

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«__» _____ 2021 г.

Приложение-навигатор для построения адаптивного маршрута
по достопримечательностям «Личный гид»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____ В.А. Парасич
«__» _____ 2021 г.

Автор работы,
студент группы КЭ-405
_____ А.Д. Борзых
«__» _____ 2021 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«__» _____ 2021 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ

Заведующий кафедрой ЭВМ

_____ Г.И. Радченко

«___» _____ 2021 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра
студенту группы КЭ-405
Борzych Андрею Дмитриевичу
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы:** «Приложение-навигатор для построения адаптивного маршрута по достопримечательностям «Личный гид»» утверждена приказом по университету от 26 апреля 2021 г. №714-13/12
2. **Срок сдачи студентом законченной работы:** 1 июня 2021 г.
3. **Исходные данные к работе:**

Обеспечить основной функционал приложения:

1. Отображение имеющихся достопримечательностей в выбранном городе на карте.
2. Размещение информации о достопримечательностях.
3. Возможность указания индивидуальных предпочтений.
4. Организация сохранения маршрутов.
5. Организация построения адаптивного маршрута с учетом наличия индивидуальных предпочтений пользователя.
6. Возможность составления заявки на добавление достопримечательности на карту.

7. Возможность составления отзыва, предложения по улучшению или обращения в техподдержку.

Предусмотреть наличие ролей гостя, активного пользователя, модератора, администратора.

Обеспечить регистрацию и смену пароля с отправкой письма на почту пользователя.

Среда и средства реализации – по выбору автора.

4. Перечень подлежащих разработке вопросов:

1. Анализ аналогов разрабатываемого приложения.
2. Детализация требований к приложению.
3. Выбор среды и средств реализации.
4. Проектирование архитектуры приложения.
5. Организация базы данных.
6. Создание серверной и клиентской частей приложения.
7. Тестирование разработанного программного обеспечения.

5. Дата выдачи задания: 1 февраля 2021 г.

Руководитель работы _____ /В.А. Парасич/

Студент _____ /А.Д. Борзых/

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение	05.02.2021	
Обзор литературы	12.02.2021	
Разработка модели, проектирование	19.03.2021	
Реализация системы	26.03.2021	
Тестирование системы	05.04.2021	
Оформление	20.04.2021	
Подготовка презентации	25.04.2021	

Руководитель работы _____ /В.А. Парасич/

Студент _____ /А.Д. Борзых/

АННОТАЦИЯ

А.Д. Борзых. Приложение-навигатор для построения адаптивного маршрута по достопримечательностям «Личный гид». – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭЖН; 2021, 151- с., 56 ил., библиогр. список – 26 наим.

В рамках выпускной квалификационной работы разрабатывается веб-приложение для построения адаптивного маршрута по достопримечательностям. Система учитывает недостатки аналогичных приложений и является их улучшенной версией с более обширным функционалом построения маршрута.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.....	7
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ	9
1.1. ОБЗОР АНАЛОГОВ.....	9
ВЫВОД.....	12
1.2. АНАЛИЗ ОСНОВНЫХ ТЕХНОЛОГИЧЕСКИХ РЕШЕНИЙ	13
ВЫБОР ЯЗЫКА ПРОГРАММИРОВАНИЯ ДЛЯ BACKEND	13
ВЫБОР ФРЕЙМВОРКА.....	16
ВЫБОР СИСТЕМЫ УПРАВЛЕНИЯ БАЗОЙ ДАННЫХ.....	18
1.3. ВЫВОД.....	26
2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ	27
2.1. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ	27
2.2. НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ.....	37
3. ПРОЕКТИРОВАНИЕ	40
3.1. АРХИТЕКТУРА ПРЕДЛАГАЕМОГО РЕШЕНИЯ	40
3.2. АЛГОРИТМЫ РЕШЕНИЯ ЗАДАЧИ	61
3.3. ОПИСАНИЕ ДАННЫХ	74
4. РЕАЛИЗАЦИЯ	86
5. ТЕСТИРОВАНИЕ	103
ЗАКЛЮЧЕНИЕ	119
БИБЛИОГРАФИЧЕСКИЙ СПИСОК	120
ПРИЛОЖЕНИЕ А	123

ВВЕДЕНИЕ

В России культурный туризм уже давно стал самым популярным и массовым видом туризма. С каждым годом растет количество людей, желающих познакомиться с историческими и культурными достопримечательностями страны. Многие хотят своими глазами увидеть известные исторические и природные памятники, которыми так богата российская земля. Особенно привлекает любителей культурного туризма центральный и северо-западный регионы, где сосредоточены основные достопримечательности России [1].

В современном мире культурный туризм занимает очень важное место, так как во время путешествий человек познает себя и окружающую природу, изучает исторические, культурные факты.

Кроме того, путешествия подразумевают взаимодействие между культурами. Ни один человек не может жить без взаимоотношения с другими людьми. Для развития необходим контакт с другими народами, и культурный туризм этому способствует. Именно культурный туризм позволяет не только накопить знания, но и получить бесценный опыт в ходе встречи с людьми, которые принадлежат к разным культурам. В дальнейшем подобный подход позволяет выработать свою систему ценностей и приоритетов. Люди, таким образом, познают мир и делают для себя выводы, которые в дальнейшем оказывают определяющее влияние на них в жизни, их поведение и действия в конкретных ситуациях [2].

Однако процесс развития этого движения тормозит отсутствие информационной инфраструктуры, позволяющей интегрировать систему ознакомления с различными историческими, архитектурными или культурными эпохами путём вывода справочной информации по достопримечательностям,

построения маршрута посещения архитектурных памятников, музеев. Ликвидации этой проблемы посвящена данная работа.

В одном приложении сосредоточена вся необходимая информация (о достопримечательностях, индивидуальных предпочтениях пользователя, сохраненных маршрутах). Налажена система подачи заявок на добавление новых достопримечательностей. На карте можно посмотреть расположение достопримечательностей города, построить маршрут с учетом наличия индивидуальных предпочтений пользователя и внешних факторов. Поддерживается система обратной связи с пользователями.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. ОБЗОР АНАЛОГОВ

1. Waytips.com

«Waytips создан для любителей самостоятельных путешествий. Сервис содержит базу фотографий и советов туристов о различных местах нашей планеты на русском, английском, реже - других языках.

Сервис умеет быстро и точно строить туристические маршруты по городам, основанные на ваших интересах. Любите музеи и галереи - маршрут будет один, любите прогулки и парки - другой. Каждый сможет оптимально спланировать свои перемещения в незнакомом городе.

С помощью Waytips также можно спланировать длинный автомобильный маршрут с множеством путевых точек или одно большое путешествие.

Наши пользователи могут вести свои тревел-блоги в простом и удобном виде, и что самое главное - привязывать свои записи к путешествиям или путевым точкам путешествий. В итоге каждый может наглядно показать где он был, что он видел и как перемещался по миру» [3].

Преимущества веб-приложения:

1. Возможность выбрать достопримечательности из списка или на карте.
2. Наличие краткого описания достопримечательностей.
3. Автоматическое распределение путешествия на несколько дней.
4. Возможность сохранить маршрут.

Недостатки веб-приложения:

1. Нельзя задать способ передвижения по достопримечательностям.
2. Построение маршрута сводится к соединению достопримечательностей прямыми на карте, не учитываются личные

предпочтения пользователя, его местоположение.

2. Youroute.ru

«Путеводитель уже содержит карты маршрутов, описания и фотографии достопримечательностей, информацию об отелях и ресторанах, общие данные о стране/странах, а также полное расписание поездки в удобном формате календаря.

Вы также сможете отредактировать любой объект, добавить или изменить фотографии, полностью переписать его описание или внести небольшие коррекционные правки, которые после перепроверки нашими экспертами, будут размещены на сайте.

Сохраняйте свои будущие маршруты в своем личном кабинете, рекомендуем проверенные маршруты, делитесь впечатлениями о поездках в блоге, создайте тематический маршрут, добавляйте фотографии в свои рассказы и на главную страницу сайта, отправляйте ваши альбомы с фотографиями о совершенных поездках вашим друзьям, задавайте и отвечайте на вопросы» [4].

Преимущества веб-приложения:

1. Возможность выбрать достопримечательности из списка или на карте.
2. Отобразить места выбранной категории.
3. Возможность сохранить маршрут.

Недостатки веб-приложения:

1. Нельзя задать способ передвижения по достопримечательностям.
2. Построение маршрута сводится к соединению достопримечательностей прямыми на карте, не учитываются личные предпочтения пользователя, его местоположение.
3. Отсутствует описание достопримечательностей.

3. Maps.sygic.com

«Sygic Travel Maps-это первые в мире карты, отображающие лучшие достопримечательности, отели, рестораны или магазины непосредственно на карте» [5].

Преимущества веб-приложения:

1. Возможность выбрать достопримечательности на карте.
2. Распределить путешествие на несколько дней.
3. Возможность сохранить маршрут.
4. Выбрать способ передвижения между каждой парой достопримечательностей.
5. Присутствует описание достопримечательностей.

Недостатки веб-приложения:

1. Можно указать только передвижение пешком или на автомобиле, при выборе общественного транспорта для передвижения между достопримечательностями проводится прямая линия.
2. Не учитываются личные предпочтения пользователя.

4. Evertravel.me

На сайте и в приложении EverTravel имеется несколько десятков городов. В свою очередь, в каждом городе вы обнаружите до 150 интересных мест, музеев, кафе, торговых центров, парков отдыха и развлечений. Каждый объект сопровождается подробным описанием, фото, меткой на карте и сведениями о часах работы и стоимости билетов [6].

Преимущества веб-приложения:

1. Возможность выбрать достопримечательности на карте.
2. Распределить путешествие на несколько дней.
3. Возможность сохранить путешествие.
4. Присутствует описание достопримечательностей.

Недостатки веб-приложения:

1. Отсутствует возможность построения маршрута путешествия.
2. Не учитываются личные предпочтения пользователя.

ВЫВОД

Таким образом, имеется несколько приложений, которые можно использовать для планирования путешествий, но все они имеют недостатки в функционале построителя маршрутов: построение маршрута сводится к соединению достопримечательностей прямыми на карте, либо построитель учитывает не все средства передвижения, также не учитываются личные предпочтения пользователя. Следовательно, создание приложения, свободного от выявленных недостатков, является актуальной задачей.

1.2. АНАЛИЗ ОСНОВНЫХ ТЕХНОЛОГИЧЕСКИХ РЕШЕНИЙ

Выбор языка программирования для backend

Для разработки веб-приложения можно использовать следующие языки: Python, Java, Ruby, PHP, JavaScript(Node.js). Далее рассмотрим особенности каждого языка, которые повлияли на наш выбор.

PHP – язык программирования, исполняемый на стороне веб-сервера.

Достоинства PHP [7]:

1. Является свободным программным обеспечением, распространяемым под особой лицензией.
2. Легок в освоении на всех этапах.
3. Поддерживается большим сообществом пользователей и разработчиков.
4. Имеет развитую поддержку баз данных.
5. Имеется огромное количество библиотек и расширений языка.
6. Может быть развёрнут почти на любом сервере.
7. Портирован под большое количество аппаратных платформ и операционных систем.

Ruby – динамический императивный объектно-ориентированный язык программирования.

Недостатки Ruby [7]:

1. Обучение языку выше начального уровня может оказаться непростым.
2. Информационных ресурсов, посвящённых Ruby, недостаточно.
3. Ruby менее производителен по сравнению со многими другими языками, применяемыми в веб-разработке.
4. Ruby относительно медленно разрабатывается и развивается.

Python широко применяется как интерпретируемый язык для скриптов различного назначения.

Недостатки Python [7]:

1. Не слишком удачная поддержка многопоточности.
2. Отсутствие коммерческой поддержки средств разработки.
3. Изначальная ограниченность средств для работы с базами данных.
4. Бенчмарки показывают меньшую производительность Python по сравнению с остальными языками программирования, что создаёт этому языку репутацию медленного.

Java — язык программирования общего назначения, который следует парадигме объектно-ориентированного программирования и подходу «Написать один раз и использовать везде».

Недостатки Java [8]:

1. Платное коммерческое использование.
2. Низкая производительность.
3. Отсутствие нативного дизайна.
4. Многословный и сложный код.

Node.js — это среда, которая позволяет использовать JavaScript как для backend, так и для frontend разработки, а также для решения проблем совместимости.

Недостатки Node.js [9]:

1. Требуется чистой архитектуры (Это событийно-ориентированная среда, поэтому она может запускать несколько событий одновременно, но только если отношения между ними хорошо прописаны.).
2. Не поддерживает задачи с высокой загрузкой процессора.
3. Недостаточно развитая документация.

Для создания веб-приложения выбран язык программирования PHP, так как данный интерпретируемый язык программирования высокого уровня ориентирован именно для решения задач веб-разработки. Главными факторами выбора языка PHP являются:

1. Простота.
2. Эффективность.
3. Огромное количество библиотек и расширений языка.
4. Обширная документация
5. Поддержка различных баз данных.
6. Возможность работы почти на любом сервере.

Для упрощения разработки можно использовать либо CMS, либо фреймворк.

CMS – это система управления контентом. С помощью CMS вы полностью можете управлять сайтом, его наполнением, страницами, изображениями, видео, оформлением.

Рассмотрим особенности фреймворка и CMS, которые повлияли на наш выбор [10-11].

Недостатки использования CMS:

1. Ограниченная функциональность. Большинство CMS удовлетворительно решают не более двух задач, на которые они рассчитаны, что ограничивает круг их использования.
2. Избыточность. В погоне за универсальностью, разработчики CMS закладывают много функций, скорее всего для одного проекта всё это не нужно. В итоге получаем много неиспользуемого кода.
3. Сложное ядро. Порой, чтобы чуть поправить или видоизменить какую-то часть сайта, приходится разбираться с устройством системы.
4. Сайт на CMS всегда уступает в производительности хорошо

написанному сайту на фреймворке.

Фреймворк - это надстройка над языком, набор библиотек.

Достоинства использования фреймворка:

1. Гибкость.
2. Производительность. По скорости работы могут уступать только веб-приложениям, полностью написанным на PHP.

Таким образом, используя CMS, мы привязываемся к структуре, созданной разработчиками, но в шаблонных решениях экономим время. Фреймворк даёт полную свободу действий. За нас пишут только основу, фундамент. Но для качественной разработки на фреймворке необходимо обладать достаточным уровнем знаний. В данной работе будет использоваться фреймворк.

Выбор фреймворка

Для сравнения инструментов разработки были выбраны следующие PHP фреймворки:

1. Laravel.
2. Yii.
3. Symfony.
4. Zend Framework.
5. Codeigniter.

Далее рассмотрим особенности каждого фреймворка, которые повлияли на наш выбор [12].

Laravel — это бесплатный PHP-фреймворк с открытым исходным кодом, созданный Тейлором Отвеллом для разработки веб-приложений по архитектурному шаблону MVC.

Недостатки Laravel:

1. Синтаксический сахар в Laravel как плюс, так может быть и минусом. Очень легко привыкнуть к нему и позабыть, как пишутся чистые

запросы и функции.

2. Нарушение обратная совместимости между версиями фреймворка.

Yii - это бесплатный объектно-ориентированный компонентный full-stack PHP фреймворк.

Недостатки Yii:

1. Хотя фреймворк и позволяет делать код простым, но далеко не элегантным. Если его синтаксис сравнивать с фреймворком Laravel, то он уступает.
2. Yii отстает от языка, стандартов и других фреймворков. Новые обновления с действительно полезными функциями выходят не так часто.
3. Слишком большая связанность backend и frontend частей Yii2. Фреймворк предлагает использовать библиотеку jQuery и Bootstrap, которые встроены по умолчанию в ядро фреймворка.

Symfony - свободный PHP фреймворк для быстрой разработки веб-приложений и решения рутинных задач веб-программистов.

Недостатки Symfony:

1. Небольшое сообщество разработчиков.
2. Перенасыщенность разного рода сущностей.

Zend Framework - это свободный объектно-ориентированный PHP фреймворк для разработки веб-приложений, разработанный и поддерживаемый компанией Zend.

Недостатки Zend Framework:

1. Не подходит для быстрой разработки проектов.
2. Для русскоязычного сегмента разработчиков мало полезных материалов по разработке.

CodeIgniter - это популярный PHP микро-фреймворк с открытым исходным кодом, для разработки веб-систем и приложений.

Достоинства CodeIgniter:

1. Отличная документация и англоязычное сообщество.
2. Высокая производительность фреймворка.
3. Небольшой размер фреймворка.
4. Предоставляет легкие и простые решения для разработки.
5. Подходит для быстрой разработки небольших сайтов и веб-приложений.
6. Структура фреймворка не требует строгих правил кодирования.
7. Не требует сложной настройки, почти нулевая конфигурация.
8. MVC-архитектура веб-приложения.
9. Слабая связанность компонентов.
10. Множество подключаемых библиотек и помощников.

Таким образом, для разработки проекта выбран PHP-фреймворк CodeIgniter. Главными достоинствами для выбора данного фреймворка послужили следующие преимущества:

1. Высокая производительность.
2. Небольшой размер.
3. Легкие и простые решения для разработки.

Выбор системы управления базой данных

Базы данных — это специально разработанное хранилище для различных типов данных. Каждая база данных, имеет определённую модель (реляционная, документно-ориентированная), которая обеспечивает удобный доступ к данным. Системы управления базами данных (СУБД) — специальные приложения (или библиотеки) для управления базами данных различных размеров и форм.

Существует множество систем управления базами данных, которые можно использовать для веб-приложений: MySQL, Microsoft SQL Server, PostgreSQL, MongoDB, MariaDB, SQLite.

SQLite.

Недостатки SQLite [13]:

1. Отсутствие системы пользователей - более крупные СУБД включают в свой состав системы управления правами доступа пользователей.
2. Отсутствие возможности увеличения производительности.

PostgreSQL.

