	3
Возможности оптимизирования запросов	2	2	2
Надежность			
Восстановление после сбоев	2	2	2
Резервное копирование	2	2	2
Наличие средств управления и контроля доступа пользователей к данным	3	3	3
Требования к рабочей среде			
Поддерживаемые аппаратные платформы	3	3	3
Минимальные требования к оборудованию	2	2	2
Операционные системы, под управлением которых способна работать СУБД	1	3	3
Смешанные критерии			
Качество и полнота документации	3	2	2
Локализованность	3	2	2
Стоимость	3	3	3
Стабильность производителя	3	3	2
Распространенность СУБД	3	3	3
ИТОГО	**241**	**209**	**204**

С учётом итоговых баллов таблицы, а также того факта, что в рамках образовательной программы были получены навыки работы с этой же СУБД – база данных будет реализована с помощью СУБД Microsoft SQL Server [33].

Microsoft SQL Server – это система управления реляционными базами данных, использующая в качестве основного языка запросов – Transact-SQL (T-SQL), разработанный совместно корпорацией Microsoft и Sybase.

В качестве среды управления инфраструктурой SQL Server будет использована SQL Server Management Studio, включающая графическую программу и скриптовый редактор.

Веб-приложение будет реализовано с помощью фреймворка Django [34] на языке программирования Python. Данный язык программирования обладает большим количеством математических модулей, которые понадобятся для реализации алгоритма оптимального построения маршрутов и рекомендательного алгоритма. Django – это фреймворк с открытым исходным кодом, использующийся для веб-приложений на языке Python. Проект на Django использует паттерн проектирования Model-View-Template, являющийся аналогом паттерна проектирования MVC (Model-View-Controller) [35].

Django содержит огромное количество функциональности для решения задач веб-разработки.

Высокоуровневые возможности Django:

- собственный ORM-механизм (Object-Relational Mapping), позволяющий обезопасить сервис от SQL-инъекций;
- миграции базы данных, позволяющие переносить изменения в модели Django на структуру БД (создание базы данных, таблиц и полей, их изменение и удаление), а также являющаяся системой контроля версий для базы данных;

- система аутентификации пользователей, обеспечивающая аккаунты пользователей, группы, права и сессии на основе куки-файлов [36], поддающаяся гибкой настройке под проекты;
- встроенная панель администратора – автоматическое создание части сайта, предназначенной для управления таблицами базы данных, включающая создание, просмотр, обновление и удаление данных.

Мобильное приложение будет реализовано с использованием Flutter SDK [37] на языке программирования Dart. Flutter – это SDK с открытым исходным кодом для создания высокопроизводительных кроссплатформенных мобильных приложений от компании Google. Основное преимущество данного SDK заключается в возможности использования единого языка программирования и единой кодовой базы для разработки мобильного приложения под ОС Android и iOS. Это существенно сокращает время и стоимость разработки мобильных приложений.

Dart – основной язык программирования, используемый для Flutter. Данный язык компилируется в бинарный код, за счёт чего достигается высокая скорость выполнения операций, сравнимая с такими языками, как Java, Kotlin, Swift или Objective-C.

Небольшое сравнение Flutter с нативными приложениями (написанные на «родных» языках ОС, Swift для iOS и Kotlin для Android), а также с конструкторами приложений, представлено на рисунке 11.

Рисунок 11 – Сравнение SDK [38]

Данное сравнение подтверждает, что Flutter обладает всеми преимуществами для разработки мобильных приложений.

Итоговый стек используемых средств разработки:

- MS SQL Server;
- Django (Python);
- Flutter (Dart).

3.2. Разработка веб-приложения

Разработка веб-приложения осуществляется в IDE PyCharm Community Edition 2019.1.3. Это актуальная среда разработки для языка программирования Python, поддерживающая стандарт PEP 8 [39]. Для разработки используется Python версии 3.6. Для создания Django-проекта понадобится установить web-framework Django, что можно сделать прямо из IDE (рис. 13).

После установки фреймворка, необходимо создать проект, прописав в терминале команду «django-admin startproject project_name» (рис. 14). Структура вновь созданного Django-проекта представлена на рисунке 15.

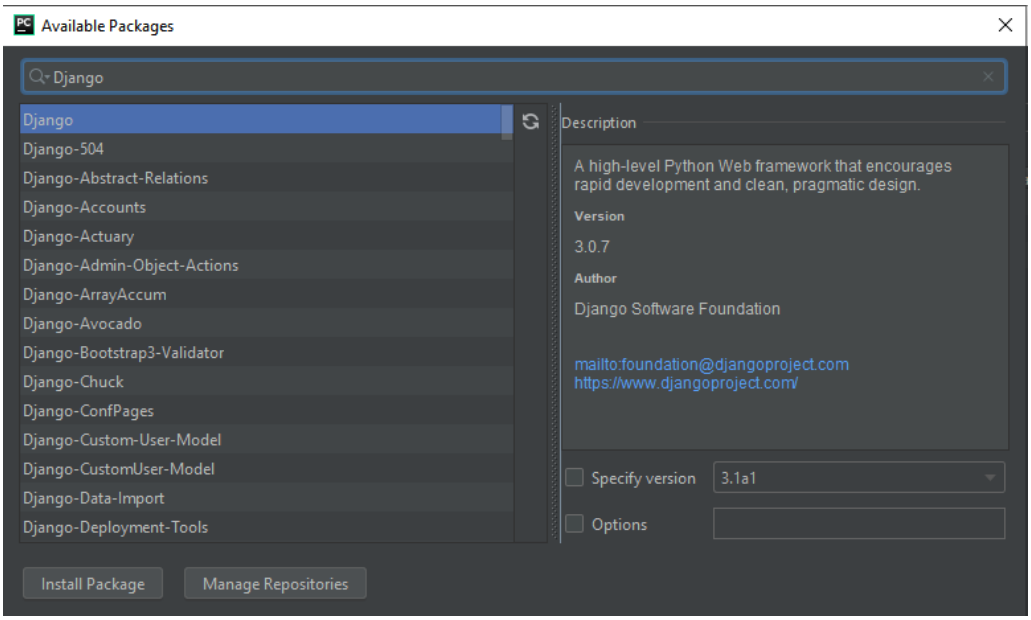


Рисунок 13 – Установка Django

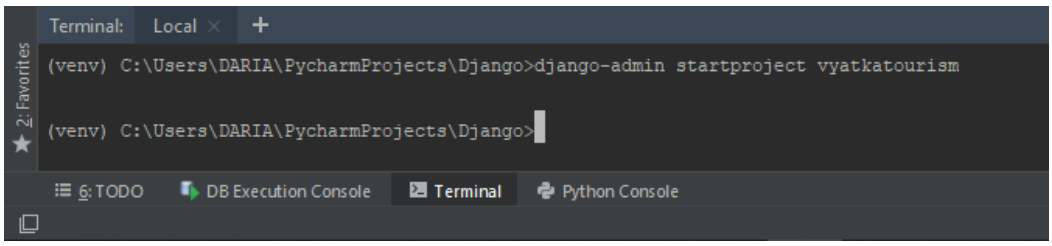


Рисунок 14 – Создание Django-проекта

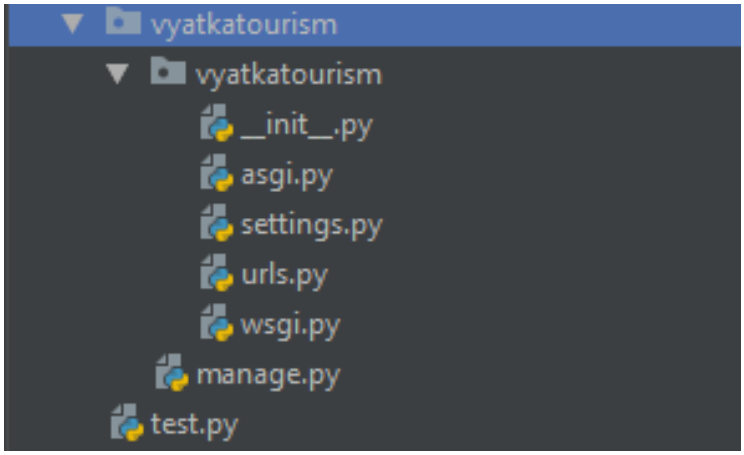


Рисунок 15 – Структура Django-проекта

Django использует модульную систему проекта, когда один проект включает несколько приложений, отвечающих каждое за свой функционал. Первоначальная структура проекта содержит следующие файлы:

- manage.py – скрипт для управления проектом. С помощью этого скрипта создаются приложения, производится работа с базой данных и запуск отладочного сервера;
- __init__.py – классический для Python пустой файл, предназначенный для распознавания текущей папки как модуля, с целью использования содержащихся в ней объектов в других частях проекта;
- settings.py – хранит различные настройки проекта (регистрацию приложений, настройки базе данных и др.).
- urls.py – задаёт ассоциации URL адресов с представлениями.

Остальные файлы являются утилитами, поддерживающими синхронизацию проекта с веб-сервером.

Для веб-сервиса понадобится создать приложение routes, отвечающее за работу с маршрутами. Создание приложений осуществляется командой «python manage.py startapp app_name». После создания приложения, его необходимо зарегистрировать в проекте, чтобы различные утилиты смогли его использовать. Регистрация осуществляется путём добавления названий приложений в список INSTALLED_APPS, находящийся в настройках проекта settings.py (рис. 16). Структура приложения представлена на рисунке 17.

```
33  INSTALLED_APPS = [
34      'django.contrib.admin',
35      'django.contrib.auth',
36      'django.contrib.contenttypes',
37      'django.contrib.sessions',
38      'django.contrib.messages',
39      'django.contrib.staticfiles',
40      'routes',
41  ]
```

Рисунок 16 – Регистрация приложений в проекте

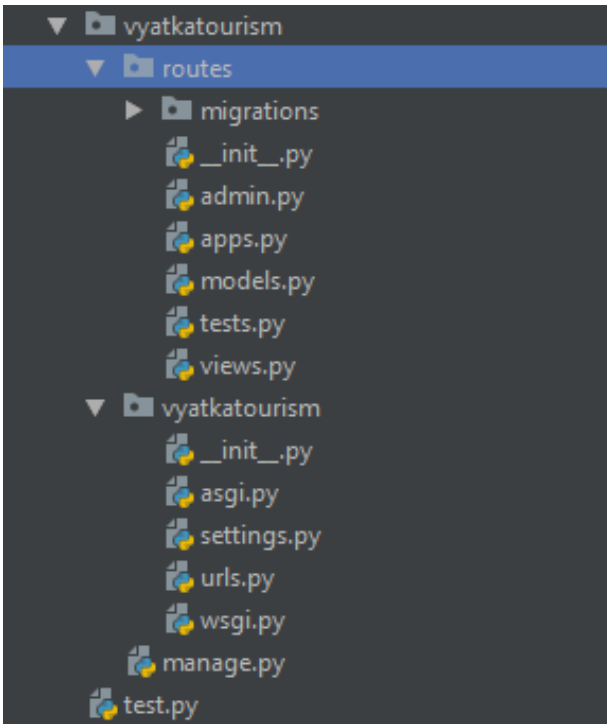


Рисунок 17 – Структура приложения routes

Приложение содержит такие файлы, как admin.py – административные настройки, apps.py – файл, необходимый для регистрации приложения, tests.py – файл для тестов приложения, а также models.py (модели) и views.py (контроллеры). Последние два файла реализуют паттерн проектирования MVT. Как пояснялось выше, Django использует шаблон Model-View-Template, аналогичный классическому паттерну MVC. Схематичное представление архитектуры MVT представлено на рисунке 18.