Недостатки PostgreSQL [13-14]:

1. Производительность - при простых операциях чтения PostgreSQL может значительно замедлить сервер.
2. Хостинг. Иногда довольно сложно найти хостинг с поддержкой этой СУБД.

PostgreSQL идеально подходит для проектов с ограниченным бюджетом, но квалифицированными специалистами.

MongoDB.

Недостатки MongoDB [14]:

1. SQL не используется в качестве языка запросов.
2. Инструменты для перевода SQL-запросов в MongoDB доступны, но их следует рассматривать именно как дополнение.

Подходит для организаций, работающих с разнородными данными, которые тяжело поддаются классификации.

MariaDB.

Недостатки MariaDB [14]:

1. На данный момент стабильность ниже, чем у MySQL.
2. Движок довольно новый, поэтому пока нет никаких гарантий

дальнейших обновлений.

3. Как и во многих других бесплатных базах данных, придется платить за поддержку.

Microsoft SQL Server.

Недостатки Microsoft SQL Server [14]:

1. Цена для юридических лиц оказывается неприемлемой для большей части организаций.
2. Даже при тщательной настройке производительности SQL Server способен занять все доступные ресурсы.

Microsoft SQL Server идеально подходит для крупных организаций, которые уже используют ряд продуктов Microsoft.

MySQL.

Достоинства MySQL [14]:

1. Простота в работе - установить MySQL довольно просто.
2. Богатый функционал - MySQL поддерживает большинство функционала SQL.
3. Безопасность - большое количество функций, обеспечивающих безопасность, которые поддерживаются по умолчанию
4. Масштабируемость - MySQL легко работает с большими объемами данных и легко масштабируется
5. Скорость - упрощение некоторых стандартов позволяет MySQL значительно увеличить производительность.
6. Распространяется бесплатно.

MySQL идеально подходит для: организаций, которым требуется надежный инструмент управления базами данных, но бесплатный [13-14].

Таким образом, при разработке веб-приложения будет использоваться MySQL, так как эта СУБД сочетает в себе продвинутый функционал и

свободный доступ к исходному коду, высокий уровень безопасности (система безопасности включает в себе простые и в то же время достойные способы защиты доступа к данным), скорость (упрощение некоторых стандартов позволяет значительно увеличить производительность) и масштабируемость.

Выбор средств разработки для frontend

Для разработки клиентской части будут использоваться следующие средства:

1. Язык разметки HTML.
2. Таблицы каскадных стилей CSS.
3. Фреймворк Bootstrap 4.
4. Язык программирования JavaScript.

Bootstrap 4.

Разработка адаптивных шаблонов в разы сложнее разработки фиксированных макетов. Вам нужно сделать так, чтобы на любых разрешениях экранов ваше приложение отображалось хорошо. Для этого придется использовать медиа-запросы, которых для крупных шаблонов может понадобиться очень много.

Главным элементом Bootstrap является адаптивная сетка. Именно благодаря сетке можно быстро создавать адаптивные шаблоны. При использовании Bootstrap можно вообще не быть знакомым с медиа-запросами, они не нужны, потому что фреймворк берет на себя реализацию адаптивности, нужно лишь задать блокам правильные классы [15].

Карты.

На рынке картографических и справочных сервисов можно выделить три основных конкурента:

1. Яндекс.Карты.
2. 2ГИС.

3. Google Maps.

Сравнение карт по конкурирующим критериям приведено в таблице 1.2.1 [16]:

Таблица 1.2.1 – Сравнение карт по конкурирующим критериям

Критерий	Яндекс.Карты	2ГИС	Google Maps
Покрытие	Лучшее покрытие России, уступает Google в покрытии мира	Уступает конкурентам в покрытии как в России, так и в других странах	Лучшее покрытие всего мира
Детализация	Хорошая детализация России, достаточная в мире	Одна из лучших детализаций в городах присутствия	Хорошая детализация по всему миру. На карте России могут отсутствовать крупные города. В плане отображения невнятная детализация. Объекты хорошо видны только при достаточно сильном приближении.

Продолжение таблицы 1.2.1

Критерий	Яндекс.Карты	2ГИС	Google Maps
Детализация на уровне здания	Нет	Небольшие склады, кафедры университетов	Крупные торговые центры
Возможность загрузки и использования офлайн	Да. Большой размер данных	Да	Да. Большой размер данных
Редактирование карт	Сервис «Народная карта» (web); Сообщение об ошибках	Сообщение об ошибках	Сообщение об ошибках
Вариант выбора отображения ландшафта	Карта, спутник, народная карта	Карта	Карта, спутник, Велокарта, общественный транспорт
Отображение пробок в крупных городах	Да. Отображение доп. Информации о дорожной обстановке	Не все города	Не все города. Интеграция с сервисом Waze
Возможность общения между пользователями	«Разговорчики»	Нет	Нет

Продолжение таблицы 1.2.1

Критерий	Яндекс.Карты	2ГИС	Google Maps
Обзорные фотографии улиц (Streetview)	Яндекс Панорамы	Нет	Google Streetview
Поиск универсальный	Да. Интеллектуальный поиск	Да	Да. Интеллектуальный поиск
Голосовой ввод (на русском)	Да	Нет	Да
Режим 3D	Одинаковая высота зданий	Да	Да
Ночной режим	Да	Нет	Да
Построение маршрута	Автомобиль, общественный транспорт. Строит с учетом пробок. Требуется интернет для построения	Автомобиль, общественный транспорт. Возможность отдельно выбрать вариант «Метро». Не требует интернета для построения маршрута	Автомобиль, общественный транспорт, пешеходный маршрут. Возможность выбрать только один из видов транспорта или вариант пешком. Строит с учетом пробок

Продолжение таблицы 1.2.1

Критерий	Яндекс.Карты	2ГИС	Google Maps
Справочная информация	Подробная информация об организациях	Подробная информация об организациях. Ежемесячные обновления	Хуже других знает российские организации
Актуализация справочной информации	Нет информации	Обновления каждый месяц	Нет информации
Возможность оставить отзывы и оценить организацию	Оценка. Развивается сервис Яндекс.Город	Интеграция с сервисом Фламп	Отзыв и оценка
Интерфейс и удобство	Современный интерфейс. Осуществление большинства функций возможно в два шага	Интерфейс iPhone версии не адаптирован для iOS 7	Современный интерфейс. Некоторые функции не до конца понятны на интуитивном уровне

Продолжение таблицы 1.2.1

Критерий	Яндекс.Карты	2ГИС	Google Maps
Стоимость	Бесплатная, если количество запросов в сутки не превышает 25 тысяч.	Бесплатная при Использовании не в коммерческих целях	Бесплатная, если количество запросов в месяце превышает 28 тысяч

Таким образом, будут использоваться Яндекс.Карты. Потому что Яндекс.Карты обладают наилучшим покрытием России среди конкурентов, хотя и уступают в покрытии всего мира Google Maps. Также в данной работе не требуется такая обширная детализация зданий, которую предоставляет 2ГИС.

1.3. ВЫВОД

Исследование рынка приложений для планирования маршрутов путешествий показало, что ни одно из них не является полнофункциональным. Следовательно, создание приложения, свободного от выявленных недостатков, является актуальной задачей. Для ее решения выбраны следующие технологические решения: для backend — язык программирования PHP, фреймворк — CodeIgniter, СУБД — MySQL. Для frontend — язык разметки HTML, таблицы каскадных стилей CSS, язык программирования — JavaScript, фреймворк — Bootstrap 4.

2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

2.1. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

1. Обеспечить отображение всех имеющихся достопримечательностей в выбранном городе на карте с учетом индивидуальных предпочтений пользователя, так и без них.
2. Наличие возможности просмотра информации о достопримечательностях (описание, местоположение).
3. Возможность задания индивидуальных предпочтений в личном кабинете пользователем для дальнейшего использования при построении маршрутов.
4. Система сохранения составленного маршрута посещения достопримечательностей. Возможность просмотра и редактирования сохраненных составленных маршрутов.
5. Система построения адаптивного маршрута с учетом наличия индивидуальных предпочтений пользователя и внешних факторов (доступность достопримечательностей, местоположение пользователя).
6. Возможность написания и отправки заявки на добавление достопримечательности на карту.
7. Возможность написания и отправки отзыва о приложении, предложения по улучшению приложения, обращения в техподдержку.
8. Система модерирования приложения.
9. Система администрирования приложения.

Требования к функционалу подсистемы регистрации и авторизации

Подсистема имеет следующий функционал:

1. Зарегистрировать пользователя в системе.
2. Авторизовать пользователя в системе.

Подсистема имеет связь с:

1. Хранилищем информации о пользователях.
2. Всеми другими подсистемами, поскольку авторизует пользователя и информация о нем используется в других подсистемах.

Входные данные при регистрации:

1. Имя пользователя (необязательно настоящее).
2. E-mail пользователя.
3. Пароль для входа в систему.
4. Повторный пароль для входа в систему.

Входные данные при авторизации:

1. Имя пользователя.
2. Пароль для входа в систему.

Выходные данные: пользователь регистрируется или авторизуется в приложении.

Требования к функционалу подсистемы взаимодействия с личным кабинетом

Подсистема имеет функционал редактирования информации о пользователе.

Подсистема имеет связь с:

1. Хранилищем информации о пользователях.

2. Хранилищем категорий достопримечательностей.
3. Хранилищем информации о всех отзывах, предложениях по улучшению, обращениях в техподдержку.
4. Хранилищем информации о всех заявках на добавление достопримечательностей.
5. Хранилищем сохраненных маршрутов посещения достопримечательностей.
6. Подсистемой регистрации и авторизации, поскольку пользователю сначала нужно авторизоваться, прежде чем перейти в личный кабинет.
7. Подсистемой взаимодействия с сохраненными маршрутами посещения достопримечательностей.
8. Подсистемой создания отзывов, предложений по улучшению и обращений в техподдержку.
9. Подсистемой создания заявок на добавление достопримечательностей.

Входные данные:

1. Категории достопримечательностей, которые интересуют пользователя и имеются в приложении (категории достопримечательностей задаются администратором).
2. Имя пользователя.

Выходные данные: добавленная/измененная в хранилище данных информация о пользователе, которая будет использоваться при построении маршрута.

Требования к функционалу подсистемы создания заявок на добавление достопримечательностей

Подсистема имеет следующий функционал:

1. Просмотр списка созданных заявок пользователем.
2. Предоставить пользователю форму для создания заявки.
3. Отправить заявку.

Подсистема имеет связь с:

1. Хранилищем информации о пользователях.
2. Хранилищем информации о всех заявках на добавление достопримечательностей.
3. Хранилищем информации о достопримечательностях.
4. Хранилищем категорий достопримечательностей.
5. Подсистемой регистрации и авторизации, неавторизованный пользователь не может отправлять заявки.
6. Подсистемой работы с картой (пользователь должен отметить местоположение предлагаемой достопримечательности).

Входные данные:

1. Имя пользователя.
2. Название достопримечательности.
3. Описание достопримечательности.
4. Категория достопримечательности из имеющихся в приложении, либо указывается своя категория.
5. Местоположение достопримечательности (пользователь имеет возможность отметить местоположение достопримечательности на карте меткой, по метке будут получены координаты и адрес достопримечательности).

Выходные данные: составленная заявка на добавление достопримечательности, которая добавлена в хранилище заявок.

Требования к функционалу подсистемы работы с картой (отображение достопримечательностей на карте, построение маршрута)

Подсистема имеет следующий функционал:

1. Отобразить достопримечательности города на карте (с учетом интересов пользователя, если он авторизован).
2. Построить маршрут по выбранным достопримечательностям.
3. Сохранить список выбранных достопримечательностей.

Подсистема имеет связь с:

1. Хранилищем информации о достопримечательностях.
2. Хранилищем информации о пользователях.
3. Хранилищем сохраненных маршрутов посещения достопримечательностей.
4. Хранилищем категорий достопримечательностей.
5. Внешней системой API карты.
6. Подсистемой регистрации и авторизации.
7. Подсистемой взаимодействия с сохраненными маршрутами посещения достопримечательностей.

Входные данные для отображения карты города:

1. Название города.
2. Предпочтения пользователя (если он авторизован).

Входные данные для построения маршрута:

1. Список выбранных достопримечательностей.
2. Предпочтения пользователя (если он авторизован).

Входные данные для сохранения маршрута посещения достопримечательностей:

1. Список выбранных достопримечательностей.
2. Имя пользователя.
3. Дополнительные условия построения маршрута (время начала путешествия, изначальное местоположение, способ передвижения).

Выходные данные:

1. Карта города с достопримечательностями.
2. Построенный маршрут посещения достопримечательностей.
3. Сохраненный маршрут посещения достопримечательностей.

Требования к функционалу подсистемы создания отзывов, предложений по улучшению и обращений в техподдержку

Подсистема имеет следующий функционал:

1. Просмотр списка написанных ранее пользователем отзывов, предложений по улучшению и обращений в техподдержку.
2. Предоставить форму для написания отзыва.
3. Отправить отзыв.

Подсистема имеет связь с:

1. Хранилищем информации о пользователях.
2. Хранилище информации о всех отзывах, предложениях по улучшению, обращениях в техподдержку.
3. Подсистемой регистрации и авторизации.

Входные данные:

1. Имя пользователя.
2. Отзыв.

Выходные данные: отзыв о достопримечательности, добавленный в хранилище информации о всех отзывах, предложениях по улучшению, обращениях в техподдержку.

Требования к функционалу подсистемы взаимодействия с построенными маршрутами посещения достопримечательностей

Подсистема имеет следующий функционал:

1. Отобразить список сохраненных маршрутов.
2. Редактировать сохраненные маршруты.

3. Удалить сохраненные маршруты.

Подсистема имеет связь с:

1. Хранилищем сохраненных маршрутов посещения достопримечательностей.
2. Хранилищем информации о пользователях.
3. Подсистемой взаимодействия с личным кабинетом.
Входные данные: имя пользователя.

Выходные данные: изменения в хранилище сохраненных маршрутов посещения достопримечательностей.

Требования к функционалу подсистемы модерирования приложения

Подсистема имеет следующий функционал:

1. Просмотр списка нерассмотренных отзывов.
2. Разрешить или запретить опубликование отзыва.
3. Удалить отзыв.

Подсистема имеет связь с:

1. Хранилище информации о всех отзывах, предложениях по улучшению, обращениях в техподдержку.
2. Подсистемой регистрации и авторизации.

На вход подается отзыв из хранилища с отзывами.

Выходные данные: отзыву ставится пометка «одобрен» или «отклонен».

При удалении одобренный ранее отзыв удаляется из хранилища отзывов.

Требования к функционалу подсистемы администрирования приложения

Подсистема должна иметь следующий функционал:

1. Администрирование достопримечательностей:
 - a. Редактировать информацию о достопримечательностях.
 - b. Добавить достопримечательность.

- c. Удалить достопримечательность.
 - d. Добавить категорию достопримечательностей.
 - e. Удалить категорию достопримечательностей.
2. Рассмотрение заявок на добавление достопримечательностей:
- a. Просмотр списка заявок на добавление достопримечательностей.
 - b. Одобрить заявку.
 - c. Отклонить заявку.
 - d. Прокомментировать заявку.
3. Рассмотрение предложений по улучшению и обращений в техподдержку:
- a. Просмотр списка предложений по улучшению и обращений в техподдержку.
 - b. Ответить на предложение по улучшению.
 - c. Ответить на обращение в техподдержку.

Подсистема имеет связь с:

- 1. Хранилищем информации о достопримечательностях.
- 2. Хранилищем информации о всех заявках на добавление достопримечательностей.
- 3. Хранилищем категорий достопримечательностей.
- 4. Хранилище информации о всех отзывах, предложениях по улучшению, обращениях в техподдержку.
- 5. Подсистемой регистрации и авторизации, поскольку вносить изменения может только администратор.

Входные данные для редактирования: информация о достопримечательности из базы данных (название достопримечательности, описание достопримечательности, категория, местоположение достопримечательности (адрес, координаты)).

Входные данные для удаления:

1. Название города.
2. Название достопримечательности.
3. Адрес достопримечательности (если в городе имеется несколько достопримечательностей с одинаковым названием в разных местах города).

Входные данные для добавления:

1. Название достопримечательности.
2. Описание достопримечательности.
3. Категория достопримечательности может определяться администратором на основе:
 - a. Имеющегося описания достопримечательности.
 - b. Категории, указанной пользователем.
 - c. Имеющегося списка категорий в приложении.

При отсутствии подходящей категории достопримечательность может быть добавлена только после добавления соответствующей категории.

4. Местоположение достопримечательности (администратор имеет возможность отметить местоположение достопримечательности на карте меткой, по метке будут получены координаты и адрес достопримечательности).

Выходные данные: добавленная\измененная\удаленная в хранилище данных информация о достопримечательности

Входные данные при рассмотрении заявок: заявка на добавление достопримечательности.

Выходные данные: сохранение заявки с пометкой «одобрена» или «отклонена» и комментарием администратора.

Входные данные на добавление/удаление категории: название категории (так как список официальных категорий достопримечательностей отсутствует, то добавляя категорию достопримечательностей, администратор должен руководствоваться источниками информации, которые он считает правильными).

Выходные данные: добавленная/удаленная категория достопримечательностей.

2.2. НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

Требования к пользователям

Каждый пользователь, за исключением неавторизованного пользователя, работающий в системе, обязан: зарегистрироваться в системе и заполнить следующие обязательные поля в регистрационной форме:

1. Имя пользователя.
2. E-mail пользователя.
3. Пароль для входа в систему.
4. Повторный пароль для входа в систему.

Пользователи приложения должны обладать базовыми навыками владения компьютером.

В системе должны быть выделены следующие типы пользователей:

1. Неавторизованный пользователь.
2. Авторизованный пользователь.
3. Модератор.
4. Администратор.

Неавторизованный пользователь имеет возможность воспользоваться только построителем маршрутов, настроить индивидуальные предпочтения в личном кабинете он не может, только указывать их при создании маршрута. Так же такой пользователь не может сохранить построенный маршрут.

Авторизованный пользователь должен иметь возможность воспользоваться любым функционалом системы, за исключением функций модерирования и администрирования.

Модератор должен иметь возможность воспользоваться функционалом системы модерирования.

Администратор должен иметь возможность воспользоваться функционалом системы администрирования.

Требования к обучению персонала

Пользователь, имеющий права модератора отзывов должен быть ознакомлен с критериями одобрения отзывов и функционалом подсистемы модерирования приложения (редактирования/удаления отзывов).

Пользователь, имеющий права администратора должен быть ознакомлен с критериями одобрения заявок на добавление достопримечательностей и функционалом подсистемы администрирования приложения (редактирование информации о достопримечательностях, добавление/удаление достопримечательностей, добавление/удаление категорий достопримечательностей).

Требования к системе безопасности

Система должна обладать следующими средствами обеспечения безопасности:

1. Пользователь при входе в систему обязан указывать логин и пароль.
2. Пользователь при регистрации в системе обязан указывать пароль два раза (для подтверждения).
3. Пароль должен храниться в зашифрованном виде.

Требования к объему данных

Всего в системе имеются следующие хранилища данных:

1. Хранилище информации о пользователях.
2. Хранилище информации о достопримечательностях.
3. Хранилище сохраненных маршрутов посещения достопримечательностей.
4. Хранилище информации о всех отзывах, предложениях по улучшению,

обращениях в техподдержку.

5. Хранилище информации о всех заявках на добавление достопримечательностей.
6. Хранилище категорий достопримечательностей.

Возможный хранимый объем данных определяется характеристиками аппаратного обеспечения.

Требования к временным характеристикам

При стабильном интернет соединении отклик от приложения не должен превышать более 5 секунд, если количество пользователей одновременно (в течение 1 секунды) выполняющих запрос к приложению не превышает 50.

Требования к качественным характеристикам

Разрабатываемое приложение должно быть веб-приложением и поддерживать работу с не менее чем с 10 наиболее популярными браузерами в России. Наиболее популярные браузеры определяются на основе статистики Яндекса [17].

3. ПРОЕКТИРОВАНИЕ

3.1. АРХИТЕКТУРА ПРЕДЛАГАЕМОГО РЕШЕНИЯ

При разработке приложения используется архитектура MVC. Архитектура MVC позволяет разделить код приложения на 3 части: Модель (Model), Вид или Представление (View) и Контроллер (Controller).

Модель содержит в себе всю логику приложения, она хранит и обрабатывает данные, при этом не взаимодействуя с пользователем напрямую (обратиться к Модели можно только из кода, вызывая ее функции). Например, сохранение информации в БД.

Представление отображает данные, которые ему передали. В веб-приложении оно обычно состоит из HTML-шаблонов. Представление - это код, который отвечает за отображение информации на экране, отрисовку кнопок и других элементов интерфейса.

В PHP оно не должно обращаться к внешним переменным (`$_GET` и другие), его задача просто отобразить те данные, которые ему передали.

Контроллер отвечает за выполнение запросов, пришедших от пользователя. В веб-приложении обычно контроллер разбирает параметры HTTP-запроса из `$_POST/$_GET`, обращается к модели, чтобы получить или изменить какие-то данные, и в конце вызывает Представление, чтобы отобразить результат выполнения запроса. Число контроллеров определяется числом разделов или страниц сайта [18].

Взаимодействие между компонентами MVC отображено на рисунке 1.

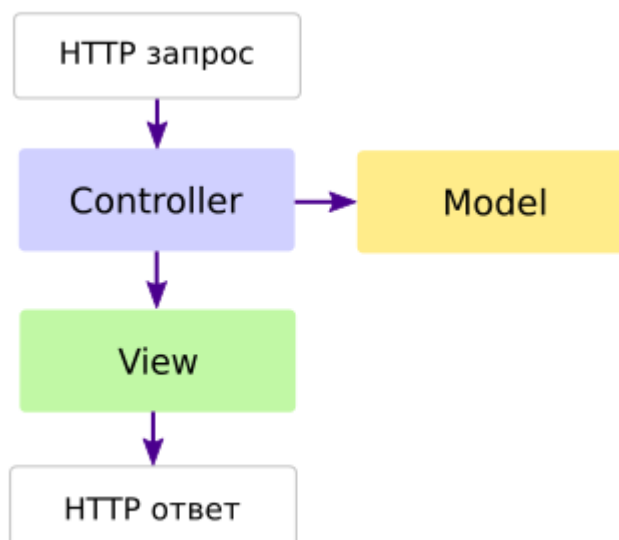


Рисунок 3.1.1 – Взаимодействие между компонентами MVC

В таблице 3.1.1 представлены все контроллеры приложения и функции, которые они выполняют, а также их взаимодействие с моделями и представлениями.

Таблица 3.1.1 – Контроллеры веб приложения

Наименование контроллера	Функции	Описание функций	Взаимодействие с другими компонентами	
			Модели	Представления
Administration	menu	Вызов представления меню администрирования приложения	Отсутствует	admin_panel

Продолжение таблицы 3.1.1

Наименование контроллера	Функции	Описание функций	Взаимодействие с другими компонентами	
			Модели	Представления
Administration	list_feedbacks	Вызов представления списка предложений по улучшению или обращений в техподдержку	feedback_model	list_unverified_techsupport, list_unverified_offers
	list_request_techsupport	Вызов list_feedbacks с передачей о желании вывода списка обращений в техподдержку	Отсутствует	Отсутствует
	list_offers	Вызов list_feedbacks с передачей о желании вывода предложений по улучшению	Отсутствует	Отсутствует

Продолжение таблицы 3.1.1

Наименование контроллера	Функции	Описание функций	Взаимодействие с другими компонентами	
			Модели	Представления
Administration	check_request	Вызов представления с обращением в техподдержку	feedback_model	check_request
	check_offer	Вызов представления с предложением по улучшению	feedback_model	check_offer
	save_answer_on_request	Обработка формы ответа на обращение в техподдержку	feedback_model	list_request_techsupport
	save_answer_on_offer	Обработка формы ответа предложение по улучшению	feedback_model	list_offers
	list_categories	Вывод списка имеющихся категорий достопримечательностей	category_model	list_categories

Продолжение таблицы 3.1.1

Наименование контроллера	Функции	Описание функций	Взаимодействие с другими компонентами	
			Модели	Представления
Administration	save_category	Обработка формы добавления новой категории	category_model	Отсутствует
	delete_category	Обработка удаления категории	category_model	Отсутствует
	list_requests	Вызов представления списка непроверенных заявок на добавление достопримечательностей	request_model	list_unverified_requests_for_add_place
	check_request_for_add_place	Вызов представления заявки на добавление достопримечательности	request_model	list_requests, user_request_for_adding_place

Продолжение таблицы 3.1.1

Наименование контроллера	Функции	Описание функций	Взаимодействие с другими компонентами	
			Модели	Представления
Administration	save_answer_for_request_for_added_place	Обработка формы ответа на заявку о добавлении достопримечательности	request_model	list_requests
	add_place	Вызов представления с формой добавления достопримечательности	category_model, request_model, place_model, profile_model	add_place
	save_place	Обработка формы добавления достопримечательности	place_model, category_model, country_model, city_model	Отсутствует

Продолжение таблицы 3.1.1

Наименование контроллера	Функции	Описание функций	Взаимодействие с другими компонентами	
			Модели	Представления
Administration	find_place	Вызов представления поиска нужной достопримечательности	place_model, city_model, country_model	find_place
	find	Обработка формы поиска достопримечательности	place_model	Отсутствует
	delete_place	Вызов представления удаления достопримечательности	place_model	confirm_delete
	delete	Обработка формы удаления достопримечательности	place_model	Отсутствует

Продолжение таблицы 3.1.1

Наименование контроллера	Функции	Описание функций	Взаимодействие с другими компонентами	
			Модели	Представления
Administration	edit_place	Вызов представления редактирования достопримечательности	place_model, category_model, city_model, country_model	edit_place
	save_update_place	Обработка формы редактирования достопримечательности	place_model, category_model, country_model, city_model	Отсутствует
Authorization	registration	Вызов представления регистрации	Отсутствует	registration
	check_form_registration	Обработка формы регистрации	profile_model	Отсутствует
	login	Вызов представления авторизации	Отсутствует	login
	check_login	Обработка формы авторизации	profile_model	Отсутствует

Продолжение таблицы 3.1.1

Наименование контроллера	Функции	Описание функций	Взаимодействие с другими компонентами	
			Модели	Представления
Authorization	forgot_password	Вызов представления восстановления забытого пароля	Отсутствует	forgot_password
	check_forgot_password	Обработка формы забытого пароля	profile_model	message_sent_successfully
	change_password	Вызов представления смены пароля	Отсутствует	change_password
	check_change_password	Обработка формы смены пароля	profile_model	Отсутствует
Category	list_my_category	Вызов представления списка предпочитаемых категорий достопримечательностей пользователем	category_model	list_user_categories

Продолжение таблицы 3.1.1

Наименование контроллера	Функции	Описание функций	Взаимодействие с другими компонентами	
			Модели	Представления
Category	add_category	Обработка добавления новой категории в список предпочитаемых	category_model	Отсутствует
	delete	Обработка удаления категории из списка предпочитаемых	category_model	Отсутствует
Cities	list	Вызов представления со списком доступных городов для посещения	place_model, city_model	cities
Feedback	create	Вызов представления создания отзыва, предложения по улучшению или обращения в техподдержку	feedback_model	feedback

Продолжение таблицы 3.1.1

Наименование контроллера	Функции	Описание функций	Взаимодействие с другими компонентами	
			Модели	Представления
Feedback	save_feedback	Обработка формы обратной связи	feedback_model	Отсутствует
	list_my_feedbacks	Вызов представления списка составленных пользователем сообщений	feedback_model	edit_feedback
	edit	Вызов представления редактирования сообщения	feedback_model	edit_feedback
	save_edit	Обработка формы редактирования	feedback_model	Отсутствует
	delete	Обработка удаления отзыва, предложения по улучшению или обращения в техподдержку	feedback_model	Отсутствует

Продолжение таблицы 3.1.1

Наименование контроллера	Функции	Описание функций	Взаимодействие с другими компонентами	
			Модели	Представления
Main	index	Вызов представления главной страницы	main_model	index
Moderation	list_reviews	Вызов представления списка отзывов	feedback_model	list_unverified_reviews
	check_review	Вызов представления просмотра отзыва	feedback_model	check_review
	save_review	Обработка формы проверки отзыва	feedback_model	Отсутствует
Profile	personal_account	Вызов представления личного кабинета	profile_model	about_me
	save_settings_personal_account	Обработка изменений в личном кабинете	profile_model	Отсутствует
	change_password	Вызов представления смены пароля	Отсутствует	set_new_password

Продолжение таблицы 3.1.1

Наименование контроллера	Функции	Описание функций	Взаимодействие с другими компонентами	
			Модели	Представления
Profile	set_new_password	Обработка изменения пароля	Отсутствует	profile_model
Request	list_my_request	Вызов представления списка заявок на добавление достопримечательностей	request_model	list_user_request
	create	Вызов представления создания заявки	category_model	request
	add_request	Обработка формы создания заявки	request_model, category_model	Отсутствует
	delete	Обработка удаления заявки	request_model	Отсутствует
	edit	Вызов представления редактирования заявки	request_model, category_model	edit_request

Продолжение таблицы 3.1.1

Наименование контроллера	Функции	Описание функций	Взаимодействие с другими компонентами	
			Модели	Представления
Request	update_request	Обработка формы редактирования заявки	request_model, category_model	Отсутствует
Route_builder	build_route	Вызов представления построения маршрута	category_model, profile_model, city_model, place_model	route_builder
	save_route	Обработка составленного маршрута	place_model	Отсутствует

Таблица с моделями веб-приложения приведена в таблице 3.1.2.

Таблица 3.1.2 – Модели веб-приложения:

Наименование модели	Функции	Описание функций
category_model	get_user_category	Получение из базы данных предпочитаемых пользователем категорий достопримечательностей
	get_all_category	Получение всех категорий достопримечательностей
	add_category_preferred_user	Добавление категории в предпочтения пользователя

Продолжение таблицы 3.1.2

Наименование модели	Функции	Описание функций
category_model	delete_category_preferred_user	Удаление категории из предпочтений пользователя
	get_category	Получение конкретной категории достопримечательностей
	add_category	Добавление новой категории
	delete_category	Удаление категории
	remove_category_from_users_preferences	Удаление категории из предпочтений всех пользователей
	get_count_places_appropriate_category	Получение количества достопримечательностей конкретной категории
city_model	get_cities	Получение списка городов
	check_city	Проверка наличия указанного города в указанной стране
	get_city	Получение города по указанной стране и названию города
	add_city	Добавление нового города в базу данных
country_model	check_country	Проверка наличия указанной страны в базе данных
	add_country	Добавление новой страны
	get_country	Получение указанной страны из базы данных

Продолжение таблицы 3.1.2

Наименование модели	Функции	Описание функций
country_model	get_countries	Получение списка всех стран в базе данных
feedback_model	get_types_feedback	Получение видов сообщений от пользователей
	save_feedback	Сохранение сообщения от пользователя
	get_user_feedbacks	Получение списка сообщений от указанного пользователя
	get_statuses_feedback	Получение статусов сообщений
	get_feedback	Получить конкретное сообщение от пользователя
	save_update_feedback	Сохранение изменения сообщения
	delete_feedback	Удаление сообщения
	get_unverified_feedbacks	Получение списка нерассмотренных сообщений
	save_answer_on_feedback	Сохранение ответа на сообщение
	get_review	Получение конкретного отзыва
	get_request	Получение конкретного обращения в техподдержку
	get_offer	Получение конкретного предложения по улучшению
Main_model	getReviews	Получение всех проверенных отзывов, либо одного конкретного

Продолжение таблицы 3.1.2

Наименование модели	Функции	Описание функций
Main_model	getNameUser	Получение имени пользователя по его ид
place_model	add_place	Добавление достопримечательности
	check_place	Проверка наличия достопримечательности с указанным именем и адресом
	check_update_place	Проверка наличия достопримечательности с указанным именем и адресом при обновлении
	get_place	Получение достопримечательности
	get_place_on_id	Получение достопримечательности по ид
	delete	Удаление достопримечательности
	update_place	Изменение достопримечательности
	get_work_schedules	Получение типов графиков работы
	get_days_of_week	Получение списка дней недели
	get_city	Получение города по ид
	add_work_schedule	Добавление рабочего дня в расписание графика работы достопримечательности
	delete_work_schedule	Удаление расписания работы достопримечательности

Продолжение таблицы 3.1.2

Наименование модели	Функции	Описание функций
place_model	update_work_schedule	Изменение расписания работы достопримечательности
	get_opening_hours_place	Получение расписания работы достопримечательности
	get_all_place	Получение списка всех достопримечательностей
	get_all_place_for_view	Получение списка всех достопримечательностей в удобном для восприятия виде
	get_number_place_in_city	Получение количества достопримечательностей в указанном городе
profile_model	get_name_transport	Получение названия способа передвижения по указанному ид
	get_all_transport	Получение всех способов передвижения
	get_all_time	Получение списка времени
	save_settings_personal_account	Сохранение настроек пользователя в личном кабинете
request_model	get_statuses_request	Получение списка статусов проверки заявок на добавление достопримечательностей
	get_user_requests	Получение списка всех заявок пользователя

Продолжение таблицы 3.1.2

Наименование модели	Функции	Описание функций
request_model	check_place	Проверка наличия достопримечательности
	check_update_place	Проверка наличия достопримечательности при изменении заявки
	add_request_with_selected_category	Добавление заявки при указании категории достопримечательности из имеющихся в списке
	add_request_with_user_category	Добавление заявки при написании пользователем своей категории достопримечательности
	get_id_user	Получение ид пользователя по ид заявки
	delete_request	Удаление заявки
	get_request	Получение заявки
	update_request_with_selected_category	Изменение заявки при указании категории достопримечательности из имеющихся в списке
	update_request_with_user_category	Изменение заявки при написании пользователем своей категории достопримечательности
	get_unverified_requests	Получение списка непроверенных заявок

Продолжение таблицы 3.1.2

Наименование модели	Функции	Описание функций
request_model	save_answer_for_request_f or_add_place	Сохранение ответа на заявку пользователя

Таблица с представлениями веб-приложения приведена в таблице 3.1.3.

Таблица 3.1.3 – Представления веб-приложения:

Наименование представления	Назначение
add_place	Форма добавления достопримечательности
admin_panel	Меню системы администрирования
check_offer	Форма просмотра предложения по улучшению
check_request	Форма просмотра обращения в техподдержку
confirm_delete	Подтверждения удаления достопримечательности
edit_place	Форма редактирования достопримечательности
find_place	Форма поиска достопримечательности
list_categories	Список категорий достопримечательностей
list_unverified_offers	Список нерассмотренных предложений по улучшению

Продолжение таблицы 3.1.3

Наименование представления	Назначение
list_unverified_requests_for_add_place	Список непроверенных заявок на добавление достопримечательностей
list_unverified_techsupport	Список нерассмотренных обращений в техподдержку
user_request_for_adding_place	Форма проверки пользовательской заявки на добавление достопримечательности
change_password	Форма смены пароля при сбросе
forgot_password	Форма восстановления пароля
login	Форма авторизации
message_sent_successfully	Сообщение о сбросе пароля
registration	Форма регистрации
list_user_categories	Список предпочитаемых пользователем категорий достопримечательностей
cities	Список доступных городов для посещения
edit_feedback	Форма редактирования отзыва, предложения по улучшению или обращения в техподдержку
feedback	Форма написания отзыва, предложения по улучшению или обращения в техподдержку

Продолжение таблицы 3.1.3

Наименование представления	Назначение
list_feedbacks	Список написанных отзывов, предложений по улучшению или обращений в техподдержку пользователем
index	Главная страница
check_review	Форма проверки отзыва пользователя
list_unverified_reviews	Список непроверенных отзывов
about_me	Личный кабинет пользователя
set_new_password	Форма смены пароля через личный кабинет
edit_request	Форма редактирования заявки на добавление достопримечательности
list_user_request	Список заявок пользователя на добавление достопримечательности
request	Форма создания заявки на добавление достопримечательности
route_builder	Страница построения\редактирования маршрута

3.2. АЛГОРИТМЫ РЕШЕНИЯ ЗАДАЧИ

Построение маршрута сводится к решению задачи коммивояжёра. Задача коммивояжёра — пожалуй, одна из самых известных оптимизационных задач. Ее цель заключается в нахождении самого выгодного маршрута (кратчайшего,

самого быстрого, наиболее дешевого), проходящего через все заданные точки (пункты, города) по одному разу, с последующим возвратом в исходную точку.