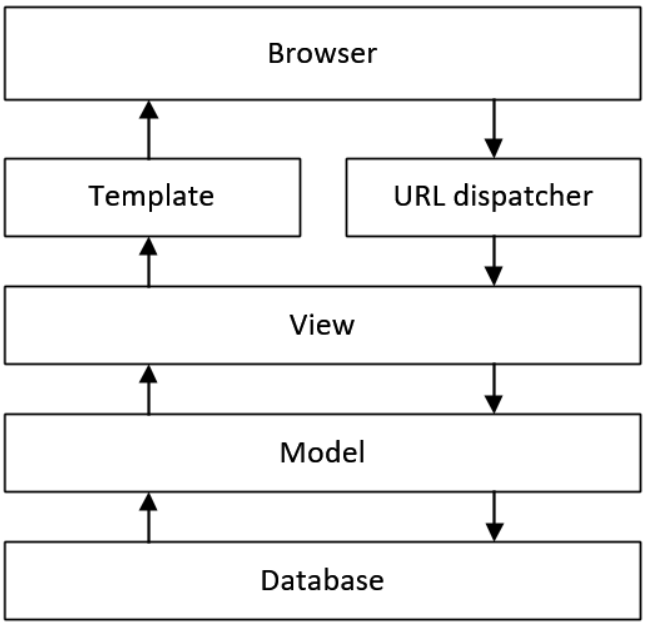


Рисунок 18 – Архитектура MVT

Основные элементы паттерна Model-View-Template [40]:

- URL dispatcher на основании запрошенного адреса URL определяет, какой ресурс должен обрабатывать данный запрос;
- View получает запрос, обрабатывает его и отправляет пользователю некоторый ответ. При необходимости обращения к модели и базе данных, View взаимодействует с ними. Для создания ответа View может применять Template. В архитектуре MVC этому компоненту соответствуют контроллеры;

- Model описывает данные, используемые в приложении. Отдельные классы соответствуют таблицам в базе данных;
- Template представляет логику в виде сгенерированной разметки html. В MVC этому компоненту соответствует View, т. е. представления.

В models.py приложения routes прописывается модель данных (Приложение Б), согласно модели, описанной на рисунке 4.

Для синхронизации модели данных с MS SQL Server для Django необходимо установить модуль django-mssql-backend. Настройка базы данных осуществляется в файле settings.py. Необходимо зарегистрировать приложение 'sql_server.pyodbc', и настроить подключение в переменной DATABASES (рис. 19).

```
78  DATABASES = {
79      'default': {
80          'ENGINE': 'sql_server.pyodbc',
81          'NAME': 'vyatkatourismdb',
82      }
83  }
84
```

Рисунок 19 – Подключение к MS SQL Server

Для того, чтобы создать базу данных, необходимо осуществить миграцию. Миграция создаётся с помощью команды в терминале «python manage.py makemigrations» и осуществляется командой «python manage.py migrate». В этот момент, Django создаёт и запускает SQL-скрипт. Происходит создание базы данных с именем, указанным в настройках (рис. 19) и создание таблиц согласно модели данных (models.py). С помощью SQL Microsoft Management Studio можно увидеть результат создания базы данных (рис. 20).

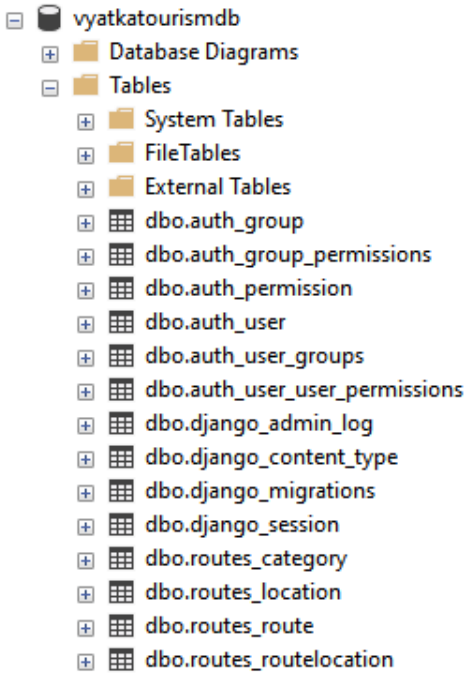


Рисунок 20 – База данных в СУБД MS SQL Server

URL-соотношения хранятся в файле urls.py (рис. 21) в переменной urlpatterns, которая является списком функций path(). Каждая path() функция ассоциирует шаблон URL с контроллером, либо с другим дочерним списком. Список urlpatterns инициализирует список функции, которая, например, соотносит «admin/» с модулем admin.site.urls, содержащим собственное соотношение.

```
6   urlpatterns = [
7       path('', views.index, name='index'),
8       path('admin/', admin.site.urls),
9       path('routes/', routes_views.index)  # include('accounts.urls'))
10  ]
11
```

Рисунок 21 – URL-соотношения

Для обеспечения администратора возможностью управлять таблицами базы данных из веб-приложения, необходимо вывести их в панель администратора. Это делается путём регистрации моделей в файле admin.py (рис. 22).

```python
from django.contrib import admin

from .models import *

admin.site.register(Route)
admin.site.register(RouteLocation)
admin.site.register(Category)
admin.site.register(Location)
admin.site.register(Saved)
admin.site.register(History)
admin.site.register(AuthorizationType)
```

Рисунок 22 – Регистрация моделей в панели администратора

Запуск веб-приложения осуществляется на встроенном в Django сервере runserver командой «python manage.py runserver». По умолчанию запуск производится по адресу 127.0.0.1:8000. Панель администратора представлена на рисунке 23.

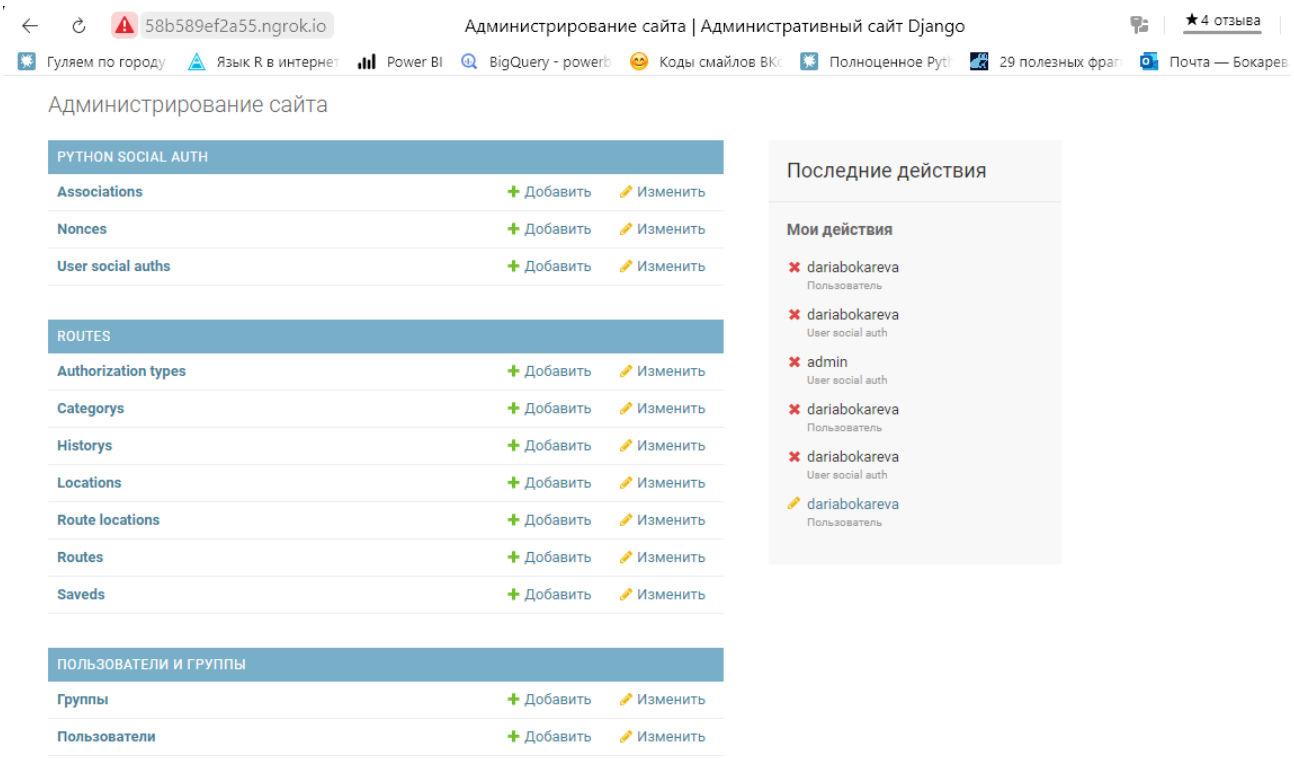


Рисунок 23 – Панель администратора

Соединение базы данных и веб-приложения осуществляется с помощью T-SQL запросов. Передача данных между другими сервисами производится при помощи обмена JSON файлами POST/GET запросами.

Чтобы настроить соединение с сервисами необходимы дополнительные модули. Например, авторизация через социальные сети, осуществляется при помощи модуля social-auth-app-django. После установки его необходимо зарегистрировать в списке приложений проекта, а также осуществить миграцию. Данный модуль создаёт дополнительные таблицы в базе данных, для хранения авторизованных через социальные сети пользователей и их привязке к общему списку пользователей. Чтобы реализовать авторизацию при помощи социальной сети ВКонтакте, необходимо создать приложение (рис. 24) и получить ключи доступа, которые прописываются в настройках проекта. Аналогично реализуется авторизация через другие соц. сети.

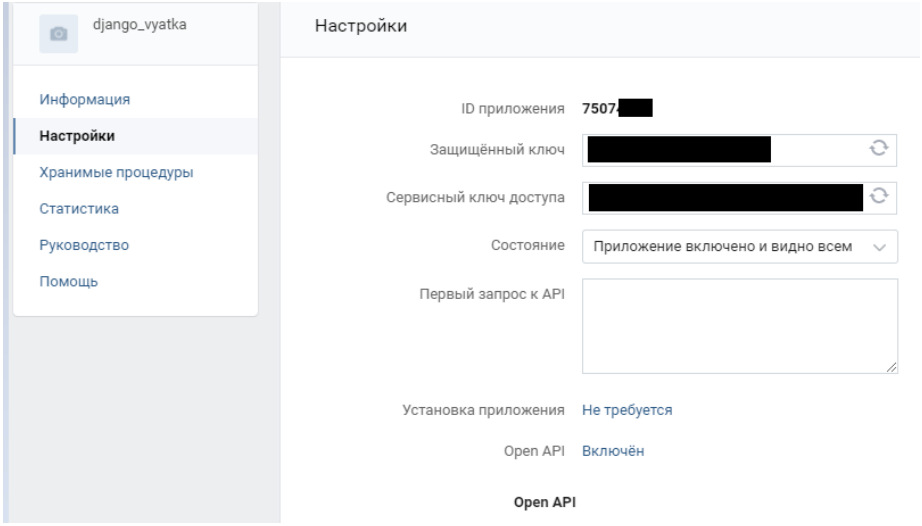


Рисунок 24 – Приложение для авторизации через VK

В urls.py необходимо прописать путь к URL адресам модуля авторизации. Для отладки веб-приложения необходимо заменить адрес localhost:8000 на общедоступный домен. Программа Ngrok позволяет сделать это командой «ngrok http 8000» (рис. 25).