Условия задачи должны содержать критерий выгодности маршрута (т. е. должен ли он быть максимально коротким, быстрым, дешевым или все вместе), а также исходные данные в виде матрицы затрат (расстояния, стоимости, времени и т. д.) при перемещении между рассматриваемыми пунктами.

Особенности задачи в том, что она довольно просто формулируется и найти хорошие решения для нее также относительно просто, но вместе с тем поиск действительно оптимального маршрута для большого набора данных — непростой и ресурсоемкий процесс.

Для решения задачи коммивояжера ее надо представить, как математическую модель. При этом исходные условия можно записать в формате матрицы — таблицы, где строкам соответствуют города отправления, столбцам — города прибытия, а в ячейках указываются расстояния (время, стоимость) между ними [19-21].

Виды задачи коммивояжера по симметричности ребер графа [19-21]:

1. Симметричная — все пары ребер, соединяющих одни и те же вершины, имеют одинаковую длину (т. е. граф, представляющий исходные данные задачи, является неориентированным). Иными словами, длина прямого пути от города А до города В и длина обратного пути от города В до города А, одинаковы.
2. Асимметричная — длина пар ребер, соединяющих одни и те же города, может различаться (ориентированный граф). Для этого типа задачи коммивояжера прямой путь, например, из города А в город В может быть короче или длиннее обратного пути из города В в город А.

Типы задачи коммивояжера по замкнутости маршрута [19-21]:

1. Замкнутая — нахождение кратчайшего пути, проходящего через все вершины по одному разу с последующим возвратом в точку старта (маршрут получается замкнутым, закольцованным).
2. Незамкнутая — нахождение кратчайшего пути, проходящего через все вершины по одному разу и без обязательного возврата в исходную точку (маршрут получается разомкнутым).

В нашем случае нужно решить асимметричную замкнутую задачу коммивояжера.

Методы решения задачи коммивояжера довольно разнообразны и различаются применяемым инструментарием, точностью находимого решения и сложностью требуемых вычислений. Вот некоторые из них [22]:

1. Полный перебор (метод «грубой силы», англ. «Brute Force») — заключается в последовательном рассмотрении всех возможных маршрутов и выборе из них оптимального. Метод самый простой и точный, но неэффективный и при большом количестве городов его применение становится затруднительным ввиду значительных затрат времени и ресурсов на перебор огромного количества вариантов решения задачи.
2. Случайный перебор — в этом случае вычисляются не все возможные варианты маршрута, а лишь некоторые выбранные в случайном порядке (например, с помощью генератора случайных чисел). Из рассмотренных вариантов затем выбирается наилучший. Конечно, вероятнее всего полученное решение не будет оптимальным (впрочем, оно не будет и самым худшим), но зато данный метод требует меньших затрат времени и вычислительных ресурсов, а потому в некоторых случаях его применение оправдано.

3. Динамическое программирование — ключевая идея заключается в вычислении и запоминании пройденного пути от исходного города до всех остальных, последующем прибавлении к нему расстояний от текущих городов до оставшихся, и так далее. По сравнению с полным перебором этот метод позволяет существенно сократить объем вычислений.
4. Жадные алгоритмы — основаны на нахождении локально оптимальных решений на каждом этапе вычислений и допущении, что найденное таким образом итоговое решение будет глобально оптимальным. Т. е. на каждой итерации выбирается лучший участок пути, который включается в итоговый маршрут. Метод простой, но его большой недостаток в том, что может возникнуть ситуация, когда окажется, что начальная и конечная точки маршрута разнесены далеко друг от друга и их придется соединять длинным отрезком пути, что значительно снизит эффективность решения.
5. Муравьиный алгоритм — эвристический метод, основанный на моделировании поведения муравьев, ищущих пути от своей колонии к источникам пищи. Первую версию такого алгоритма предложил доктор наук Марко Дориго в 1992 году. Этот метод позволяет относительно быстро найти хорошее, но не обязательно оптимальное решение.
6. Генетический алгоритм — еще один эвристический метод, заключающийся в случайном подборе и комбинировании исходных параметров с использованием механизмов имитирующих естественный отбор в процессе эволюции (наследование, мутации, кроссинговер). Несмотря на довольно широкие возможности применения (и не только в логистике), этот метод часто становится объектом критики.

7. Метод ветвей и границ — один из методов дискретной оптимизации, являющийся развитием метода полного перебора, но отличающийся от него отсевом в процессе вычисления подмножеств неэффективных решений. Впервые был предложен в 1960 году английским профессором Алисой Лэнд и австралийским математиком Элисон Дойг.

Исключаем метод полного перебора, он самый точный, но в тоже время самый медленный по времени, поскольку нужно просмотреть все варианты путей, а при большом количестве узлов вычисления могут занять очень большое время, что неприемлемо для приложения. Случайный перебор так же не подходит, поскольку является аналогом полного перебора, при котором случайным образом исключаются некоторые маршруты, это приведет к непредсказуемому результату работы построителя. Метод динамического программирования не рассчитан на большое количество узлов.

Выбирать будем между методом ветвей и границ, жадным алгоритмом, генетическим алгоритмом и муравьиным алгоритмом.

Преимущество метода ветвей и границ в высокой точности получения оптимального маршрута, поскольку является развитием полного перебора, но отличающийся отсевом в процессе вычисления подмножеств неэффективных решений [22].

К недостаткам можно отнести:

1. Начальная точка маршрута выбирается алгоритмом.
2. Нельзя исключать узлы в процессе вычислений, что необходимо будет делать, если выбранная достопримечательность пользователем в данный момент закрыта (например, музей).

3. Вычисления могут выполняться долго, поскольку наихудшим вариантом является полный перебор всех возможных путей.
4. Сложность алгоритма

Жадные алгоритмы являются полной противоположностью методу ветвей и границ.

Недостаток в том, что может возникнуть ситуация, когда окажется, что начальная и конечная точки маршрута разнесены далеко друг от друга и их придется соединять длинным отрезком пути, что значительно снизит эффективность решения.

Цель работы генетического алгоритма заключается в нахождении лучшего по сравнению с имеющимся, а не оптимального решения задачи. Проблема достижения оптимума при этом является вторичной. Преимущества генетических алгоритмов особенно хорошо видны при рассмотрении их в сравнении с традиционными методами [23].

Недостатки генетического алгоритма [23]:

1. Не гарантируется получение оптимального решения.
2. Эффективно сформулировать задачу, определить критерий отбора хромосом (задать код) и другие параметры ГА может только специалист.
3. Постановка задачи в терминах ГА не дает возможности проанализировать статистическую значимость получаемого с их помощью решения.
4. Достаточно высокая вычислительная ресурсоемкость ГА приводит к тому, что в ходе моделирования эволюции многие решения отбрасываются как неперспективные.

5. При временной сложности в среднем ниже, чем у лучших конкурирующих алгоритмов, но не более, чем на один порядок.
6. Невысокая эффективность на заключительных фазах моделирования эволюции, объясняемая тем, что механизмы поиска ГА не являются жестко ориентированными на скорейшее попадание в локальный оптимум.

Идея муравьиного алгоритма – моделирование поведения муравьёв, связанного с их способностью быстро находить кратчайший путь от муравейника к источнику пищи и адаптироваться к изменяющимся условиям, находя новый кратчайший путь. При своём движении муравей метит путь феромоном, и эта информация используется другими муравьями для выбора пути. Это элементарное правило поведения и определяет способность муравьёв находить новый путь, если старый оказывается недоступным [24].

Достоинства [24]:

1. Для TSP (Traveling Salesman Problem) сравнительно эффективны.
2. Для небольшого количества узлов TSP может быть решена полным перебором.
3. Работают лучше, чем другие глобальные оптимизации для TSP (нейронные сети, генетические алгоритмы).

Вывод:

Таким образом, построение маршрута будет осуществляться на основе муравьиного алгоритма, который будет дополнен анализом предпочтений пользователя, времени и режима работы достопримечательностей (если таковое имеется). Поскольку маршрут должен меняться в реальном времени, его программная реализация будет создаваться на языке JavaScript.

Критериями для выбора послужили следующие преимущества:

1. Быстрее чем метод полного перебора, метод ветвей и границ и метод случайного перебора.
2. Исходя из исследований МА обеспечивает поиск лучших результатов, чем ГА. И скорость сходимости выше на ранних итерациях. Когда дело доходит до вычислительных затрат, тщательное сравнение показывает, что МА сходится раньше и поддерживает исследовательское поведение в течение всего доступного времени выполнения. Когда число городов и пространство решения больше МА превосходит ГА по качеству результатов [25].
3. Уступает жадным алгоритмам по скорости получения результата, но имеет лучшую эффективность, потому что при использовании жадного алгоритма может возникнуть ситуация, когда окажется, что начальная и конечная точки маршрута разнесены далеко друг от друга и их придется соединять длинным отрезком пути, что значительно снижает эффективность решения, полученного жадным алгоритмом.

Рассмотрим подробнее муравьиный алгоритм [26].

Любой муравьиный алгоритм, независимо от модификаций, представим в следующем виде:

1. Создаём муравьёв.
2. Ищем решения.
3. Обновляем феромон.

Теперь рассмотрим каждый шаг в цикле более подробно.

1. Создаём муравьёв.

Стартовая точка, куда помещается муравей, зависит от ограничений, накладываемых условиями задачи. Потому что для каждой задачи способ размещения муравьёв является определяющим. Либо все они помещаются в одну точку, либо в разные с повторениями, либо без повторений.

На этом же этапе задаётся начальный уровень феромона. Он инициализируется небольшим положительным числом для того, чтобы на начальном шаге вероятности перехода в следующую вершину не были нулевыми.

2. Ищем решения.

Вероятность перехода из вершины i в вершину j определяется по следующей формуле:

$$P_{ij}(t) = \frac{\tau_{ij}^{\alpha} \left(\frac{1}{d_{ij}}\right)^{\beta}}{\sum \tau_{ij}^{\alpha} \left(\frac{1}{d_{ij}}\right)^{\beta}}$$

где τ_{ij} – уровень феромона;

α – константный параметр;

d – эвристическое расстояние;

β – константный параметр.

При $\alpha = 0$ выбор ближайшего города наиболее вероятен, то есть алгоритм становится жадным. При $\beta = 0$ выбор происходит только на основании феромона, что приводит к субоптимальным решениям.

Поэтому необходим компромисс между этими величинами, который находится экспериментально.

3. Обновляем феромон.

Уровень феромона обновляется в соответствии с приведённой формулой

$$\tau_{ij}(t+1) = (1 - \rho)\tau_{ij}(t) + \sum \frac{Q}{L_k}$$

где ρ – интенсивность испарения;

$L_k(t)$ – цена текущего решения для k -ого муравья;

Q – параметр, имеющий значение порядка цены оптимального решения.

То есть $\frac{Q}{L_k(t)}$ – феромон, откладываемый k -ым муравьём, использующим ребро (i,j) .

Применение муравьиных алгоритмов для задачи коммивояжёра.

Задача формулируется как задача поиска минимального по стоимости замкнутого маршрута по всем вершинам без повторений на полном взвешенном графе с n вершинами. Содержательно вершины графа являются городами, которые должен посетить коммивояжёр, а веса рёбер отражают расстояния

(длины) или стоимости проезда. Эта задача является NP-трудной, и точный переборный алгоритм её решения имеет факториальную сложность.

Моделирование поведения муравьёв связано с распределением феромона на тропе – ребре графа в задаче коммивояжёра. При этом вероятность включения ребра в маршрут отдельного муравья пропорциональна количеству феромона на этом ребре, а количество откладываемого феромона пропорционально длине маршрута. Чем короче маршрут, тем больше феромона будет отложено на его рёбрах, следовательно, большее количество муравьёв будет включать его в синтез собственных маршрутов. Моделирование такого подхода, использующего только положительную обратную связь, приводит к преждевременной сходимости – большинство муравьёв двигается по локально оптимальному маршруту. Избежать этого можно, моделируя отрицательную обратную связь в виде испарения феромона. При этом если феромон испаряется быстро, то это приводит к потере памяти колонии и забыванию хороших решений, с другой стороны, большое время испарения может привести к получению устойчивого локального оптимального решения.

Теперь с учётом особенностей задачи коммивояжёра, мы можем описать локальные правила поведения муравьёв при выборе пути:

1. Муравьи имеют собственную «память». Поскольку каждый город может быть посещён только один раз, то у каждого муравья есть список уже посещённых городов – список запретов. Обозначим через $J_{i,k}$ список городов, которые необходимо посетить муравью k , находящемуся в городе i .
2. Муравьи обладают «зрением» – видимость есть эвристическое желание посетить город j , если муравей находится в городе i . Будем

считать, что видимость обратно пропорциональна расстоянию между городами:

$$\eta_{ij} = \frac{1}{D_{ij}}$$

3. Муравьи обладают «обонянием» – они могут улавливать след феромона, подтверждающий желание посетить город j из города i на основании опыта других муравьёв. Количество феромона на ребре (i,j) в момент времени t обозначим через $\tau_{ij}(t)$

На этом основании мы можем сформулировать вероятностно-пропорциональное правило, определяющее вероятность перехода k -ого муравья из города i в город j :

$$\begin{cases} P_{ij,k}(t) = \frac{[\tau_{ij}(t)]^\alpha * [\eta_{ij}]^\beta}{\sum_{l \in J_{i,k}} [\tau_{il}(t)]^\alpha * [\eta_{il}]^\beta}, j \in J_{i,k} \\ P_{ij,k}(t) = 0, j \notin J_{i,k}; \end{cases}$$

где τ_{ij} – уровень феромона;

α – константный параметр;

η – видимость, обратно пропорциональная расстоянию между городами;

β – константный параметр.

При $\alpha = 0$ алгоритм вырождается до жадного алгоритма (будет выбран ближайший город). Заметим, что выбор города является вероятностным,

правило лишь определяет ширину зоны города j ; в общую зону всех городов $J_{i,k}$ бросается случайное число, которое и определяет выбор муравья. Правило не изменяется в ходе алгоритма, но у двух разных муравьёв значение вероятности перехода будут отличаться, т.к. они имеют разный список разрешённых городов.

Пройдя ребро (i,j) , муравей откладывает на нём некоторое количество феромона, которое должно быть связано с оптимальностью сделанного выбора. Пусть $T_k(t)$ есть маршрут, пройденный муравьём k к моменту времени t . Тогда откладываемое количество феромона может быть задано в виде:

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

где Q – параметр, имеющий значение порядка длины оптимального пути;

$L_k(t)$ – длина этого маршрута.

Правила внешней среды определяют, в первую очередь, испарение феромона. Тогда правило испарения имеет вид:

$$\tau_{ij}(t+1) = (1-\rho) * \tau_{ij}(t) + \Delta\tau_{ij}(t); \Delta\tau_{ij}(t) = \sum_{k=1}^m \Delta\tau_{ij,k}(t),$$

где $\rho \in [0,1]$ – коэффициент испарения;

m – количество муравьёв в колонии.

3.3. ОПИСАНИЕ ДАННЫХ

Схема базы данных с использованием UML-диаграмм представлена на рисунках 3.3.1 и 3.3.2.