Рисунок 25 – Замена адреса программой Ngrok

Теперь любой пользователь, обратившийся по адресу https://48c1265e7e33.ngrok.io/login/vk-oauth2/ сможет авторизоваться в веб-приложении, используя аккаунт социальной сети (рис. 26). После авторизация пользователь записывается в таблицу «User social auth» и в таблицу «User» (рис. 27, 28) с привязкой по UID.

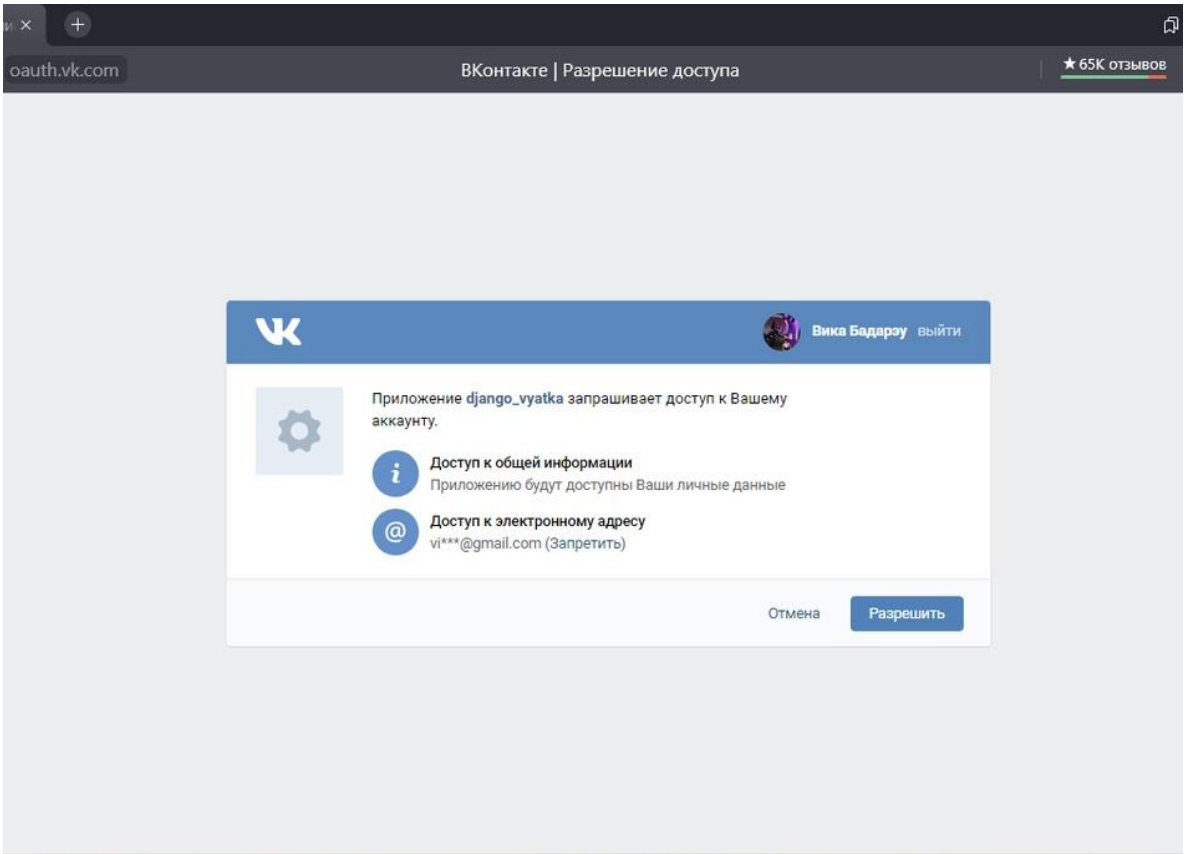


Рисунок 26 – Авторизация пользователя

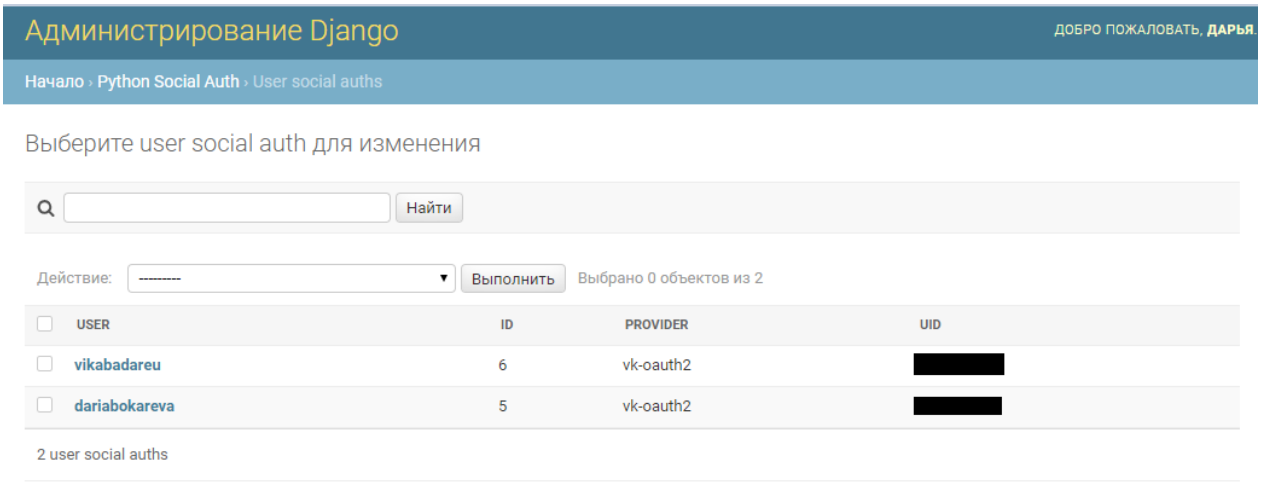


Рисунок 27 – Таблица User social auth

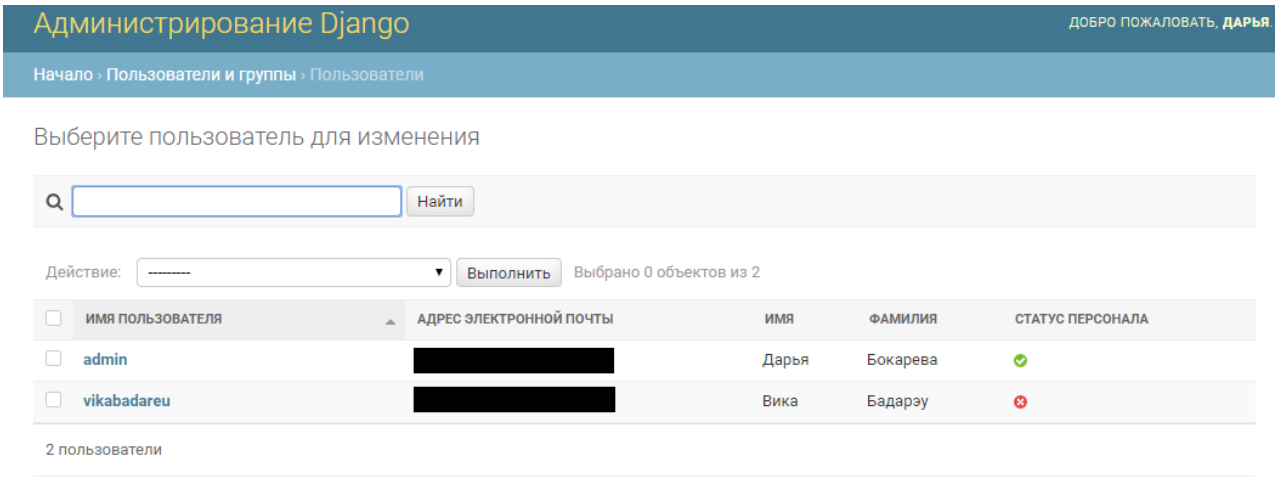


Рисунок 28 – Таблица User

Обработка информации, которая будет поступать от пользователя из мобильного приложения, осуществляется при помощи API нашего приложения. Так, чтобы реализовать построение маршрута, параметры для алгоритма, указанные пользователем, передаются в виде JSON запроса (рис. 29) по адресу «/aco_tsp», по которому происходит выполнение метода AntColony(). Результаты алгоритма в дальнейшем передаются JSON файлом на сервер Яндекса с помощью Yandex MapKit API [41] и возвращаются на устройство

пользователя. Обмен данными для реализации рекомендательного алгоритма происходит между сервером и клиентом.

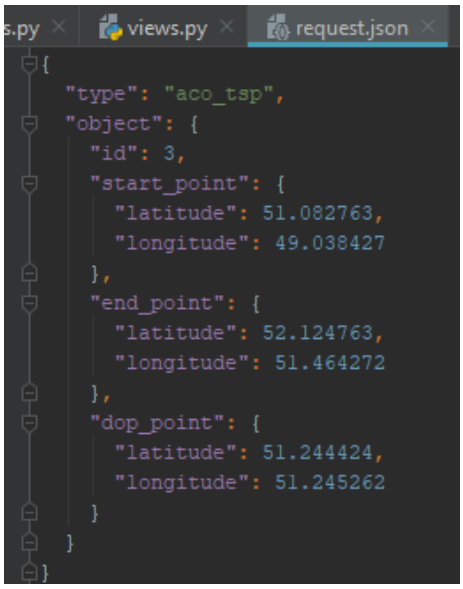


Рисунок 29 – JSON-запрос

Программный код модифицированного алгоритма ACO и рекомендательного алгоритма представлен в Приложении В и Приложении Г, соответственно.

3.3. Разработка мобильного приложения

Разработка мобильного приложения осуществлялась в IDE Visual Studio Code, с использованием Flutter SDK и языка программирования Dart. На рисунке 30 представлена общая структура проекта.

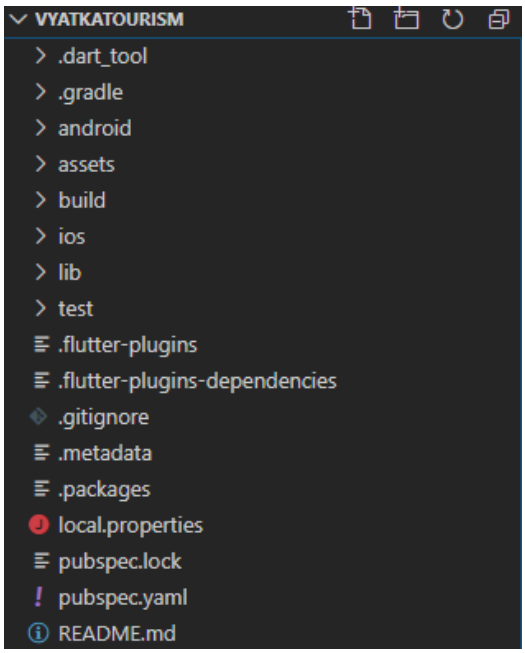


Рисунок 30 – Структура проекта мобильного приложения

Основным файлом, хранящим все подключенные пакеты, пути, настройки и версии сборки приложения во фреймворке Flutter является файл pubspec.yaml (рис. 30).