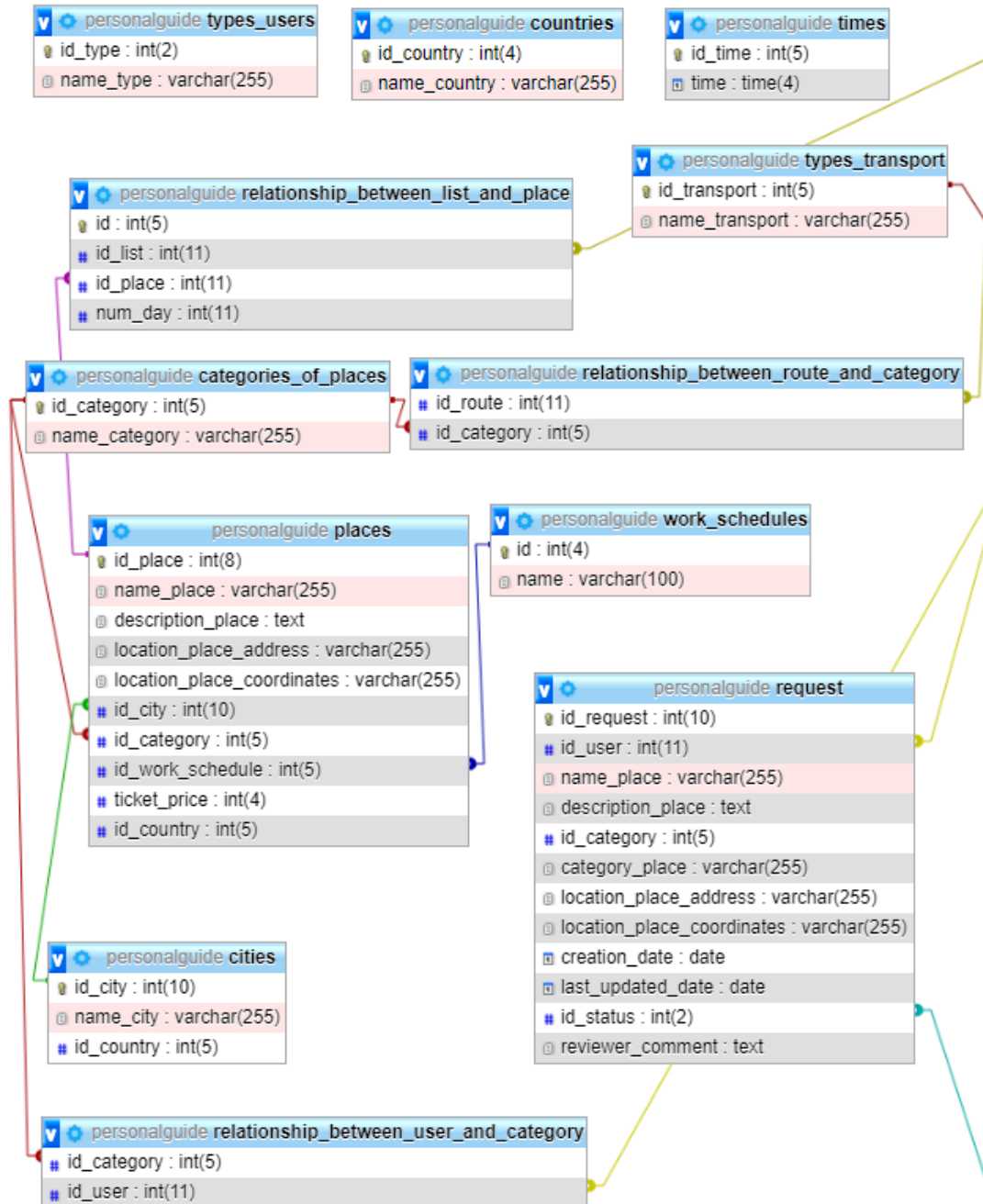


Рисунок 3.3.1 - Первая половина схемы базы данных с использованием UML-диаграмм

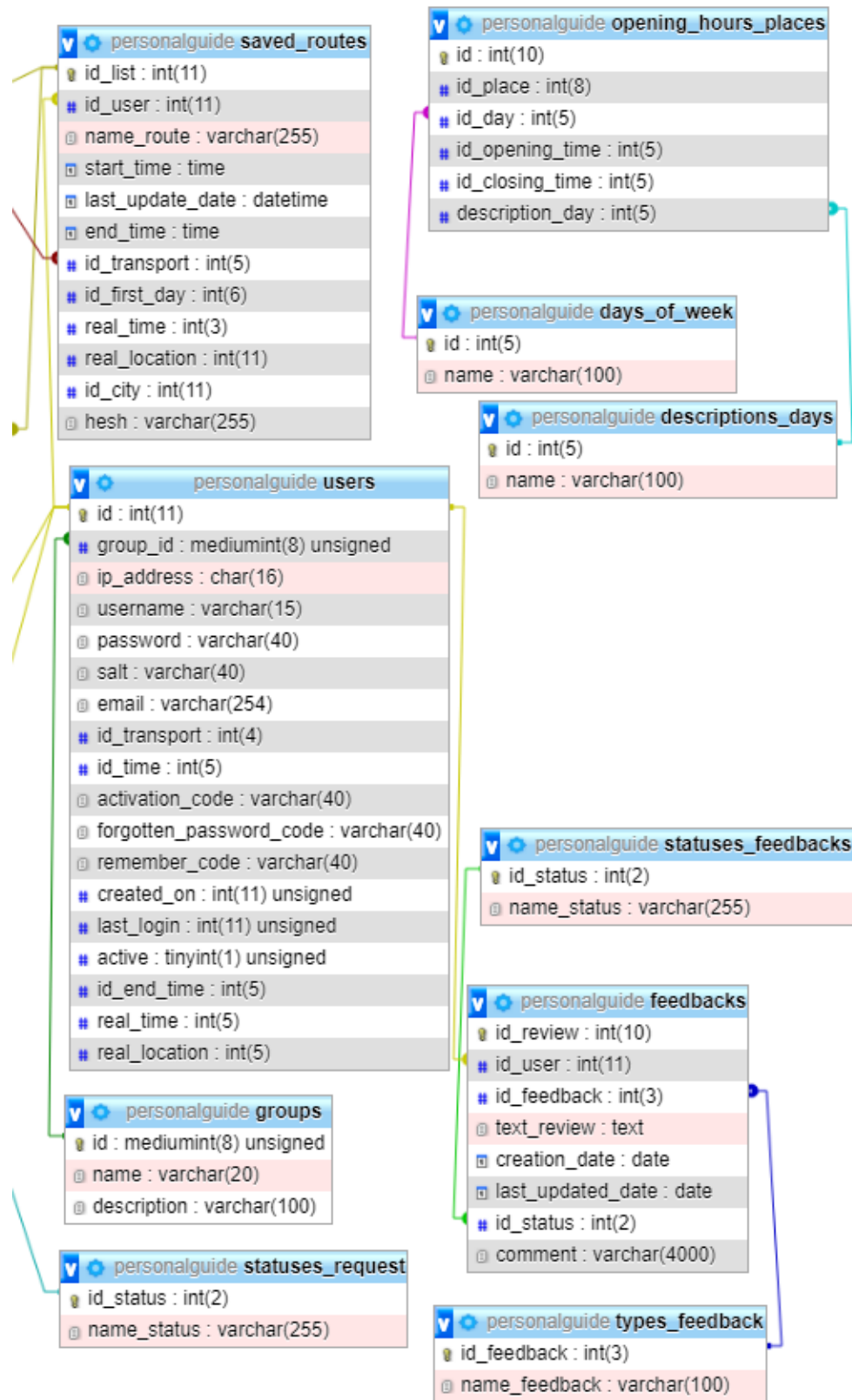


Рисунок 3.3.2 – Вторая половина схемы базы данных с использованием UML-диаграмм

В таблице 3.3.1 перечислены все элементы базы данных:

Таблица 3.3.1 – Таблицы базы данных

Наименование	Назначение
users	Пользователи
places	Достопримечательности
groups	Типы пользователей
countries	Страны
times	Список времен
relationship_between_list_and_place	Связывает сохраненные маршруты с достопримечательностями
categories_of_places	Категории достопримечательностей
saved_routes	Сохраненные маршруты
cities	Города
relationship_between_user_and_category	Связывает таблицу категорий достопримечательностей с таблицей пользователей
work_schedules	Графики работы
types_feedback	Виды сообщений обратной связи
feedbacks	Сообщения обратной связи
statuses_request	Состояния просмотра заявок
statuses_feedbacks	Состояния просмотра сообщений обратной связи
days_of_week	Дни недели
descriptions_days	Описание дней (рабочий или выходной)

Продолжение таблицы 3.3.1

Наименование	Назначение
opening_hours_places	Графики работы достопримечательностей (если они не доступны круглосуточно)

Поля таблицы «users» приведены в таблице 3.3.2:

Таблица 3.3.2 – Описание полей таблицы «users»

Наименование	Тип данных	Назначение
id	int(11)	Идентификатор пользователя (первичный ключ)
group_id	mediumint(8)	Группа пользователя (вторичный ключ)
ip_address	char(16)	Ip адресс
username	varchar(15)	Имя пользователя
password	varchar(40)	Пароль пользователя
email	varchar(254)	Почта пользователя
id_transport	int(4)	Идентификатор предпочитаемого способа передвижения (вторичный ключ)
id_time	int(5)	Идентификатор предпочитаемого времени начала путешествия (вторичный ключ)
forgotten_password_code	varchar(40)	Код при забытом пароле
remember_code	varchar(40)	Код для запоминания пользователя
created_on	int(11)	Время регистрации
last_login	int(11)	Время последней

		авторизации
--	--	-------------

Продолжение таблицы 3.3.2

Наименование	Тип данных	Назначение
Id_end_time	Int(5)	Идентификатор предпочитаемого времени окончания путешествия
real_location	int(3)	Статус учета реального времени при построении маршрута
real_location	int(5)	Статус учета реального местоположения при построении маршрута

Поля таблицы «places» приведены в таблице 3.3.3:

Таблица 3.3.3 – Описание полей таблицы «places»

Наименование	Тип данных	Назначение
id_place	int(8)	Идентификатор достопримечательности (первичный ключ)
name_place	varchar(255)	Наименование достопримечательности
description_place	text	Описание достопримечательности
location_place_address	varchar(255)	Адрес
location_place_coordinates	varchar(255)	Координаты
id_city	int(10)	Ид города (вторичный ключ)
id_category	int(5)	Ид категории (вторичный ключ)
id_work_schedule	int(5)	Ид графика работы (вторичный ключ)
ticket_price	int(5)	Цена билета
id_country	int(5)	Ид страны (вторичный ключ)

Поля таблицы «groups» приведены в таблице 3.3.4:

Таблица 3.3.4 – Описание полей таблицы «groups»

Наименование	Тип данных	Назначение
id	mediumint(8)	Идентификатор группы пользователей (первичный ключ)
name	varchar(20)	Наименование
description	varchar(100)	Описание

Поля таблицы «countries» приведены в таблице 3.3.5:

Таблица 3.3.5 – Описание полей таблицы «countries»

Наименование	Тип данных	Назначение
id_country	int(4)	Идентификатор страны (первичный ключ)
name_country	varchar(255)	Наименование

Поля таблицы «times» приведены в таблице 3.3.6:

Таблица 3.3.6 – Описание полей таблицы «times»

Наименование	Тип данных	Назначение
id_time	int(5)	Идентификатор времени (первичный ключ)
time	time(4)	Время

Поля «relationship_between_list_and_place» приведены в таблице 3.3.7:

Таблица 3.3.7 – Описание полей «relationship_between_list_and_place»

Наименование	Тип данных	Назначение
id_list	int(11)	Идентификатор маршрута (вторичный ключ)

id_place	int(11)	Идентификатор достопримечательности (вторичный ключ)
----------	---------	--

Продолжение таблицы 3.3.7

Наименование	Тип данных	Назначение
num_day	int(5)	Номер дня путешествия

Поля таблицы «categories_of_places» приведены в таблице 3.3.8:

Таблица 3.3.8 – Описание полей таблицы «categories_of_places»

Наименование	Тип данных	Назначение
id_category	int(5)	Идентификатор категории достопримечательности (первичный ключ)
name_category	varchar(255)	Наименование

Поля таблицы «saved_routes» приведены в таблице 3.3.9:

Таблица 3.3.9 – Описание полей таблицы «saved_routes»

Наименование	Тип данных	Назначение
id_list	int(11)	Идентификатор маршрута (первичный ключ)
id_user	int(11)	Идентификатор пользователя (вторичный ключ)
name_route	varchar(255)	Наименование маршрута
start_time	time	Время начала дня
location_user	varchar(255)	Местоположение пользователя на момент сохранения маршрута
last_update_date	datetime	Дата последнего изменения маршрута
end_time	time	Время окончания дня
id_transport	int(5)	Идентификатор выбранного способа передвижения между

		достопримечательностями (вторичный ключ)
--	--	---

Продолжение таблицы 3.3.9

Наименование	Тип данных	Назначение
id_first_day	int(6)	Идентификатор дня недели, с которого начинается путешествие
real_location	int(3)	Статус включения отслеживания реального времени
real_location	int(11)	Статус включения отслеживания реального местоположения
id_city	int(11)	Идентификатор города
hesh	varchar(255)	Строка, по которой можно просмотреть уже сохраненный маршрут

Поля таблицы «cities» приведены в таблице 3.3.10:

Таблица 3.3.10 – Описание полей таблицы «cities»

Наименование	Тип данных	Назначение
id_city	int(10)	Идентификатор города (первичный ключ)
name_city	varchar(255)	Наименование
id_country	int(5)	Идентификатор страны, в которой расположен город (вторичный ключ)

Поля «relationship_between_user_and_category» приведены в таблице 3.3.11:

Таблица 3.3.11– Описание полей «relationship_between_user_and_category»

Наименование	Тип данных	Назначение
id_category	int(5)	Идентификатор категории достопримечательности

		(вторичный ключ)
id_user	int(11)	Идентификатор пользователя (вторичный ключ)

Поля таблицы «work_schedules» приведены в таблице 3.3.12:

Таблица 3.3.12 – Описание полей таблицы «work_schedules»

Наименование	Тип данных	Назначение
id	int(4)	Идентификатор графика работы (первичный ключ)
name	varchar(100)	Наименование

Поля таблицы «types_feedback» приведены в таблице 3.3.13:

Таблица 3.3.13 – Описание полей таблицы «types_feedback»

Наименование	Тип данных	Назначение
id_feedback	int(3)	Идентификатор типа сообщения обратной связи (первичный ключ)
name_feedback	varchar(100)	Наименование

Поля таблицы «feedbacks» приведены в таблице 3.3.14:

Таблица 3.3.14 – Описание полей таблицы «feedbacks»

Наименование	Тип данных	Назначение
id_review	int(10)	Идентификатор сообщения обратной связи (первичный ключ)
id_user	int(11)	Идентификатор пользователя (вторичный ключ)
id_feedback	int(3)	Идентификатор типа сообщения (вторичный ключ)

text_review	text	Текст сообщения
creation_date	date	Дата создания
last_updated_date	date	Дата последнего обновления

Продолжение таблицы 3.3.14

Наименование	Тип данных	Назначение
id_status	int(2)	Идентификатор состояния проверки (вторичный ключ)
comment	varchar(4000)	Комментарий проверяющего

Поля таблицы «statuses_request» приведены в таблице 3.3.15:

Таблица 3.3.15 – Описание полей таблицы «statuses_request»

Наименование	Тип данных	Назначение
id_status	int(2)	Идентификатор статуса проверки заявки (первичный ключ)
name_status	varchar(255)	Наименование

Поля таблицы «statuses_feedbacks» приведены в таблице 3.3.16:

Таблица 3.3.16 – Описание полей таблицы «statuses_feedbacks»

Наименование	Тип данных	Назначение
id_status	int(2)	Идентификатор статуса проверки сообщения обратной связи (первичный ключ)
name_status	varchar(255)	Наименование

Поля таблицы «days_of_week» приведены в таблице 3.3.17:

Таблица 3.3.17 – Описание полей таблицы «days_of_week»

Наименование	Тип данных	Назначение
--------------	------------	------------

id	int(5)	Идентификатор дня недели (первичный ключ)
name	varchar(100)	Наименование

Поля таблицы «descriptions_days» приведены в таблице 3.3.18:

Таблица 3.3.18 – Описание полей таблицы «descriptions_days»

Наименование	Тип данных	Назначение
id	int(5)	Идентификатор описания дня (первичный ключ)
name	varchar(100)	Наименование

Поля таблицы «opening_hours_places» приведены в таблице 3.3.19:

Таблица 3.3.19 – Описание полей таблицы «opening_hours_places»

Наименование	Тип данных	Назначение
id	int(10)	Идентификатор дня из графика достопримечательности (первичный ключ)
id_place	int(8)	Идентификатор достопримечательности (вторичный ключ)
id_day	int(5)	Идентификатор дня (вторичный ключ)
id_opening_time	int(5)	Идентификатор времени открытия в этот день (вторичный ключ)
id_closing_time	int(5)	Идентификатор времени закрытия в этот день (вторичный ключ)
description_day	int(5)	Описание дня: рабочий или выходной (вторичный ключ)

Поля «relationship_between_route_and_category» приведены в таблице 3.3.20:

Таблица 3.3.20– Описание полей «relationship_between_route_and_category»

Наименование	Тип данных	Назначение
id_route	int(11)	Идентификатор маршрута (вторичный ключ)
id_category	int(5)	Идентификатор категории (вторичный ключ)

4. РЕАЛИЗАЦИЯ

Ниже представлены страницы веб-приложения.

1. Главная страница.

На рисунках 4.1 – 4.4 представлена главная страница приложения.