Основными компонентами проекта, с которыми происходит взаимодействие при разработке приложения, являются следующие компоненты (рис. 31, 32):

- assets – хранит все ресурсы приложения (иконки, шрифты, изображения и др.)
- lib – хранит файлы расширения .dart. Сюда входят экранные формы, дополнительные виджеты и утилиты. Инициализирующим файлом является main.dart.

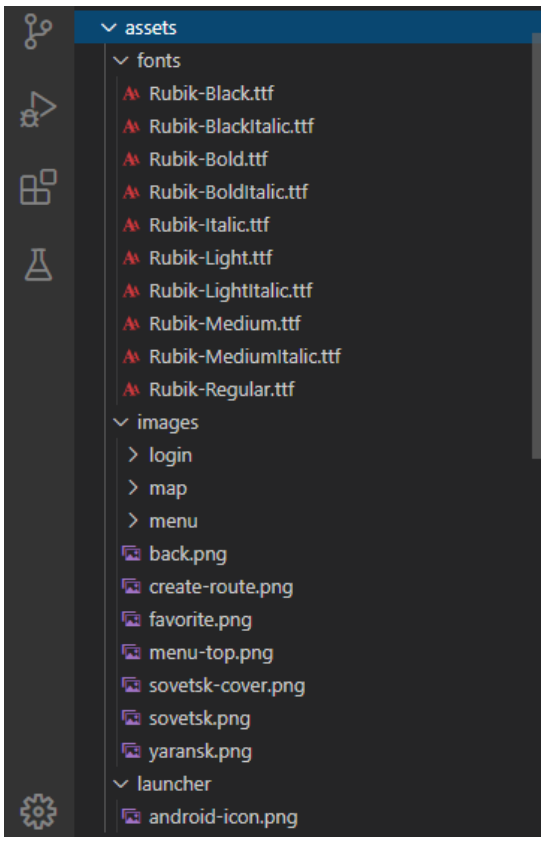


Рисунок 31 – Структура компонента assets

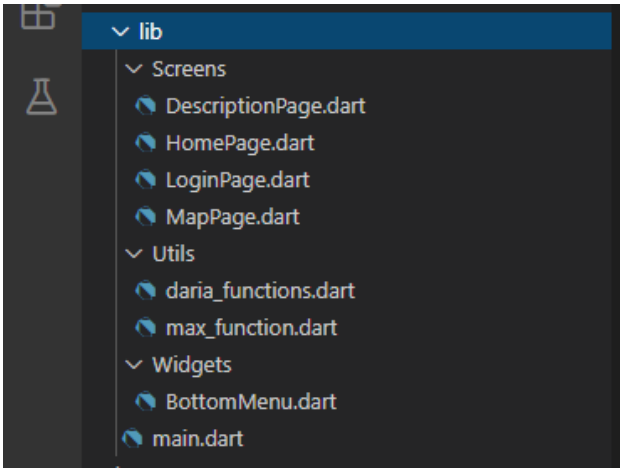


Рисунок 32 – Структура компонента lib

Функциональность мобильного приложения включает CRUD функции для работы пользователя с локальными данными и методы работы с картой. Реализация CRUD подхода наиболее функциональна с использованием Flutter, так как данная библиотека создавалась для реализации данного подхода. Расширение базовых функциональных возможностей работы приложения

производится через системы плагинов. В методологии языка Dart плагин называется package. Плагин представляет собой инструмент реализации дополнительного функционала работы с Flutter SDK. В проекте мобильного приложения используются плагины, представленные в таблице 4.

Таблица 4 – Список плагинов, используемых в мобильном приложении

№	Наименование	Назначение
1	flutter_bloc	Работа с потоками
2	shared_preferences	Организация хранения данных на сервере
3	hive	Работа с БД
4	equatable	Функции сравнения объектов
5	url_launcher	Запуск ссылок
6	link_text	Виджет ссылок в тексте
7	flutter_launcher_icons	Виджет настраиваемой иконки закуска приложения
8	permission_handler	Работа с разрешениями
9	geolocator	Работа с координатами
10	http	Работа с сетью
11	convert	Преобразование данных
12	Provider	Асинхронная работа с данными
13	path_provider	Работа с путями
14	async	Асинхронные функции работы
15	io	Потоки ввода и вывода данных
16	flutter_slidable	Настаиваемый плагин работы списков
17	json_serializable	Функции работы с json
18	connectivity	Функции работы с сетью
19	extended_image	Настраиваемый плагин работы с изображениями

Существующий плагин работы с картами через API MapKit yandex_mapkit не поддерживает функции построения маршрутов, через указанные точки, поэтому логичным решением является дополнить его нужными функциональными возможностями. Все плагины являются открытыми и публикуются в GitHub репозиториях [48]. Такой подход позволяет сделать

ответвление от заданной кодовой базы плагина и в дальнейшем поддерживать его функциональность самостоятельно. На момент разработки выбрана последняя версия плагина 0.3.8.

На его основе был создан плагин yandex_mapkit_navigation (рис. 33), в который были добавлены функции работы с точками и маршрутами на карте. Данный функционал был реализован как обертка над модулем Direction для Yandex Mapkit для Android. В файле основного проекта мобильного приложения pubspec.yaml был прописан путь к проекту yandex_mapkit_navigation.

Подключение к API осуществляется при помощи токена, полученного от сервиса Яндекс.Карты.

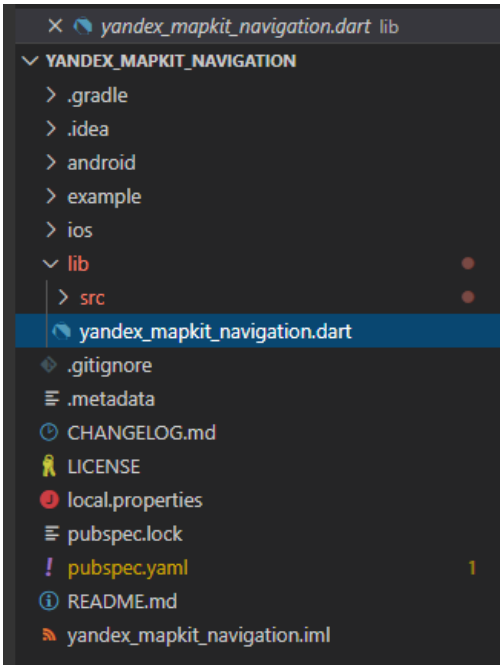


Рисунок 33 – Структура проекта пакета yandex_mapkit_navigation

Примеры исполняемых экранных форм с выводом маршрута на карту и афишей мероприятий представлены на рисунках 34, 35.

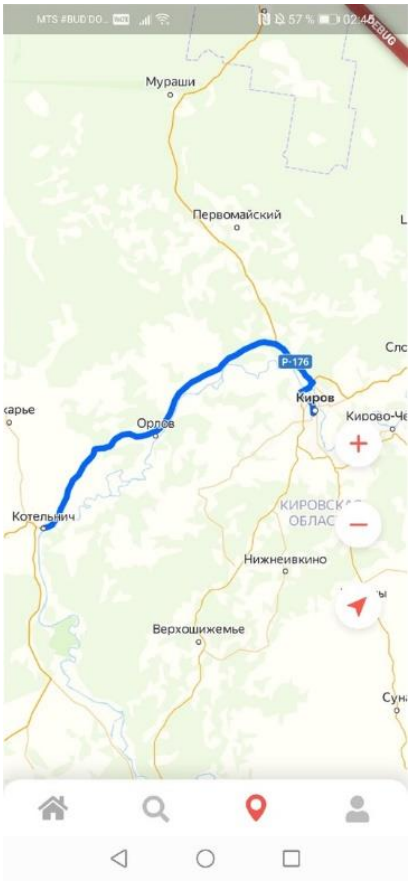


Рисунок 34 – Экранная форма вывода маршрута в режиме отладки

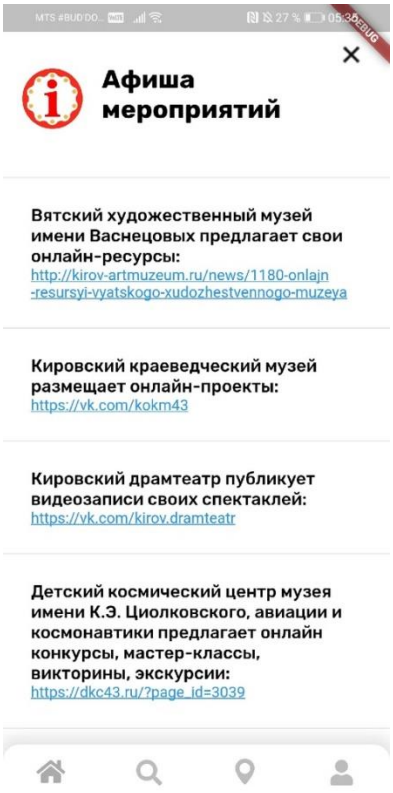


Рисунок 35 – Экранная форма афиши мероприятий в режиме отладки

3.4. Выводы

В данной главе в виде UML диаграмм были формализованы требования к функционалу Сервиса, обозначенные в техническом задании, спроектирована архитектура Сервиса и база данных. Представлена структура экранных форм клиентской части и спроектирован дизайн пользовательского интерфейса.

Проведен анализ и сравнение возможных средств реализации и выбраны наиболее подходящие из них.

Разработано веб-приложение, отвечающее за бизнес-логику сервиса и управление базой данных. Реализованы в виде программного кода алгоритмы, описанные во второй главе.

Разработано мобильное приложение, содержащее основной функционал согласно ТЗ.

4. Оценка экономической эффективности

4.1. Технико-экономическое обоснование проекта

Целью выпускной квалификационной работы является разработка сервиса, позволяющего расширить каналы информирования потребителей «Центра развития туризма Кировской области».

Внедрение Сервиса приведёт к следующим положительным эффектам:

1) снижение трудозатрат на услуги консультирования;
2) повышение качества предоставляемых Центром развития туризма туристских услуг;
3) снижение расходов на создание и распространение печатной продукции, при частичном переносе информации с печатной продукции в функционал приложения;
4) повышение туристического потока на афишируемые организацией мероприятия;
5) повышение количества упоминаний событий, связанных с организацией, и самой организации в социальных медиа;
6) увеличение въездного потока туристов на территорию Кировской области, в т. ч. увеличение доходов региона с туристической отрасли;
7) повышение качества и доступности внутреннего туризма на территории Кировской области.

Разрабатываемый Сервис направлен на поддержание текущей деятельности Центра развития туризма. В первоначально поставленных целях нет цели получения прямой прибыли с использования приложения. Однако разработка и сопровождение продукта требуют затрат, которые необходимо рассчитать.