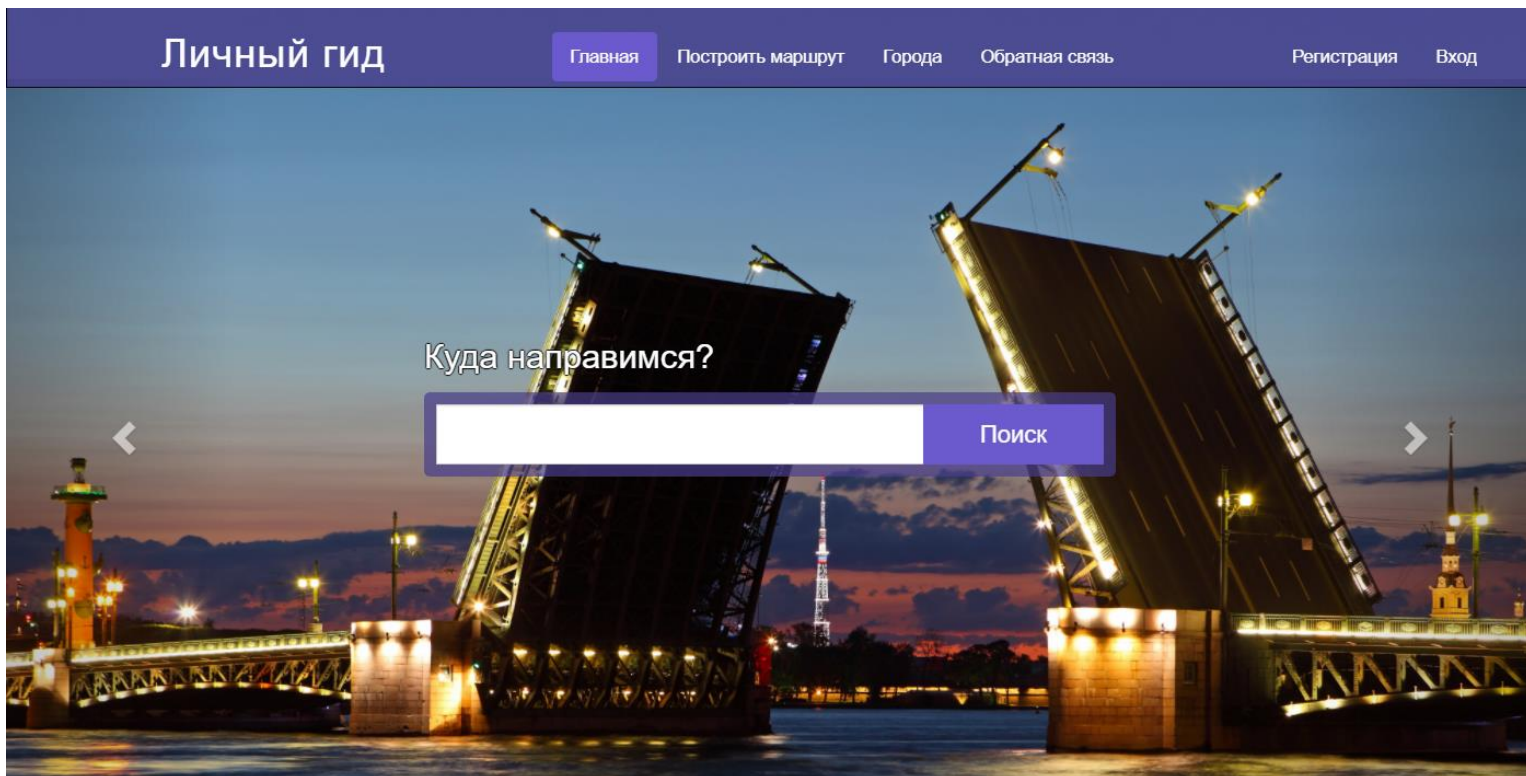


Рисунок 4.1 – Поиск города для путешествия со слайдером (первый слайд)

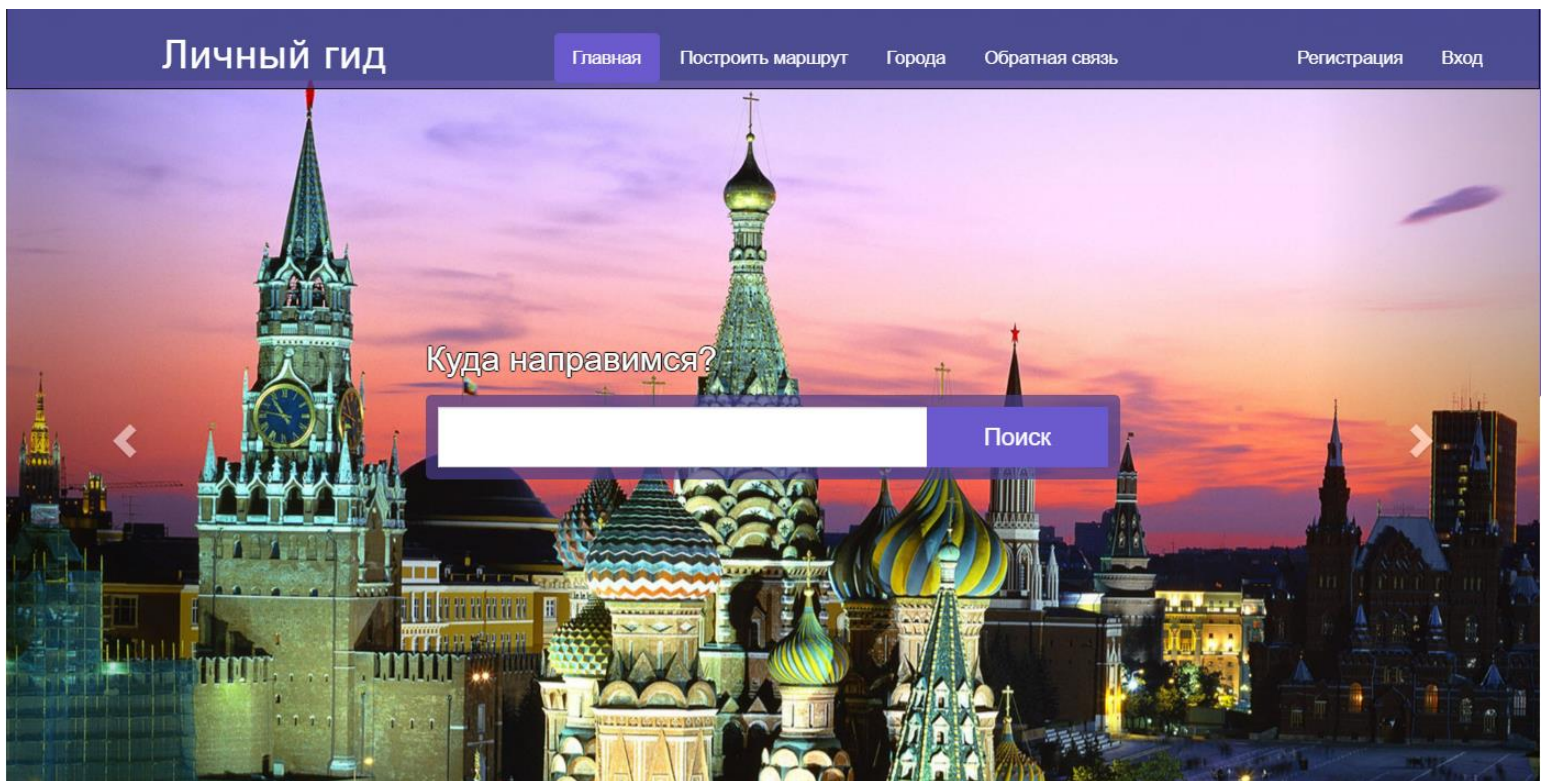



Рисунок 4.2 – Поиск города для путешествия со слайдером (второй слайд)

Культурный туризм



Культурный туризм – это форма туризма, цель которого состоит в знакомстве с культурой и культурной средой места посещения, включая ландшафт, знакомство с традициями жителей и их образом жизни, художественной культурой и искусством, различными формами проведения досуга местных жителей. Культурный туризм охватывает собой посещение исторических, культурных или географических достопримечательностей.

В России культурный туризм уже давно стал самым популярным и массовым видом туризма. С каждым годом растет количество людей, желающих познакомиться с историческими и культурными достопримечательностями страны. Многие хотят своими глазами увидеть известные исторические и природные памятники, которыми так богата российская земля.




Рисунок 4.3 – Описание предметной области на главной странице

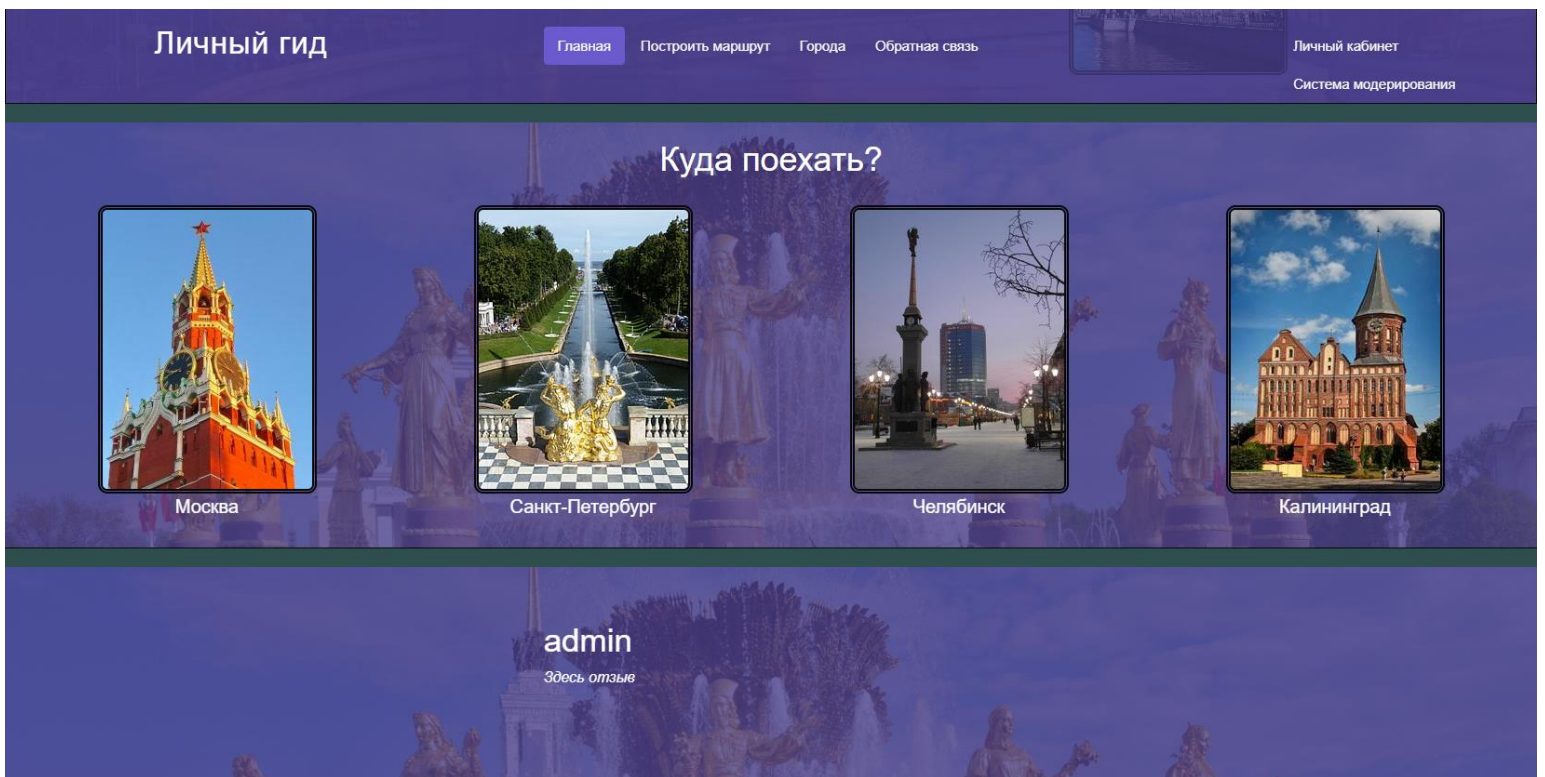


Рисунок 4.4 – Предлагаемые города для неопределившегося пользователя и отзывы пользователей

2. Построение маршрута.

На рисунках 4.5 – 4.6 представлена страница построения маршрута. Код страницы построения маршрута приведен в листинге А.1 приложения А.

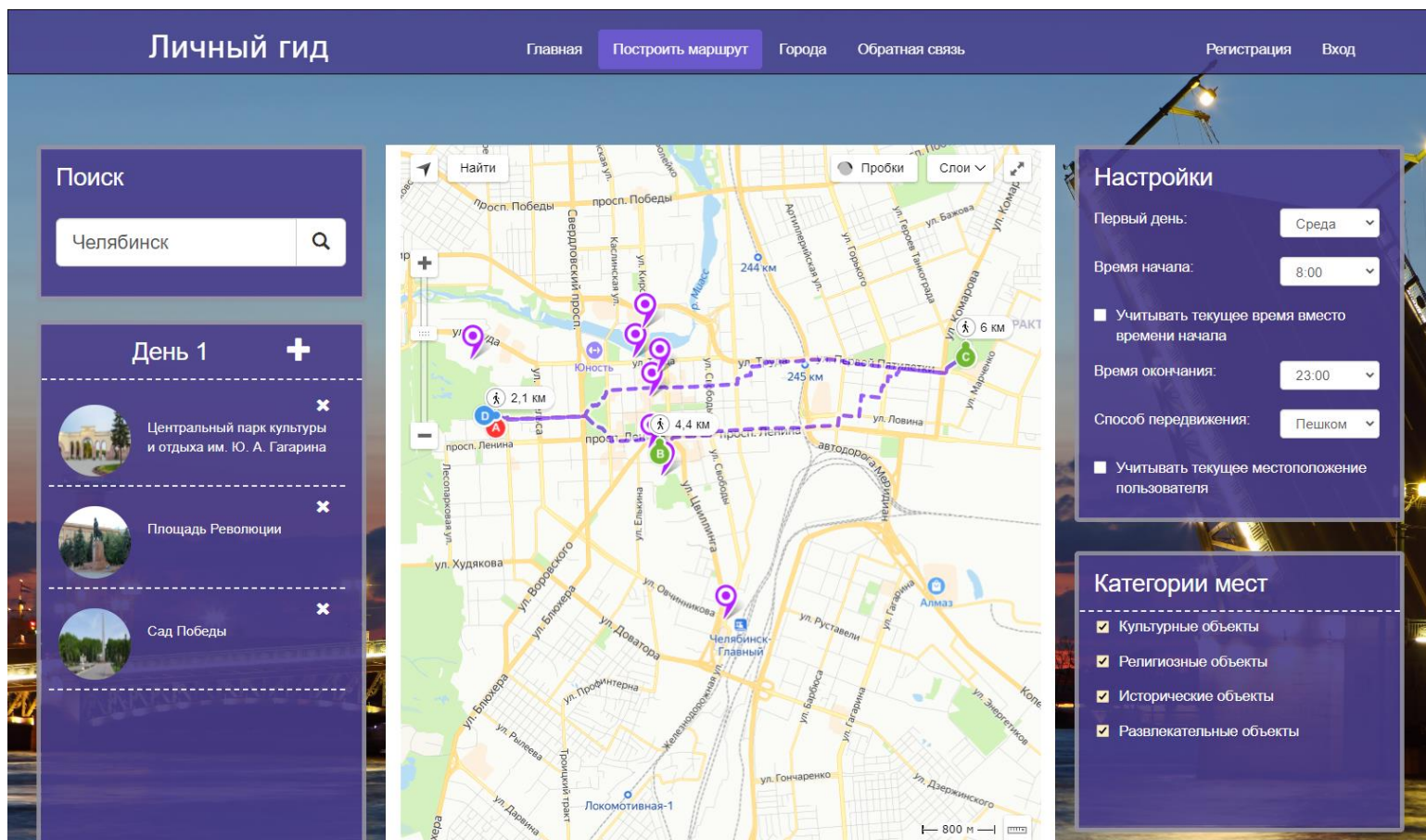


Рисунок 4.5 – Страница построения маршрута

На рисунке 4.5 отображена страница построения маршрута, в ней имеется поиск городов, настройки построения маршрута: выбор первого дня путешествия; время начала и окончания путешествия каждого дня; выбор способа передвижения; учета текущего местоположения и времени. Снизу находится фильтр достопримечательностей. С другой стороны, отображается список выбранных достопримечательностей, а также кнопка для добавления новых дней.

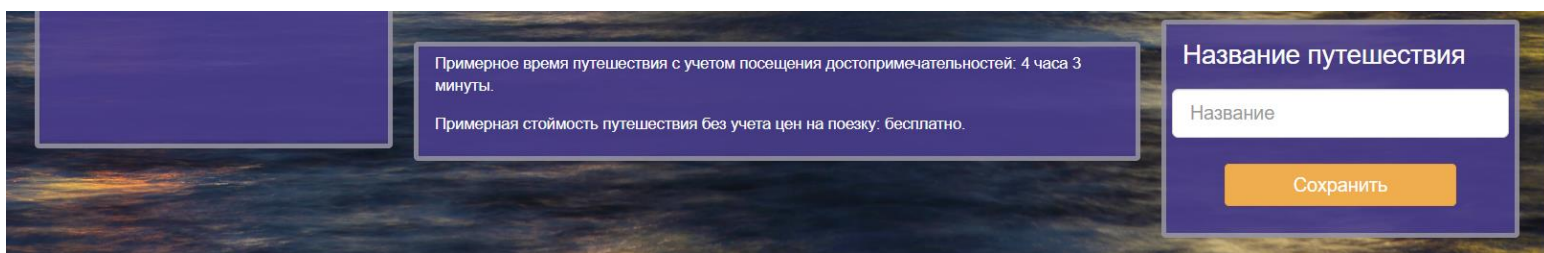


Рисунок 4.6 – Вычисляемое время путешествия и цена путешествия.

На рисунке 4.6 отображается расчет времени путешествия и его стоимости. Справа имеется форма для ввода названия путешествия и кнопка его сохранения.

3. Список городов.

На рисунке 4.7 отображается список городов.