4.2. Расчет затрат на разработку сервиса

Затраты на разработку Сервиса рассчитываются по следующей формуле [42]:

$$К_{РПР} = З_{ФОТР} + З_{ОВФ} + З_{ЭВМ} + З_{СПП} + З_{ХОН} + Р_{Н} \quad (22)$$

где:

$З_{ФОТР}$ – общий фонд оплаты труда разработчиков;

$З_{ОВФ}$ – начисления на заработную плату разработчиков во внебюджетные фонды;

$З_{ЭВМ}$ – затраты, связанные с эксплуатацией техники;

$З_{СПП}$ – затраты на специальные программные продукты, необходимые для разработки Сервиса;

$З_{ХОН}$ – затраты на хозяйственно-операционные нужды;

$Р_{Н}$ – накладные расходы ($Р_{Н}$ = 10% от $З_{ФОТР}$).

При разработке программного продукта общее время разработки составило 4 месяца. Из них машинное время (непосредственная работа с вычислительной и оргтехникой) составляет 3 мес.

Фонд оплаты труда за время работы над программным продуктом:

$$З_{ФОТР} = \sum_{j=1}^{m} О_{Pj} \cdot Т_{РПР_j} \cdot (1 + k_{Д})(1 + k_{У}), \quad (23)$$

Где:

$О_{Pj}$ – оклад j-го разработчика. В разработке участвовал 1 человек, его оклад составляет 20 000 руб.;

$Т_{РПРj}$ – общее время работы над ПР в месяцах;

$k_{Д}$ – коэффициент дополнительной зарплаты, $k_{Д} = 0$;

$k_{У}$ – районный коэффициент, $k_{У} = 0$.

Таким образом,

$З_{ФОТР}$ = 20 000 * 4 * (1 + 0) * (1 + 0) = 80 000 руб.

Отчисления во внебюджетные фонды складываются из ЕСН и взносов на обязательное социальное страхование от несчастных случаев на производстве и профессиональных заболеваний.

Годовой фонд заработной платы разработчика не превышает 865 тыс. руб., поэтому используются максимальные ставки ЕСН. Ставка страхования от

несчастных случаев в соответствии с классом профессионального риска составляет 0,2% [43]. Значения всех используемых ставок приведены в таб. 5.

Таблица 5 – Значения налоговых ставок

№	Наименование внебюджетного фонда	Размер ставок, %
2	Пенсионный фонд	22
3	Фонд социального страхования	2,9
4	Федеральный фонд обязательного медицинского страхования	5,1
6	Страховой тариф на обязательное социальное страхование от несчастных случаев на производстве и профзаболеваний (для 2 класса профессионального риска)	0,2
	ИТОГО	30,2

Сумма начислений на заработную плату во внебюджетные фонды составляет:

$$З_{ОВФ} = 0{,}302 \cdot З_{ФОТР}, \quad (24)$$

$$З_{ОВФ} = 0{,}302 \cdot 80\,000 = 24\,160 \text{ руб.}$$

Затраты, связанные с использованием вычислительной и оргтехники:

$$З_{ЭВМ} = T_{МРПР} \cdot k_Г \cdot n \cdot C_{М-Ч}, \quad (25)$$

где:

k_r – коэффициент готовности ЭВМ, $k_Г = 0{,}95$;

n – количество единиц техники, равно 2;

$C_{М-Ч}$ – себестоимость машиночаса, $C_{М-Ч} = 8$ руб.;

$T_{МРПР}$ – машинное время работы над программным продуктом, равно 4 мес.

Перевод рабочего времени в часы осуществляется по формуле:

$$T_{час} = T_{мес} \cdot Ч_{РД} \cdot T_{см} \cdot K_{см}, \quad (26)$$

где:

$Т_{час}$ – рабочее время, ч;

$Т_{мес}$ – рабочее время, мес., ($Т_{мес}$ = 3);

$Ч_{РД}$ – число рабочих дней, ($Ч_{РД}$ = 22);

$Т_{см}$ – продолжительность рабочей смены, ($Т_{см}$ = 8 ч);

$К_{см}$ – количество рабочих смен, ($К_{см}$ = 1).

Таким образом, время на разработку Сервиса с использованием ЭВМ составляет:

$$Т_{час} = 3 \cdot 22 \cdot 8 \cdot 1 = 528 \text{ часов},$$

$$З_{ЭВМ} = 528 \cdot 0{,}95 \cdot 1 \cdot 8 = 4\,013 \text{ руб.}$$

Затраты на специальные программные продукты, необходимые для разработки Сервиса рассчитываются по формуле:

$$З_{СПП} = \sum_{\rho=1}^{n} Ц_{\rho}\ , \tag{27}$$

где:

$Ц_{\rho}$ – цена ρ-го специального программного продукта.

Специальные программные продукты были предоставлены бесплатно:

$З_{СПП} = 0$.

Затраты на хозяйственно-организационные нужды вычисляются по формуле [42]:

$$З_{ХОН} = \sum_{\tau=1}^{n} Ц_{\tau} \cdot К_{\tau}\ , \tag{28}$$

где:

$Ц_{\tau}$ – цена τ-го товара, руб.;

$К_{\tau}$ – количество τ-го товара.

$$З_{ХОН} = 0.$$

Накладные расходы:

$$Р_{Н} = З_{ФОТР} \times k_{НР}\ , \tag{29}$$

$$P_H = 80\,000 \cdot 0{,}1 = 8\,000 \text{ руб.}$$

Таким образом, затраты на разработку программного продукта составят:

$$З_{РПР} = 80\,000 + 24\,160 + 4\,013 + 8\,000 = 116\,173 \text{ руб.}$$

4.3. Расчет затрат на внедрение и сопровождение сервиса

Затраты на внедрение программного продукта ($З_{ВПР}$) рассчитываются по формуле [44]:

$$З_{ВПР} = З_М + З_{КТС} \times (1 + k_{ТУН}) + З_{ПО} + З_{ФОТВ} + З_{ОВФ} + З_{ЭВМ} + Р_{ком} + Р_Н, \quad (30)$$

где:

- $З_М$ – затраты на приобретение материалов, руб.;
- $З_{КТС}$ – затраты на приобретение комплекса технических средств, руб.;
- $З_{ПО}$ – затраты на приобретение программного обеспечения, руб.;
- $З_{ФОТВ}$ – затраты на оплату труда работников, занятых внедрением проекта, руб.;
- $З_{ОВФ}$ – отчисления во внебюджетные фонды с заработной платы работников, занятых внедрением проекта, руб.;
- $З_{ЭВМ}$ – затраты, связанные с эксплуатацией ЭВМ при внедрении проектного решения, руб.;
- $Р_{ком}$ – командировочные расходы, руб.;
- $Р_Н$ – накладные расходы, руб.;

$k_{ТУН}$ – коэффициент транспортирования, установки и наладки комплекса технических средств, определяется действующими нормативами организации, а также спецификой конкретного проекта.

Так как для внедрения Сервиса расходных материалов не требуется, то $З_М$ равен нулю.

Серверная часть продукта будет размещена на двух облачных серверах AWS [45], мобильное приложение в сервисе цифровой дистрибуции Google Play. Затраты на покупку серверов и размещения приложения в Google Play войдут в $З_{КТС}$. Стоимость одного сервера 2730 руб./мес. Стоимость кабинета

разработчика Google Play составляет 25$ или 1755 руб. по курсу на текущее время.

$$З_{КТС} = 2730 \cdot 2 + 1755 = 7\ 215 \text{ руб.}$$

Затраты на приобретение программного обеспечения в данном случае равны затратам на разработку и составляют $З_{ПО}$ = 116 173 руб.,

Внедрением занимается один сотрудник с окладом 20 000 руб. Время внедрения – 1 месяц.

$$З_{ФОТВ} = 20\ 000 \text{ руб.}$$

$$З_{ОВФ} = 6\ 040 \text{ руб.}$$

Затраты, связанные с эксплуатацией ЭВМ при внедрении проектного решения составят:

$$З_{ЭВМ} = 1 \cdot 22 \cdot 8 \cdot 8 = 1\ 408 \text{ руб.}$$

Командировочные расходы при внедрении программного продукта не планируются, следовательно, $Р_{ком} = 0$. Так как коэффициент накладных расходов по данным организации составляет $k_{НР}$ =0,1, величина накладных расходов составляет 2 000 руб.

Суммарные затраты на внедрение составят:

$$З_{ВПР} = 7\ 215 + 116\ 173 + 20\ 000 + 6\ 040 + 1\ 408 + 2\ 000 = 152\ 836 \text{ руб.}$$

Ежемесячные затраты на сопровождение Сервиса рассчитываются по формуле:

$$З_{СПР} = З_{КТС\ (мес.)} + З_{ФОТР} + З_{ОВФ} + З_{ЭВМ} + Р_{Н}. \quad (31)$$

Затраты на оплату труда и отчисления рассчитываются для разработчика, занятого на полставки:

$$З_{ФОТВ} = 10\ 000 \text{ руб.}$$

$$З_{ОВФ} = 3\ 020 \text{ руб.}$$

Ежемесячная плата за серверы составляет:

$$З_{КТС} = 2730 \cdot 2 = 5\ 460 \text{ руб.}$$

Затраты на ЭВМ с учётом четырёхчасовой загруженности:

$$З_{ЭВМ} = 1 \cdot 22 \cdot 4 \cdot 8 = 704 \text{ руб.}$$

Накладные расходы составят 1 000 руб.

Суммарные затраты на сопровождение составят:

$$З_{СПР} = 5\,460 + 10\,000 + 3\,020 + 704 + 1\,000 = 20\,184 \text{ руб.}$$

Таким образом, внедрение Сервиса будет стоить 152 836 рублей, а затраты на ежемесячное сопровождение составят 20 184 рублей.

4.4. Выводы

Несомненно, разработка, внедрение и сопровождение подобного Сервиса требуют понести определённых затрат, рассчитанных в данной главе. Однако Сервис является необходимым поддерживающим средством для текущей деятельности предприятия, позволяющим решить её насущные проблемы.

Кроме того, данный программный продукт способен оказать положительный эффект на массу показателей, влияющих не только на организацию, но и в целом на туристский потенциал Кировской области. Данные показатели были рассмотрены в текущей главе в качестве обоснования актуальности разработки Сервиса.

5. Безопасность жизнедеятельности

Работа с программным продуктом, разработанным в выпускной квалификационной работе осуществляется со стороны организации с помощью персонального компьютера требует нахождение на рабочем месте оператора ПК. В данной главе необходимо определить требования к безопасности и охране труда при работе на компьютере.

Безопасность жизнедеятельности (БЖД) - это комплекс мероприятий, направленных на обеспечение безопасности человека в среде обитания, сохранение его здоровья, разработку методов и средств защиты путем снижения влияния вредных и опасных факторов до допустимых значений, выработку мер по ограничению ущерба в ликвидации последствий чрезвычайных ситуаций мирного и военного времени.

5.1. Общие требования безопасности

К самостоятельной работе на персональном компьютере допускаются лица, прошедшие предварительный медицинский осмотр. К непосредственной работе с персональным компьютером допускаются лица, не имеющие медицинских противопоказаний.

При работе с персональным компьютером на сотрудников могут оказывать неблагоприятное воздействие следующие опасные и вредные производственные факторы:

- повышенный уровень электромагнитных излучений;
- повышенный уровень ионизирующих излучений;
- повышенный уровень статического электричества;
- повышенная напряженность электростатического поля;
- повышенная или пониженная ионизация воздуха;
- повышенная яркость света;
- перенапряжение зрительного анализатора;
- умственное перенапряжение;
- эмоциональные перегрузки.