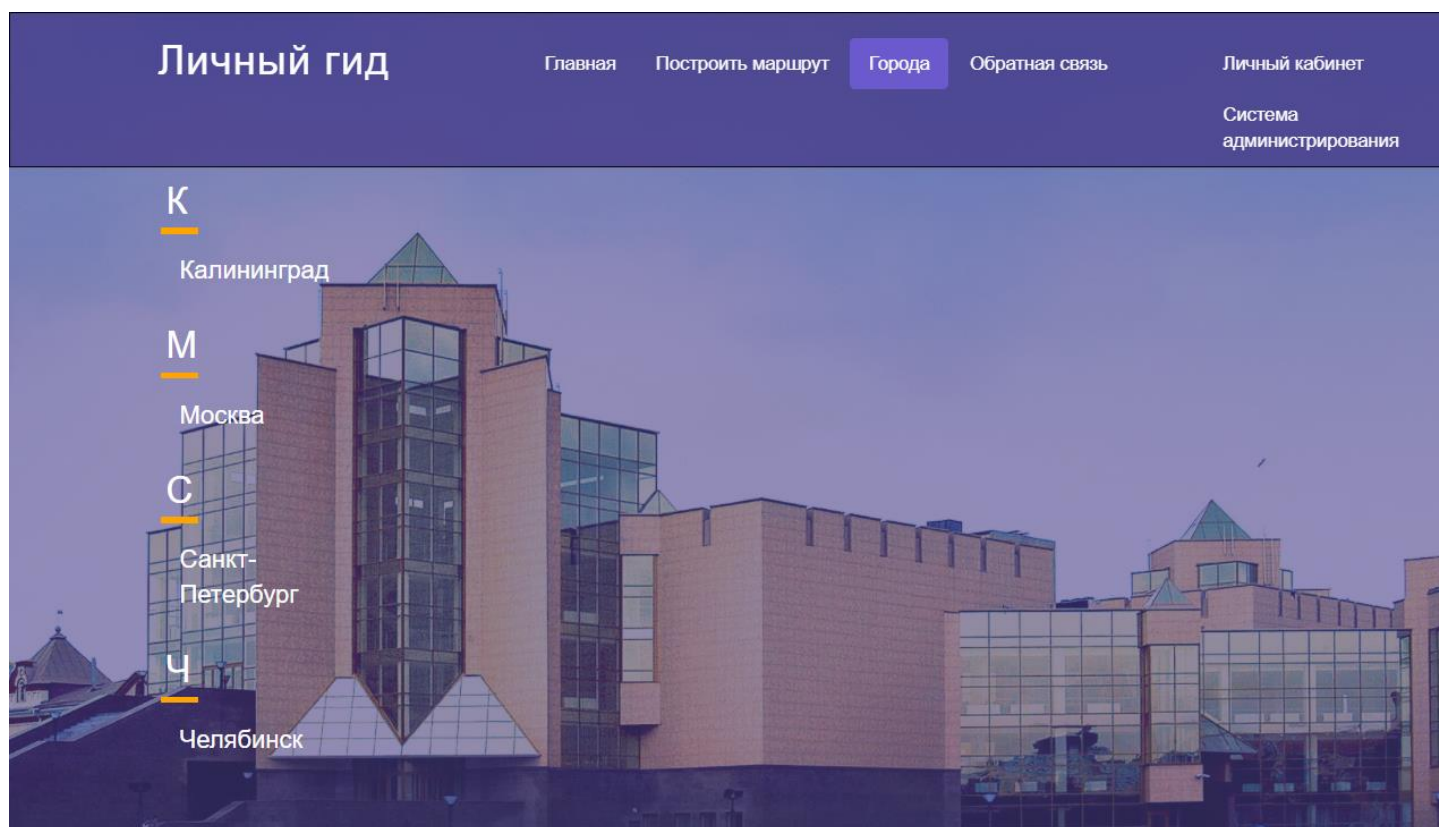


Рисунок 4.7 – Список городов

В списке отображаются города, в которых отмечена хотя бы одна достопримечательность.

4. Форма обратной связи.

На рисунке 4.8 отображена форма обратной связи.

The image shows a web interface for a 'Личный гид' (Personal Guide). The top navigation bar includes links for 'Главная', 'Построить маршрут', 'Города', 'Обратная связь', 'Личный кабинет', and 'Система администрирования'. The 'Обратная связь' link is highlighted. Below the navigation bar is a large image of a bridge at night. Overlaid on this image is a feedback form titled 'Обратная связь'. The form contains a dropdown menu with the following options: 'Отзыв', 'Отзыв', 'Предложение по улучшению', and 'Обращение в техподдержку'. The second 'Отзыв' option is currently selected. Below the dropdown is a large white text input field. At the bottom right of the form is an orange button labeled 'Отправить'.

Рисунок 4.8 – Форма обратной связи

Формой обратной связи может воспользоваться только авторизованный пользователь, в этой форме он может оставить свой отзыв о приложении, написать свои предложения по улучшению или обратиться в техподдержку.

5. Личный кабинет пользователя.

В личном кабинете пользователь может изменить пароль, задать свои предпочтения при планировании путешествий, они будут автоматически задаваться в построителе маршрутов. На рисунках 4.9 – 4.10 отображена страница личного кабинета пользователя.

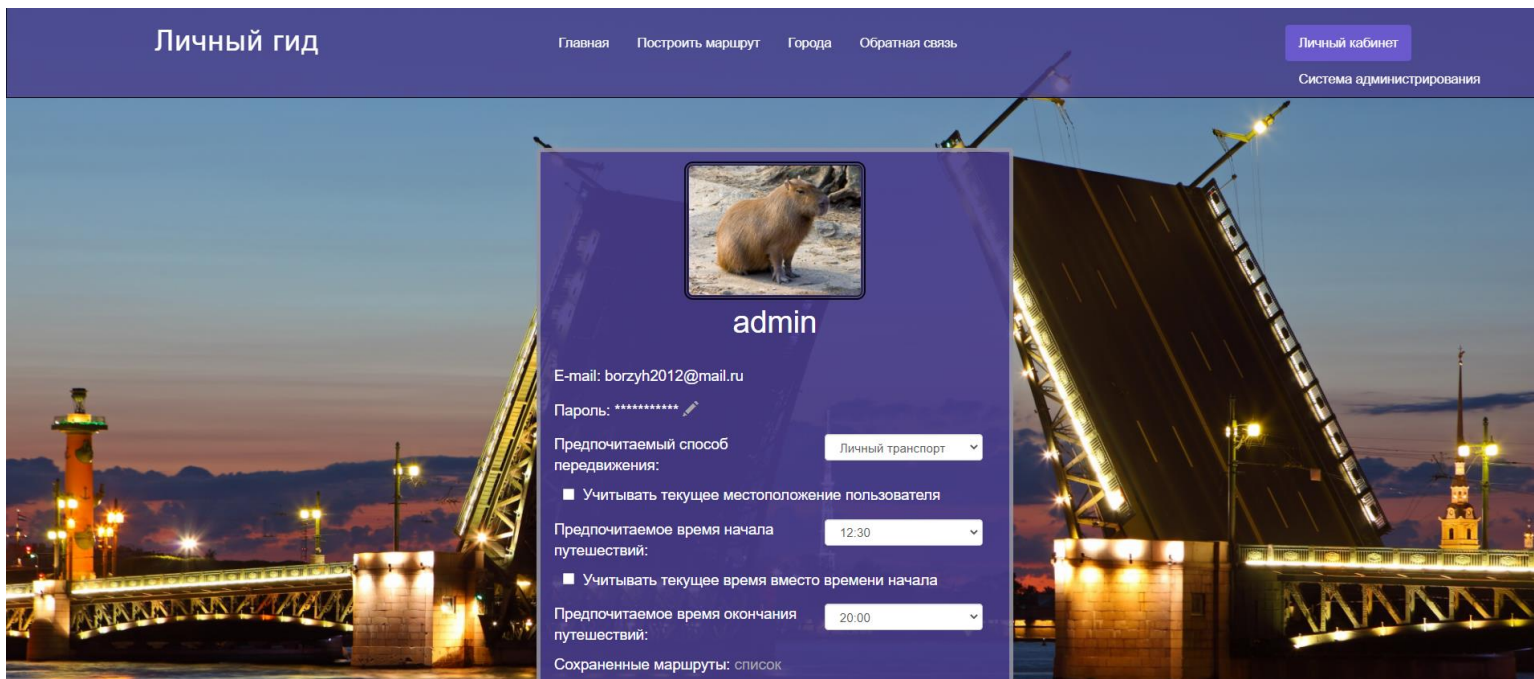


Рисунок 4.9 – Верхняя часть страницы личного кабинета

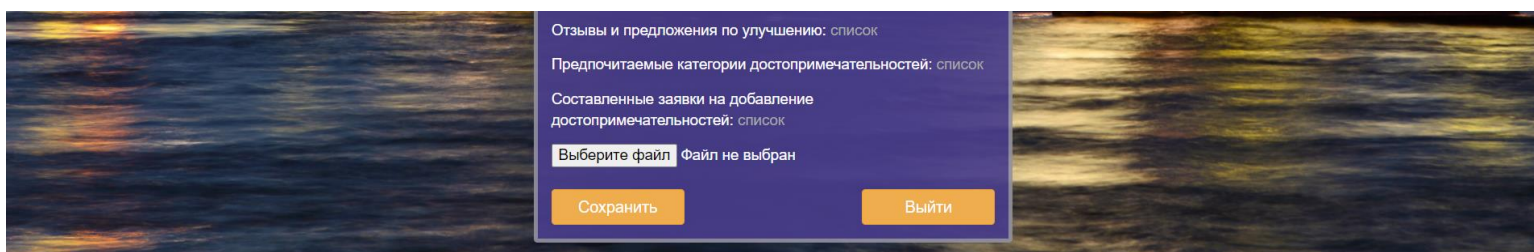


Рисунок 4.10 – Нижняя часть страницы личного кабинета

Пользователь имеет возможность просмотреть список сохраненных маршрутов, составленные им сообщения обратной связи и заявки на добавление достопримечательностей, а также изменить изображение в личном кабинете.

6. Список сохраненных маршрутов.

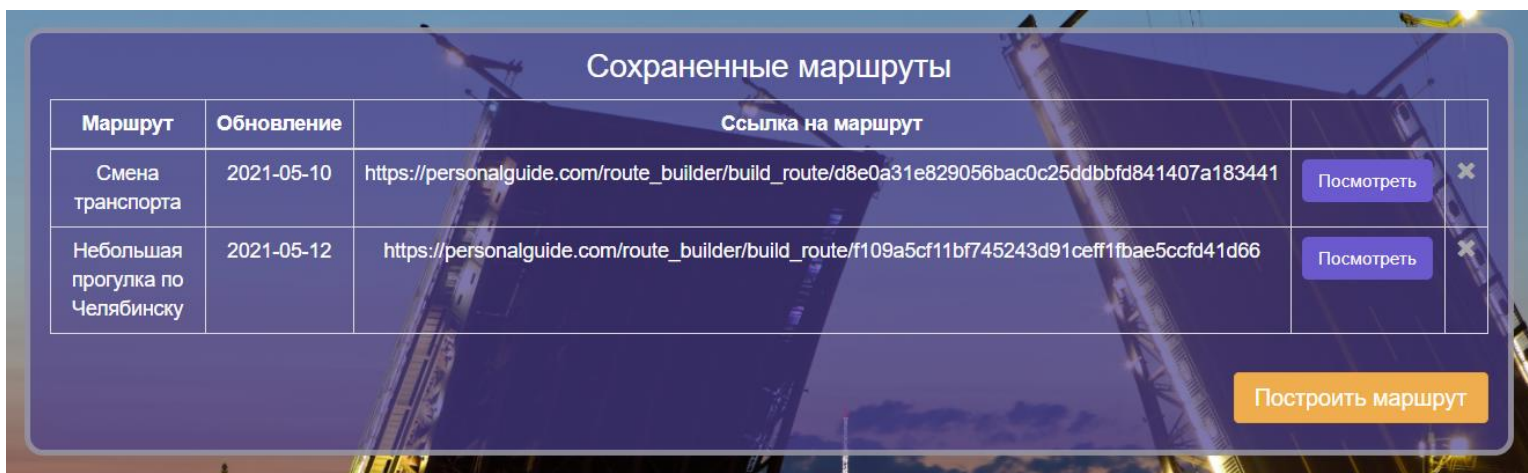


Рисунок 4.11 – Список сохраненных маршрутов пользователем

На рисунке 4.11 в списке сохраненных маршрутов отображаются следующие поля: название маршрута; дата последнего обновления; ссылка, по которой другие пользователи могут посмотреть твой маршрут; кнопки для просмотра и удаления маршрута.

7. Список написанных пользователем сообщений по обратной связи.



Рисунок 4.12 – Список сообщений по обратной связи

Пользователь может просмотреть краткую информацию о каждом сообщении, изменить или удалить его.

8. Страница предпочитаемых категорий достопримечательностей пользователем.

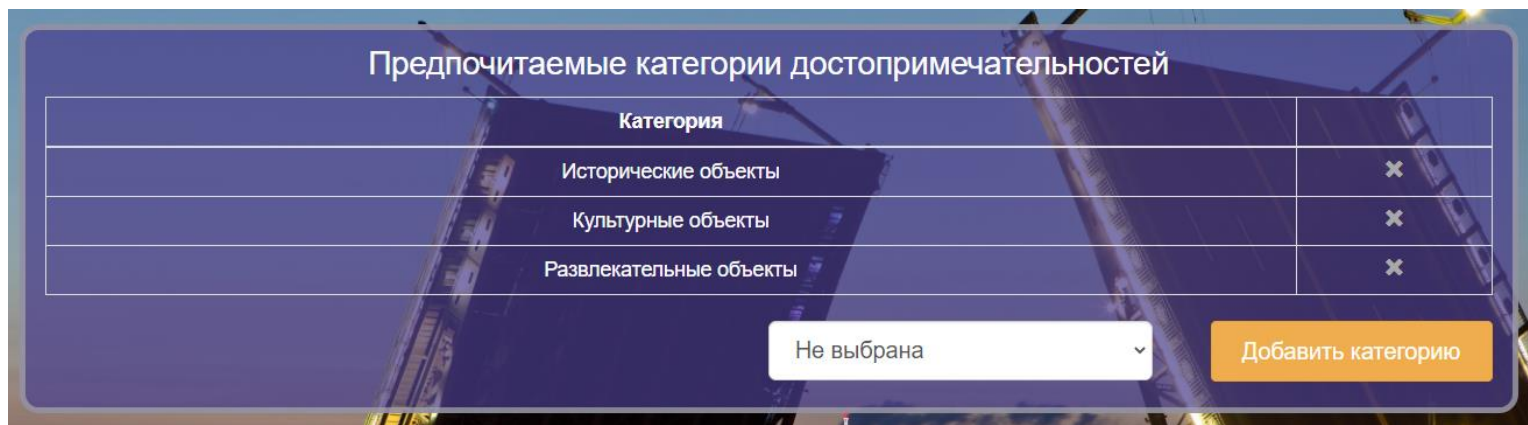


Рисунок 4.13 – Список предпочитаемых категорий пользователем
Пользователь может удалить или добавить категории достопримечательностей из списка имеющихся.

9. Страница заявок пользователя на добавление достопримечательностей.

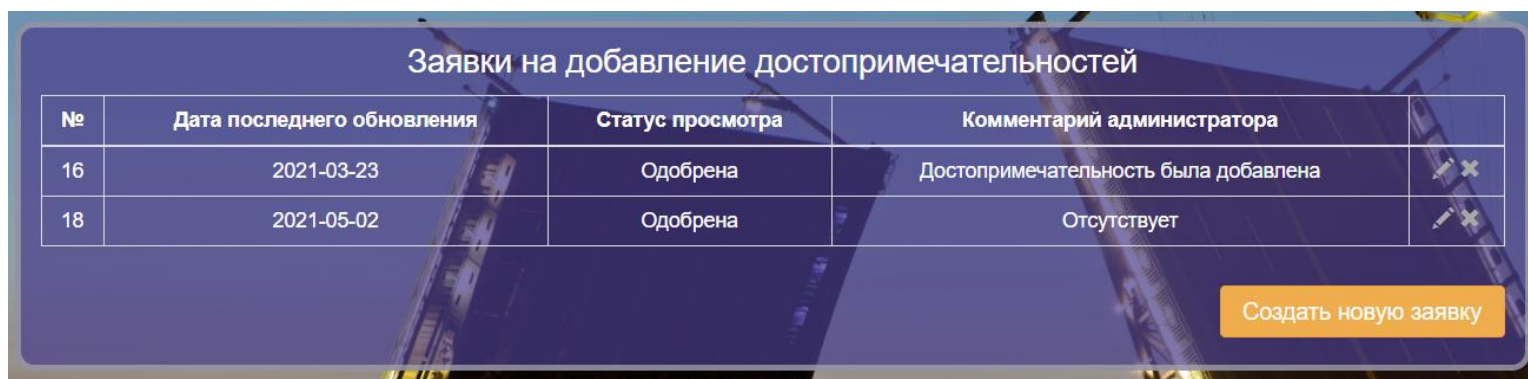
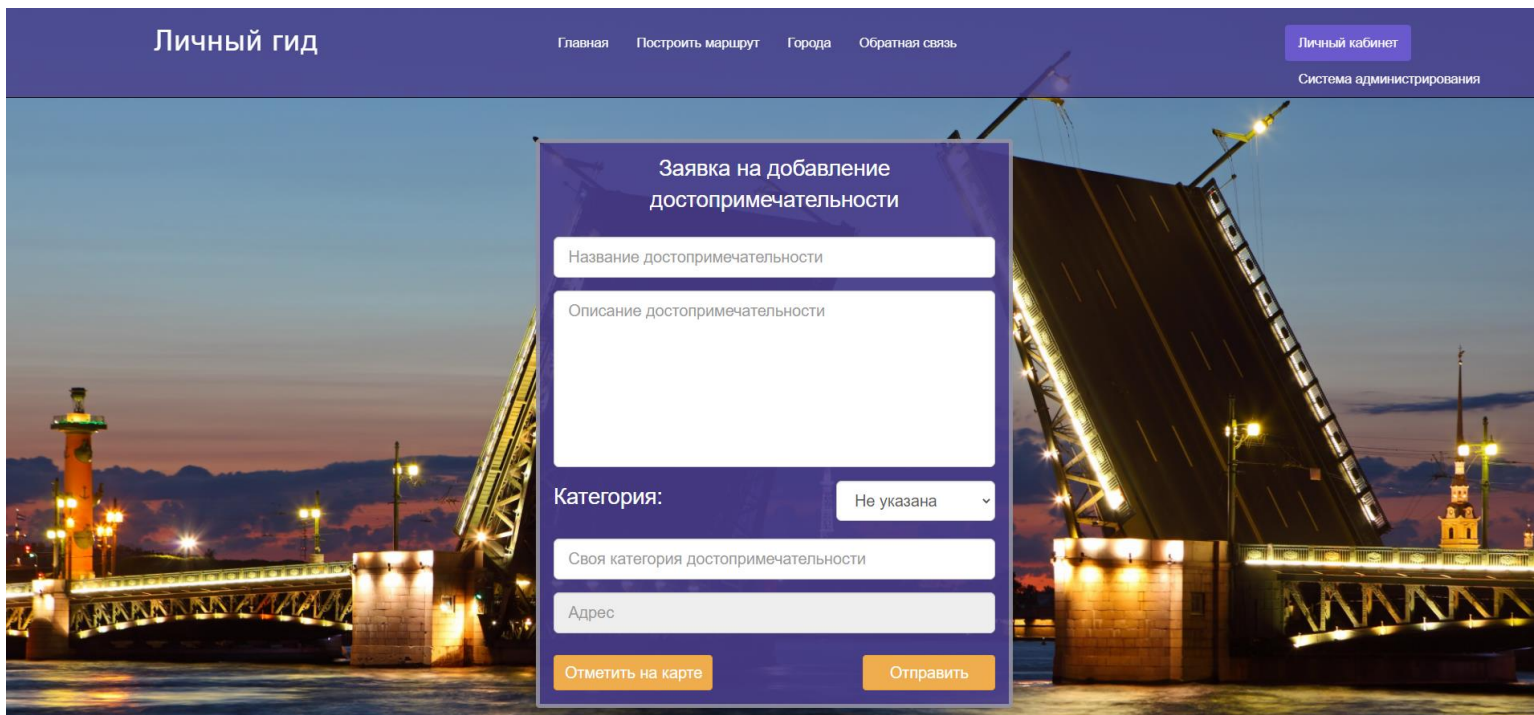


Рисунок 4.14 – Составленные заявки пользователем на добавление достопримечательностей

Пользователь может получить краткую справку по текущему состоянию заявки. Добавить, изменить или удалить заявку.