Работа с компьютером характеризуется значительным умственным напряжением и нервно-эмоциональной нагрузкой операторов, высокой напряженностью зрительной работы и достаточно большой нагрузкой на мышцы рук при работе с клавиатурой.

В процессе работы с компьютером необходимо соблюдать правильный режим труда и отдыха. В противном случае у персонала отмечаются значительное напряжение зрительного аппарата с появлением жалоб на неудовлетворенность работой, головные боли, раздражительность, нарушение сна, усталость и болезненные ощущения в глазах, в пояснице, в области шеи и руках.

Площадь на одно рабочее место с персональным компьютером для взрослых пользователей должны составлять не менее 6 $м^2$, а объем—не менее 20 $м^3$.

Для повышения влажности воздуха в помещениях с персональным компьютером следует применять увлажнители воздуха, заправляемые ежедневно дистиллированной или кипяченой питьевой водой.

Запрещается проводить ремонт персональных компьютеров непосредственно в рабочих помещениях.

Рабочие места с персональным компьютером по отношению к световым проемам должны располагаться так, чтобы естественный свет падал сбоку, преимущественно слева.

Схемы размещения рабочих мест с персональным компьютером должны учитывать расстояния между рабочими столами, которое должно быть не менее 2 м, а расстояние между боковыми поверхностями видео мониторов - не менее 1,2 м.

Рабочий стул (кресло) должен быть подъемно-поворотным и регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья.

Экран видеомонитора должен находиться от глаз пользователя на оптимальном расстоянии - 600 - 700 мм, но не ближе 500 мм, с учетом размеров алфавитно-цифровых знаков и символов.

В помещениях с персональным компьютером ежедневно должна проводиться влажная уборка.

Помещения с персональным компьютером должны быть оснащены аптечкой первой помощи и углекислотными огнетушителями.

Продолжительность непрерывной работы с персональным компьютером без регламентированных перерывов не должна превышать 2 часов.

Во время регламентированных перерывов с целью снижения нервно-эмоционального напряжения, утомления зрительного анализатора, устранения влияния гиподинамии и гипокинезии целесообразно выполнять комплексы специальных упражнений.

В случаях возникновения у работающих с персональным компьютером зрительного дискомфорта и других неблагоприятных субъективных ощущений, несмотря на соблюдение санитарно-гигиенических, эргономических требований, режимов труда и отдыха следует применять индивидуальный подход в ограничении времени работ с персональным компьютером, коррекцию длительности перерывов для отдыха или проводить смену деятельности на другую, не связанную с использованием персонального компьютера.

5.2. Общие требования безопасности перед началом работы

Перед началом работы оператор обязан:

- осмотреть и привести в порядок рабочее место;
- отрегулировать освещенность на рабочем месте, убедиться в достаточной освещенности, отсутствии отражений на экране, отсутствии встречного светового потока;
- проверить правильность подключения оборудования в электросеть;
- убедиться в наличии защитного заземления и подключения экранного проводника к корпусу процессора;

- протереть специальной салфеткой поверхность экрана и защитного фильтра;
- убедиться в отсутствии дискет в дисководах процессора персонального компьютера;
- проверить правильность установки стола, стула, подставки для ног, пюпитра, положения оборудования, угла наклона экрана, положения клавиатуры и (при необходимости) произвести регулировку рабочего стола и кресла, а также расположение элементов компьютера в целях исключения неудобных поз, длительных напряжений в соответствии с требованиями эргономики.

При включении компьютера оператор обязан соблюдать следующую последовательность включения оборудования:

- включить блок питания;
- включить периферийные устройства (принтер, монитор, сканер и др.);
- включить системный блок (процессор).

Оператору запрещается приступать к работе при:

- отсутствии информации о результатах аттестации условий труда на данном рабочем месте или при наличии информации о несоответствии параметров данного оборудования требованиям Санитарных норм;
- отсутствии защитного экранного фильтра класса «полная защита»;
- отключенном заземляющем проводнике защитного фильтра;
- обнаружении неисправности оборудования;
- отсутствии защитного заземления устройств ПЭВМ;
- отсутствии углекислотного или порошкового огнетушителя и аптечки первой помощи.

Для уменьшения воздействия вредных факторов рекомендуется:

- подготовить рабочее место так, чтобы исключить неудобные позы и длительные напряжения;
- исключить блики на экране;

- не пользоваться люминесцентными лампами, если Вы замечаете их мигание;
- стена или какая-либо поверхность позади дисплея должна быть освещена примерно так же, как экран;
- предпочтительнее использовать жидкокристаллический дисплей;
- установить фильтр на экран и заземлить его;
- расстояние от расположенных рядом ПЭВМ должно быть не менее 1,2 м;
- центр изображения на дисплее должен находиться на высоте 0,7 -1,2 м от уровня пола.

Рекомендуется оборудовать рабочее место пюпитром для расположения документов и подставкой для отдыха рук. Осмотреть рабочее место и убрать посторонние предметы.

5.3. Требования безопасности во время работы

Оператор во время работы обязан:

- выполнять только ту работу, которая ему была поручена и по которой он проинструктирован;
- содержать в порядке и чистоте рабочее место;
- держать открытыми все вентиляционные отверстия устройств;
- внешнее устройство «мышь» применять только при наличии специального коврика;
- при необходимости прекращения работы на некоторое время корректно закрыть все активные задачи;
- отключать питание только в том случае, если оператор во время перерыва в работе на компьютере вынужден находиться в непосредственной близости от видеотерминала (менее 2 метров), в противном случае питание разрешается не отключать;
- выполнять Санитарные нормы и соблюдать режимы работы и отдыха;
- соблюдать правила эксплуатации вычислительной техники в соответствии с инструкциями по эксплуатации;

- при работе с текстовой информацией выбрать наиболее физиологичный режим представления черных символов на белом фоне;
- соблюдать установленные режимом рабочего времени регламентированные перерывы в работе и выполнять в физкультпаузах и в физкультминутках рекомендованные упражнения для глаз, шеи, рук, туловища, ног;
- соблюдать расстояние от глаз до экрана в пределах 60-80 см.

Оператору во время работы запрещается:

- касаться одновременно экрана монитора и клавиатуры;
- прикасаться к задней панели системного блока (процессора) при включенном питании;
- переключать разъемы интерфейсных кабелей периферийных устройств при включенном питании;
- загромождать верхние панели устройств бумагами и посторонними предметами;
- допускать захламленность рабочего места бумагой в целях недопущения накапливания органической пыли;
- производить отключение питания во время выполнения активной задачи;
- производить частые переключения питания;
- допускать попадание влаги на поверхность системного блока (процессора), монитора, рабочую поверхность клавиатуры, дисководов, принтеров и др. устройств;
- включать сильно охлажденное (принесенное с улицы в зимнее время) оборудование;
- производить самостоятельно вскрытие и ремонт оборудования;
- превышать величину количества обрабатываемых символов свыше 30 тыс. за 4 часа работы.

Одним из оптимальных режимов работы является следующий: 40-45 мин. работы на компьютере и 15-20 мин. перерыв. При постоянной работе

экран должен находиться в центре поля обзора, документы должны располагаться слева на столе или на пюпитре, в одной плоскости с экраном.

5.4. Требования безопасности в аварийных ситуациях

Оператор обязан:

- во всех случаях обнаружения обрывов проводов питания, неисправности заземления и других повреждениях электрооборудования, появления запаха гари немедленно отключить питание и сообщить об аварийной ситуации руководителю и дежурному электрику;
- при обнаружении человека, попавшего под напряжение, немедленно освободить пострадавшего от действия тока путем отключения электропитания и до прибытия врача оказать потерпевшему первую медицинскую помощь;
- при любых случаях сбоя в работе технического оборудования или программного обеспечения немедленно вызвать представителя инженерно-технической службы эксплуатации вычислительной техники;
- в случае появления рези в глазах, резком ухудшении видимости-невозможности сфокусировать взгляд или навести его на резкость, появлении боли в пальцах и кистях рук, усилении сердцебиения следует немедленно покинуть рабочее место, сообщить о происшедшем руководителю работ и обратиться к врачу;
- при возгорании оборудования отключить питание и принять меры к тушению очага пожара при помощи углекислотного или порошкового огнетушителя, вызвать пожарную команду и сообщить о происшествии руководителю работ.

В случае отключения электропитания прекратите работу и доложите руководителю. Не пытайтесь самостоятельно выяснять и устранять причину. Помните, что напряжение может также неожиданно появиться.

При загорании или пожаре тушить электроустановки следует углекислотными или порошковыми огнетушителями, сухим песком во избежание поражения электрическим током.

5.5. Требования безопасности по окончании работы

окончании работы оператор обязан соблюдать следующую последовательность выключения вычислительной техники:

- произвести закрытие всех активных задач;
- выполнить парковку считывающей головки жесткого диска (если не предусмотрена автоматическая парковка головки);
- выключить питание системного блока (процессора);
- выключить питание всех периферийных устройств;
- отключить блок питания.

По окончании работы оператор обязан осмотреть и привести в порядок рабочее место.

5.6. Выводы

В данной главе была рассмотрена инструкция по обеспечению безопасности и охраны труда оператора ПК в КОГАУ «Центр развития туризма Кировской области».

Выявлены основные факторы, оказывающее вредное воздействие на сотрудников при работе с персональным компьютером и обозначены требования, выполнение которых способствует снижению воздействия вредных факторов. Данные требования разделены на общие, требования при аварийных ситуациях, перед началом работы, непосредственно при работе и по окончании работ.

Инструкция полностью соответствует стандартам, разработана на основе ТОИ Р-45-084-01 [46], готова к внедрению в организацию.

Заключение

В рамках выполнения выпускной квалификационной работы был проведён анализ деятельности Центра развития туризма Кировской области.

На основании проведённого анализа предоставляемых организацией услуг и существующих каналов коммуникаций с потребителями была выявлена проблема, актуальность которой подтверждает сама организация. Проблема заключается в невозможности предоставления потребителям развёрнутой информации о туристических маршрутах Кировской области, разрабатываемых организацией, и другой тематической информации.

Решение проблемы заключалось во внедрении сервиса, содержащего интерактивную карту, способную визуализировать разрабатываемые организацией маршруты и хранить их в едином каталоге. Кроме того, для организации было очень важным условием предоставить потребителям возможность самим строить персональные туристические маршруты.

С целью поиска готового решения, было проведено исследование рынка картографических сервисов, в котором сравнивались наиболее популярные и функционально наполненные сервисы. Однако на сегодняшний день на рынке не нашлось ни одного решения, полностью удовлетворяющего потребностям организации. Сделан вывод о необходимости и актуальности разработки собственного сервиса, в полной мере решающего проблемы Центра развития туризма.

Все требования к разработке сервиса, а также сроки и этапы работ были описаны в техническом задании. Требования были формализованы в виде UML диаграмм.

Часть сервиса, связанная с построением персональных туристических маршрутов, сводилась к частному случаю задачи комбинаторной оптимизации, известной как задача коммивояжёра. Классический и частный случаи задачи были описаны в виде математических моделей. Для нахождения способа решения задачи коммивояжёра были проанализированы существующие алгоритмы её решения, и выбран наиболее оптимальный из них. На его основе

был математически описан модифицированный алгоритм для решения частного случая задачи коммивояжёра.

Кроме того, математическую основу сервиса составлял рекомендательный алгоритм, удовлетворяющий потребностям организации в персонализации предоставляемых услуг. В качестве него был выбран и описан алгоритм коллаборативной фильтрации, способный порекомендовать пользователю потенциально интересный маршрут из каталога, на основе его истории путешествий, поставленных оценок и поиска подобных пользователей.

Продумана архитектура, логика и спроектированы основные интерфейсы приложения. Клиентская часть сервиса реализована в виде мобильного приложения с использованием Flutter SDK и языка программирования Dart. Серверную часть составили СУБД MS SQL Server и веб-приложение, написанное на Python с использованием фреймворка Django и реализующее алгоритмы и бизнес-логику сервиса.

Рассчитана стоимость программного продукта, включая внедрение и сопровождение. В ходе выполнения выпускной квалификационной работы в сроки был реализован объём работ, соответствующий ТЗ. Все работы были представлены организации и утверждены.

В перспективе дальнейшего сотрудничества с Центром развития туризма Кировской области стоит задача доработки сервиса, его внедрение и сопровождение.

Список использованных источников

1. Центр развития туризма Кировской области. [Электронный ресурс]. URL: https://www.kirovreg.ru/culture/objects/ckit.php (дата обращения 31.05.2020).

2. Туристско-информационный центр Кировской области. [Электронный ресурс]. URL: http://www.visitkirov.ru/tits/turistsko-informatsionnyj-tsentr-kirovskoj-oblasti/informatsiya-i-kontakty/ (дата обращения 31.05.2020).

3. Geospatial Analytics Market // MarketsandMarkets [Электронный ресурс]. URL: https://www.marketsandmarkets.com/Market-Reports/geospatial-analytics-market-198354497.html (дата обращения: 05.06.2020).

4. OAuth 2.0. [Электронный ресурс]. URL: https://oauth.net/2/ (дата обращения: 01.06.2020).

5. ГОСТ Р ИСО/ТС 10303-25-2012. Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Ч. 25. Методы реализации. Связь EXPRESS с XMI. – Введ. 2013-05-01. – М.: Стандартинформ, 2013. – 67 с.

6. Crow’s Foot Notation. [Электронный ресурс]. URL: http://www2.cs.uregina.ca/~bernatja/crowsfoot.html (дата обращения 01.06.2020).

7. Основы языка Transact-SQL : учеб. пособие / И. А. Казакова. – Пенза : Изд-во ПГУ, 2010. – 164 с.

8. ГОСТ 34.602-89. Техническое задание на создание автоматизированной системы (АС). – Введ. 1990-01-01. – М.: ИПК Издательство стандартов, 2002. – 18 с.

9. Menger K. On shortest polygonal approximations to a curve. Reports of a Mathematical Colloquium. 1940. p. 33-38.

10. Donald Davendra. Traveling Salesman Problem, Theory and Applications. Rijeka, Croatia: InTech, 2010.

11. Karp R.M. (1972) Reducibility among Combinatorial Problems. In: Miller R.E., Thatcher J.W., Bohlinger J.D. (eds) Complexity of Computer Computations. The IBM Research Symposia Series. Springer, Boston, MA

12. Левитин А. Алгоритмы. Введение в разработку и анализ. Вильямс, 2006. 35–36 с.

13. Land A. H., Doig A. G. An automatic method of solving discrete programming problems // Econometrica. 1960 Vol. 28. p. 497-520.

14. Little J. D. C., Murty K. G., Sweeney D. W., Karel C. An algorithm for the traveling salesman problem // Operations Research. 1963 Vol. 11, No 6. p. 972-989.

15. Johnson D. S., McGeoch L. A., Rothberg E. E. Asymptotic Experimental Analysis for the Held-Karp Traveling Salesman Bound. AT&T Bell Laboratories, 1999. p. 1–2.

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

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

18. Alsalibi B.A., Jelodar M.B., Venkat I. A Comparative Study between the Nearest Neighbor and Genetic Algorithms: A revisit to the Traveling Salesman Problem // International Journal of Computer Science and Electronics Engineering (IJCSEE), 2013. Vol. 1, Issue 1.

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

20. Копылова Е.С., Николаева Д.С., Бунтова Е.В. Решение задачи коммивояжёра с использованием метода ветвей и границ // Human progress. – 2018. – Том 4, № 4 [Электронный ресурс] URL: http://progress-human.com/images/2018/Tom4_4/Kopylova.pdf, свободный. (дата обращения: 02.04.2020).

21. Штовба, С.Д. Муравьиные алгоритмы / С.Д. Штовба // ExponentaPro. – 2003. – №4. – С. 70-75.

22. Беллман Р. Применение динамического программирования к задаче о коммивояжере // Кибернетический сборник. М.: Мир, 1964. Т. 9. С. 219 – 228.

23. Канцедал Сергей Андреевич, Костикова Марина Владимировна. Динамическое программирование для задачи коммивояжера // АСУ и приборы

автоматики. 2014. №166. [Электронный ресурс] URL: http://cyberleninka.ru/article/n/dinamicheskoe-programmirova. (дата обращения: 04.04.2020).

24. Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator. Hussain A, et al. Computational Intelligence and Neuroscience, 2017.

25. Abdulqader M. Mohsen. Annealing Ant Colony Optimization with Mutation Operator for Solving TSP. Computational Intelligence and Neuroscience, 2016.

26. Wei Gao. New Ant Colony Optimization Algorithm for the Traveling Salesman Problem. International Journal of Computational Intelligence Systems, Volume 13, Issue 1, 2020, p. 44 – 55.

27. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы / Под ред. В.М. Курейчика. – 2-е изд. испр. и доп. – М:ФИЗМАТЛИТ, 2017. – 368с.

28. Коллаборативная фильтрация // Википедия [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Коллаборативная_фильтрация (дата обращения: 05.06.2020).

29. Трёхуровневая архитектура // Википедия [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Трёхуровневая_архитектура (дата обращения: 01.06.2020).

30. Реляционные базы данных. Третья нормальная форма [Электронный ресурс]. URL: https://metanit.com/sql/tutorial/2.4.php (дата обращения 01.06.2020).

31. UI\UX Дизайн // Skillbox [Электронный ресурс]. URL: https://skillbox.ru/media/design/ux_ui_dizayn_chto_eto_takoe/ (дата обращения 01.06.2020).

32. Figma: the collaborative interface design tool. [Электронный ресурс]. URL: https://www.figma.com/ (дата обращения: 01.06.2020).

33. Система управления базами данных Microsoft Servers SQL // FB.ru [Электронный ресурс]. URL: http://fb.ru/article/145800/sistema-upravleniya-bazamidannyih-microsoft-servers-sql (дата обращения: 01.06.2020).

34. Django: The Web framework for perfectionists with deadlines [Электронный ресурс]. URL: https://www.djangoproject.com/ (дата обращения: 01.01.2020).

35. Model-View-Controller // Википедия [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Model-View-Controller (дата обращения 01.06.2020).

36. Аутентификация пользователей в Django [Электронный ресурс]. URL: https://djbook.ru/rel1.9/topics/auth/index.html (дата обращения 01.06.2020).

37. Flutter - Beautiful native apps in record time [Электронный ресурс]. URL: https://flutter.dev/ (дата обращения 01.06.2020).

38. Почему мобильное приложение на Flutter — хорошая идея для бизнеса в 2020 году // vc.ru [Электронный ресурс]. URL: https://vc.ru/dev/131105-pochemu-mobilnoe-prilozhenie-na-flutter-horoshaya-ideya-dlya-biznesa-v-2020-godu (дата обращения 03.06.2020).

39. PEP 8 – Style Guide for Python Code [Электронный ресурс]. URL: https://www.python.org/dev/peps/pep-0008/ (дата обращения 03.06.2020).

40. Django. Введение [Электронный ресурс]. URL: https://metanit.com/python/django/1.1.php (дата обращения 03.06.2020).

41. MapKit — SDK Яндекс.Карт — Технологии Яндекса [Электронный ресурс]. URL: https://tech.yandex.ru/maps/mapkit/ (дата обращения 01.06.2020).

42. Экономика организации (предприятий): учебник для вузов // под ред. проф. Горфинкеля, проф. Швандара. - М.: ЮНИТИ-Дана, 2006. – 608 с.

43. Скворцов, Ю. В. Организационно-экономические вопросы в дипломном проектировании: учеб. пособие. - М.: Высшая школа, 2006. – 399с.

44. Экономика фирмы: Учебник. // Под ред. проф. Н. П. Иващенко. М.: ИНФРА-М, 2007. – 528 с.

45. Amazon Web Services (AWS) - Cloud Computing Services [Электронный ресурс]. URL: https://aws.amazon.com/ru/pricing/ (дата обращения 03.06.2020).

46. Типовая инструкция по охране труда при работе на персональном компьютере // КонсультантПлюс [Электронный ресурс]. URL: http://www.consultant.ru/document/cons_doc_LAW_79762/ (дата обращения 04.06.2020).

47. Системы выработки рекомендаций // Business Data Analytics. Технологии добычи знаний и интеллектуального анализа данных [Электронный ресурс]. URL: http://www.businessdataanalytics.ru/RecommendationSystems.htm (дата обращения: 04.06.2020).

48. GitHub // Википедия [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/GitHub (дата обращения 04.06.2020).

Приложение А

Иерархическая структура экранных форм

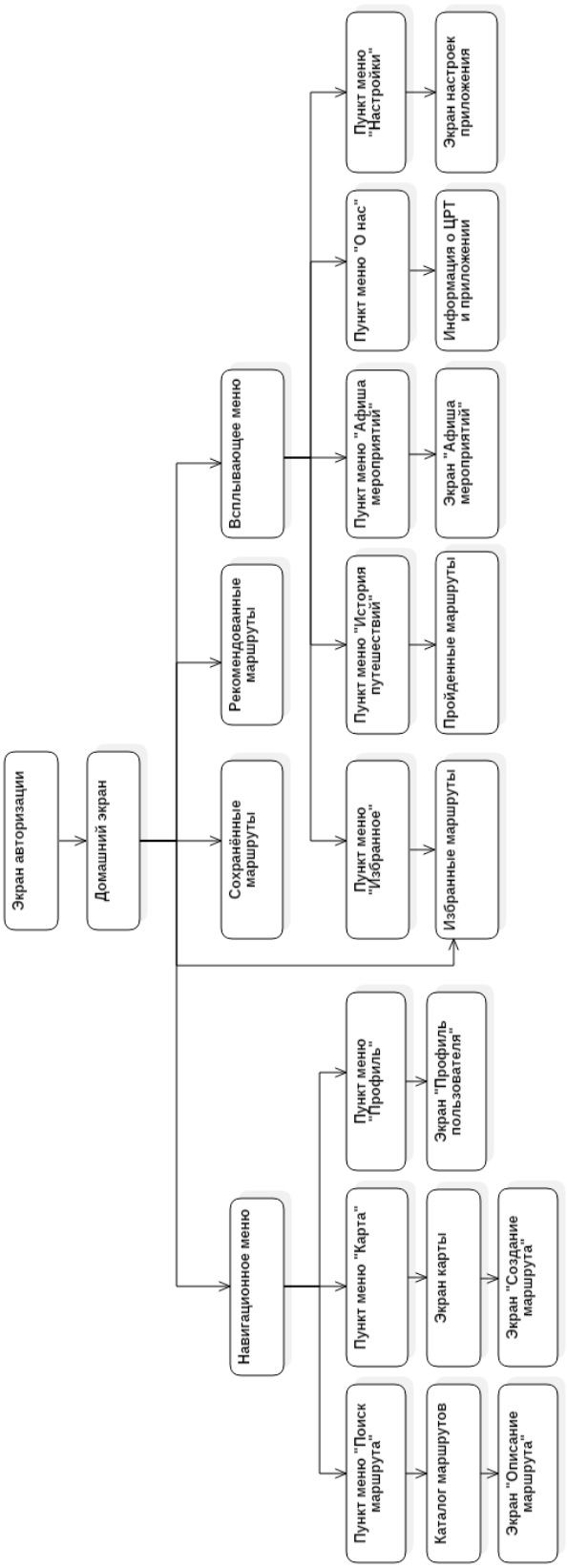


Рисунок 36 – Иерархическая структура экранных форм

Приложение Б

Программный код модели данных

Листинг 1 – Модель данных (Django)

```
from django.db import models
from django.conf import settings

class Category(models.Model):
    category_name = models.CharField('Наименование категории',
max_length=20)

    def __str__(self):
        return self.category_name

class Route(models.Model):
    route_name = models.CharField('Название маршрута', max_length=20)
    route_description = models.CharField('Описание', max_length=100)
    route_duration = models.DurationField()
    public = models.BooleanField(default=True)

    def __str__(self):
        return self.route_name

class Location(models.Model):
    loc_name = models.CharField('Название локации', max_length=30)
    loc_description = models.CharField('Описание', max_length=100)
    latitude = models.FloatField()
    longitude = models.FloatField()
    category_id = models.ForeignKey('Category', on_delete=models.CASCADE)

    def __str__(self):
        return self.loc_name

class RouteLocation(models.Model):
    route_id = models.ForeignKey('Route', on_delete=models.CASCADE)
    location_id = models.ForeignKey('Location', on_delete=models.CASCADE)
    location_num = models.IntegerField()
    start_point = models.NullBooleanField()
    end_point = models.NullBooleanField()

class Saved(models.Model):
    user_id = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE)
    route_id = models.ForeignKey('Route', on_delete=models.CASCADE)
    fav = models.BooleanField(default=False)
    saved = models.BooleanField(default=False)

class History(models.Model):
    user_id = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE)
    route_id = models.ForeignKey('Route', on_delete=models.CASCADE)
    datetime = models.DateTimeField()
    rate = models.IntegerField()
```

Приложение В

Программный код модифицированного алгоритма ACO

Листинг 2 – Модифицированный алгоритм ACO

```
import numpy as np


class Ant_Colony:
    def __init__(self, func, n_dim, s_max,
                 size_pop=10, max_iter=20,
                 distance_matrix=None, relevance=None,
                 alpha=1, beta=2, rho=0.9,
                 ):
        self.func = func
        self.n_dim = n_dim
        self.size_pop = size_pop
        self.max_iter = max_iter
        self.alpha = alpha
        self.beta = beta
        self.rho = rho
        self.relevance = relevance
        self.s_max = s_max
        self.s_max_start = s_max
        self.distance_matrix = distance_matrix

        self.Tau = np.ones((n_dim, n_dim))
        self.Table = np.zeros((size_pop, n_dim)).astype(np.int)
        self.prob_matrix_distance = 1 / (distance_matrix + 1e-
10 * np.eye(n_dim, n_dim))

        self.y = None
        self.x_best_history, self.y_best_history = [], []
        self.best_x, self.best_y = [], 0.

    def run(self, max_iter=None):

        self.max_iter = max_iter or self.max_iter

        # Нормирование релевантности
        relevance = np.copy(self.relevance)
        relevance_sum = relevance.sum()
        norm_relevance = []

        for item in range(len(self.relevance)):
            norm_relevance.append(relevance[item] / relevance_sum)
        norm_relevance = np.copy(norm_relevance)
```

```
        for i in range(self.max_iter):
            prob_matrix = (self.Tau ** self.alpha) * (self.prob_matrix_dista
nce) ** self.beta

            for j in range(self.size_pop):
                self.Table[j, 0] = 0
                self.Table[j, self.n_dim - 1] = self.n_dim - 1
                for k in range(self.n_dim - 1):
                    taboo_set = set(self.Table[j, :k + 1])
                    allow_list = list(set(range(self.n_dim)) - taboo_set)
                    prob = prob_matrix[self.Table[j, k], allow_list]
                    prob = prob / prob.sum()
                    next_point = np.random.choice(allow_list, size=1, p=prob
)[0]
                    self.Table[j, k+1] = next_point # k+1
            y = np.array([self.func(i) for i in self.Table])
            index_best = y.argmin()
            index_last = self.n_dim - 1
            x_best, y_best = self.Table[index_best, :].copy(), y[index_best]
.copy()
            self.x_best_history.append(x_best)
            self.y_best_history.append(y_best)

            delta_tau = np.zeros((self.n_dim, self.n_dim))
            for j in range(self.size_pop):
                funcm = 0
                for k in range(self.n_dim  - 1):
                    funcm += norm_relevance[k]
                    n1 = self.Table[j, k]
                    n2 = self.Table[j, k + 1]
                    delta_tau[n1, n2] += (self.s_max * pow(funcm, 1./self.rh
o)) / (y[j] * max(norm_relevance[0:k+1]))
            self.Tau = (1 - self.rho) * self.Tau + delta_tau
            best_generation = np.array(self.y_best_history).argmin()
            self.best_x = self.x_best_history[best_generation]
            self.best_y = self.y_best_history[best_generation]

            if self.best_x is None or self.best_y is None:
                self.best_x, self.best_y = 0, 0

            if (i == self.max_iter - 1) and (y[j] > self.s_max):
                print('Не найден оптимальный маршрут, повторяю поиск...')
                relevance = list(relevance[:])
                new_relevance = relevance[1:-1]
                if not new_relevance or self.n_dim == 2:
                    best_generation = np.array(self.y_best_history).argmin()

                    self.best_x = self.x_best_history[best_generation]
                    self.best_y = self.y_best_history[best_generation]
```

```
                    if self.best_x is None or self.best_y is None:
                        self.best_x, self.best_y = [], 0

                    return self.best_x, self.best_y
                else:
                    min_relevance = min(new_relevance)
                    min_index = new_relevance.index(min_relevance)
                    del relevance[min_index+1]
                    print('Оставшиеся значения релевантностей:', relevance)
                    distance_matrix = list(self.distance_matrix[:])
                    del distance_matrix[min_index+1]

                    new_distance_matrix = []

                    for el in distance_matrix:
                        el = list(el)
                        del el[min_index + 1]
                        new_distance_matrix.append(list(el))

                    new_n_dim = self.n_dim - 1
                    aca = Ant_Colony(func=self.func, n_dim=new_n_dim,
                            size_pop=self.size_pop, max_iter=self.max_iter,
                                     distance_matrix=new_distance_matrix,
                                     s_max=self.s_max_start,
                                     relevance=relevance)
                    best_x, best_y = aca.run()

                    if best_x != [] and best_y != 0:
                        # %% Plot
                        fig, ax = plt.subplots(1, 2)
                        best_points_ = best_x
                        best_points_coordinate = points_coordinate[best_poin
ts_, :]
                        ax[0].plot(best_points_coordinate[:, 0], best_points
_coordinate[:, 1], 'o-r')
                    pd.DataFrame(aca.y_best_history).cummin().plot(ax=ax[1])
                        plt.show()
                    return [], 0

            elif y[j] <= self.s_max:
                return self.best_x, self.best_y
```

Приложение Г

Программный код алгоритма коллаборативной фильтрации

Листинг 3 – Алгоритм коллаборативной фильтрации

```
import numpy as np
from math import sqrt

def sim_pearson(prefs, person1, person2):
    si = {}
    for item in prefs[person1]:
        if item in prefs[person2]:
            si[item] = 1

    if len(si) == 0:
        return 0
    sum1 = sum([prefs[person1][item] for item in si])
    sum2 = sum([prefs[person2][item] for item in si])

    sum1_sq = sum([pow(prefs[person1][item], 2) for item in si])
    sum2_sq = sum([pow(prefs[person2][item], 2) for item in si])

    p_sum = sum([prefs[person1][item] * prefs[person2][item] for item in
si])
    num = p_sum - (sum1 * sum2 / len(si))
    den = sqrt((sum1_sq - pow(sum1, 2) / len(si)) * (sum2_sq - pow(sum2,
2) / len(si)))

    if den == 0: return 0
    return num/den

def top_matches(prefs, person, n=3, similarity=sim_pearson):
    scores = [(similarity(prefs, person, other), other) for other in prefs
if other != person]

    scores.sort()
    scores.reverse()
    return scores[0:n]

def get_recommendations(prefs, person, similarity=sim_pearson):
    totals = {}
    sim_sums = {}
    for other in prefs:
        if other == person: continue
        sim = similarity(prefs, person, other)
        if sim <= 0: continue
        for item in prefs[other]:
            if item not in prefs[person] or prefs[person][item] == 0:
                totals.setdefault(item, 0)
                totals[item] += prefs[other][item]*sim
                sim_sums.setdefault(item, 0)
                sim_sums[item] += sim
    rank = [(total / sim_sums[item], item) for item, total in
totals.items()]
    rank.sort()
    rank.reverse()
    return rank
```

АВТОРСКАЯ СПРАВКА

Я, _Бокарева Дарья Геннадьевна_____
автор выпускной квалификационной работы _Разработка рекомендательного сервиса построения персональных туристических маршрутов_____
сообщаю, что мне известно о персональной ответственности автора за разглашение сведений, подлежащих защите законами РФ о защите объектов интеллектуальной собственности.

Одновременно сообщаю, что:

1. При подготовке к защите выпускной квалификационной работы не использованы источники (документы, отчеты, диссертации, литература и т.п.), имеющие гриф секретности или "Для служебного пользования" ВятГУ или другой организации.

2. Данная работа не связана с незавершенными исследованиями или уже с завершенными, но еще официально не разрешенными к опубликованию ВятГУ или другими организациями.

3. Данная работа не содержит коммерческую информацию, способную нанести ущерб интеллектуальной собственности ВятГУ или другой организации.

4. Данная работа не является результатом НИР или ОКР, выполняемой по договору с организацией (указать согласие заказчика) _____
_____
_____

5. В предлагаемом к опубликованию тексте нет данных по незащищенным объектам интеллектуальной собственности других авторов.

6. Согласен на использование результатов своей работы безвозмездно в ВятГУ для учебного процесса, а также на размещение своей работы в электронной информационно-образовательной среде ВятГУ.

7. Использование моей выпускной квалификационной работы в научных исследованиях оформляется в соответствии с законодательством РФ о защите интеллектуальной собственности.

Автор _____ ____Д. Г. Бокарева____
/личная подпись/ /И. О. Фамилия/

"19" июня 2020 г.

Сведения по авторской справке подтверждаю:

И.о. зав. кафедрой _____ _____Братухина Е.А.
/личная подпись/ /И. О. Фамилия/

"23" июня 2020 г.