10. Страница создания или редактирования заявки.



Личный гид

Главная Построить маршрут Города Обратная связь

Личный кабинет
Система администрирования

Заявка на добавление достопримечательности

Название достопримечательности

Описание достопримечательности

Категория: Не указана

Своя категория достопримечательности

Адрес

Отметить на карте Отправить

Рисунок 4.15 – Заявка на добавление достопримечательности

Форма имеет меньший функционал, в сравнении с формой добавления достопримечательности. Здесь пользователь имеет возможность написать свою категорию достопримечательности, если его не устраивают имеющиеся.

11. Страница системы администрирования.

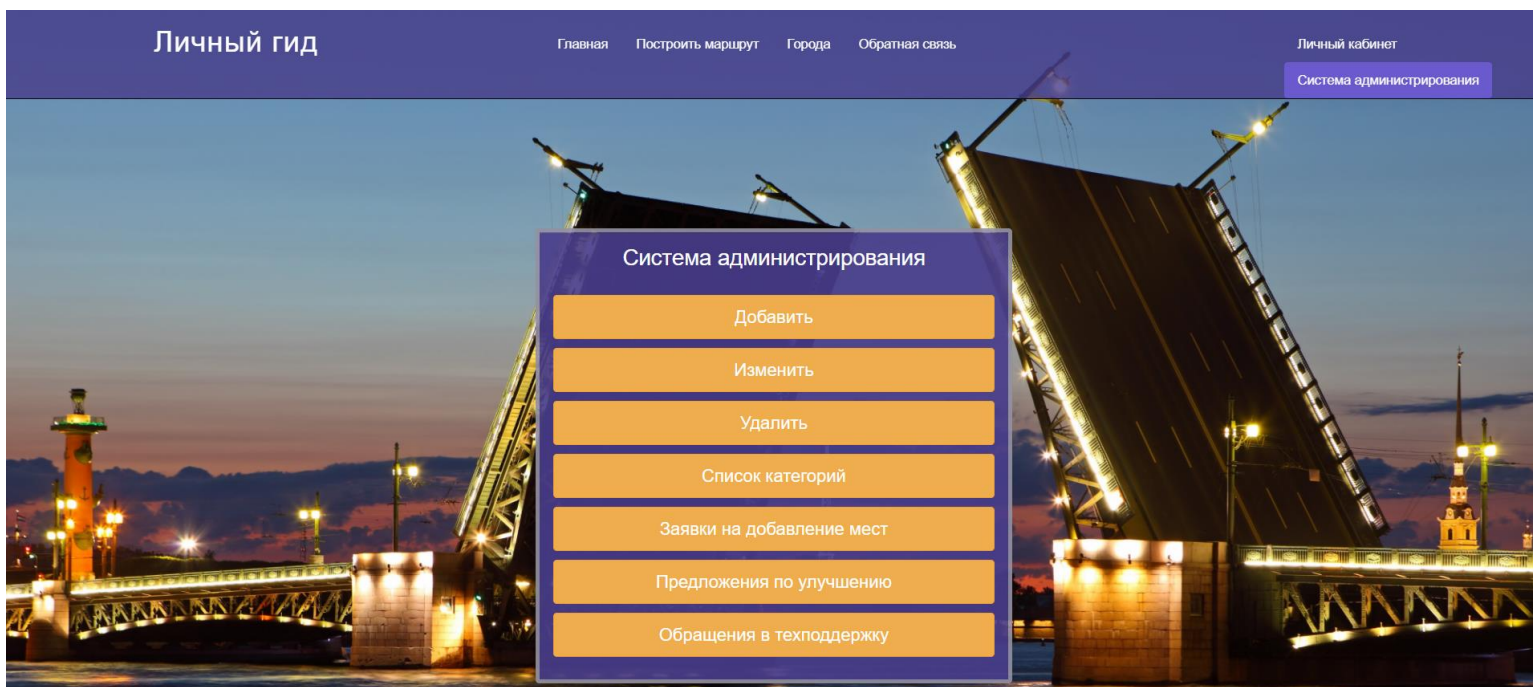
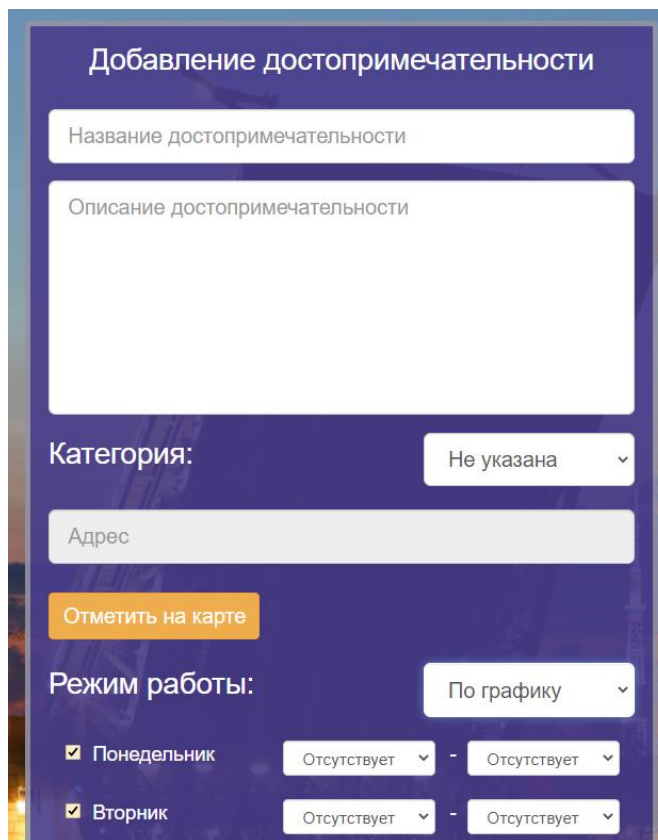


Рисунок 4.16 – Система администрирования

Администратор имеет следующий функционал: возможность добавлять, изменять и удалять достопримечательности; изменять категории достопримечательностей; отвечать на заявки, предложения по улучшению и обращения в техподдержку пользователей.

12. Форма добавления достопримечательности.

На рисунках 4.17 – 4.18 отображена форма добавления достопримечательности.



Добавление достопримечательности

Название достопримечательности

Описание достопримечательности

Категория: Не указана

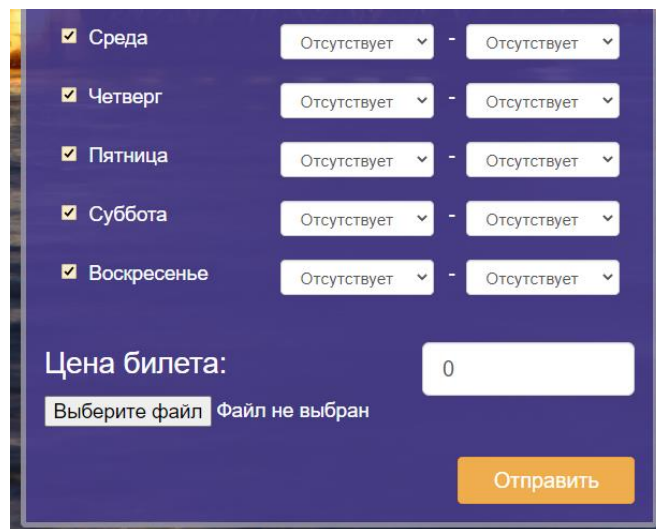
Адрес

Отметить на карте

Режим работы: По графику

<input checked="" type="checkbox"/> Понедельник	Отсутствует	-	Отсутствует
<input checked="" type="checkbox"/> Вторник	Отсутствует	-	Отсутствует

Рисунок 4.17 – Верхняя часть формы добавления достопримечательности



<input checked="" type="checkbox"/> Среда	Отсутствует	-	Отсутствует
<input checked="" type="checkbox"/> Четверг	Отсутствует	-	Отсутствует
<input checked="" type="checkbox"/> Пятница	Отсутствует	-	Отсутствует
<input checked="" type="checkbox"/> Суббота	Отсутствует	-	Отсутствует
<input checked="" type="checkbox"/> Воскресенье	Отсутствует	-	Отсутствует

Цена билета: 0

Выберите файл | Файл не выбран

Отправить

Рисунок 4.18 – Нижняя часть формы добавления достопримечательности

В данной форме указываются следующие поля: название, описание, категория, адрес, режим работы и стоимость посещения достопримечательности. Так же можно задать изображение. При выборе режима работы «По графику» выдвигаются поля для заполнения графика на неделю. Установленными галочками отмечаются рабочие дни.

13. Страница поиска достопримечательности.

Для изменения или удаления достопримечательности требуется выбрать соответствующее место, для этого используется форма поиска, в которой задаются параметры искомого места. Форма поиска отображена на рисунке 4.19.

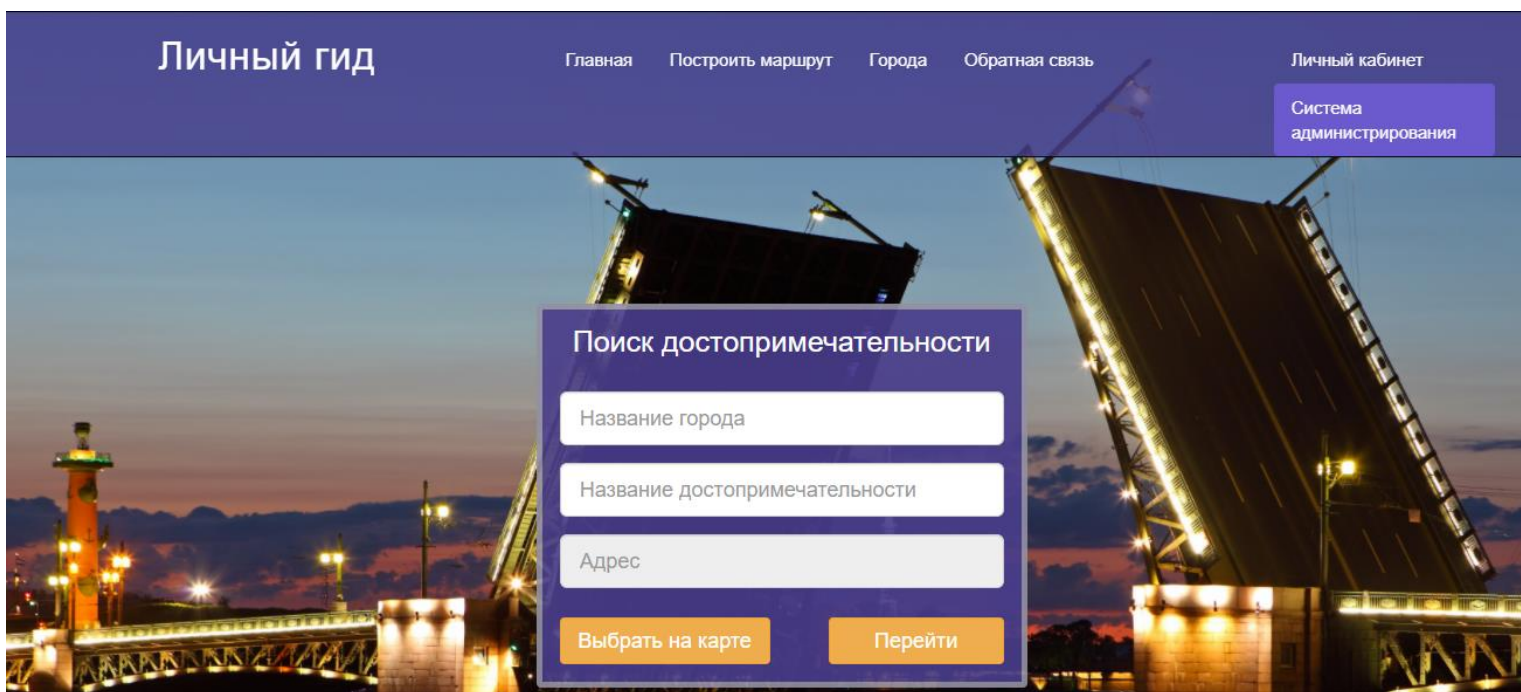


Рисунок 4.19 – Поиск достопримечательности

Администратор может задать название города и достопримечательности, но адрес указать он самостоятельно не может, ему требуется выбрать нужное место на карте и поле будет автоматически заполнено.

14. Список категорий достопримечательностей.

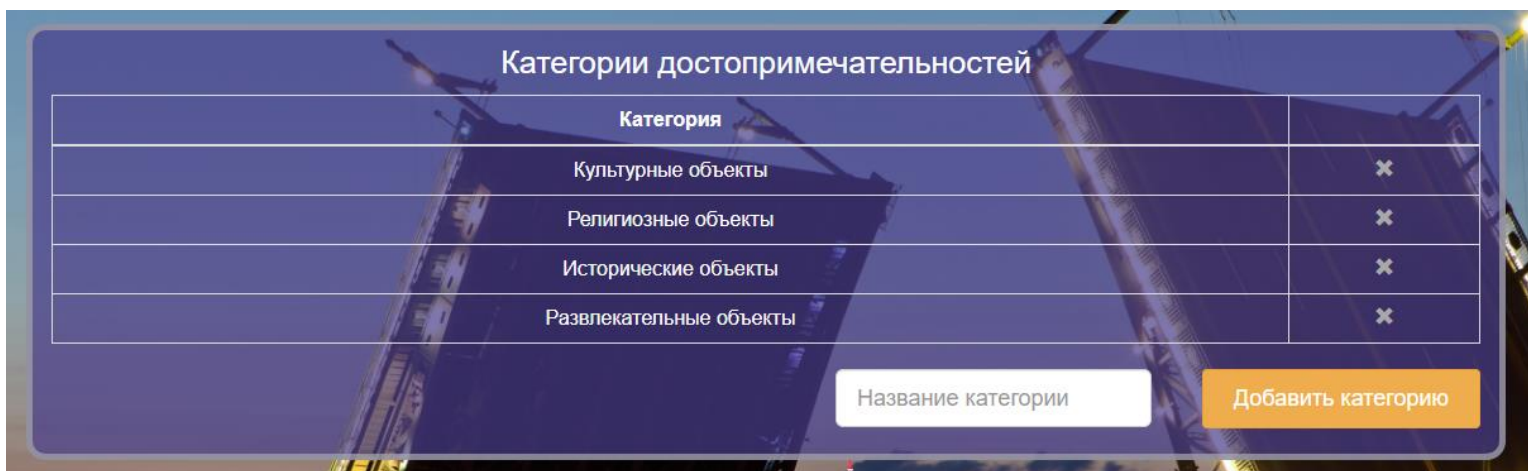


Рисунок 4.20 – Список категорий достопримечательностей

На рисунке 4.20 отображены категории достопримечательностей, администратор может добавить новые, написав название и нажав на кнопку. При удалении категории ей не должны соответствовать достопримечательности.

15. Заявки на добавление достопримечательности.

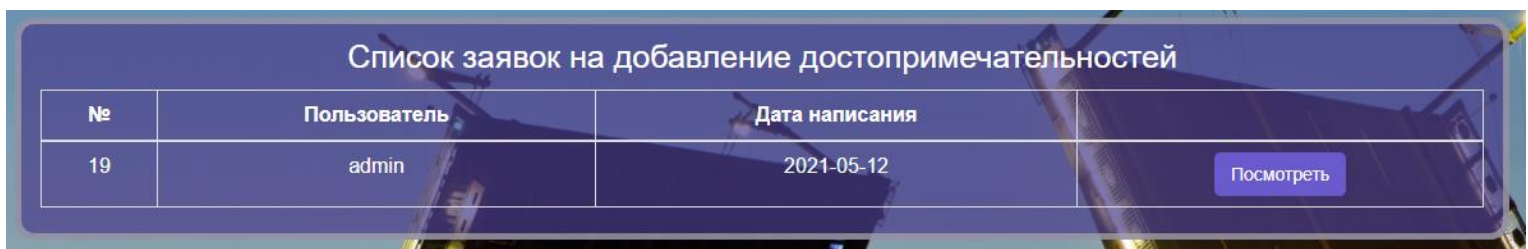


Рисунок 4.21 – Список заявок на добавление достопримечательностей

В списке выводится пользователь, оставивший заявку и дата написания. Здесь же можно перейти к просмотру выбранной заявки.

16. Просмотр заявки на добавление достопримечательности.

Заявка на добавление достопримечательности

№: 19

Пользователь: admin

Название достопримечательности: Театр «Манекен»

Описание достопримечательности: Здание в стиле советского конструктивизма построили в 1937 году специально для кинотеатра им. Пушкина (отсюда и изображение поэта на фасаде). На момент строительства это был единственный в стране кинотеатр с двумя залами. В 2000 году кинотеатр объединили с театром «Манекен». На сегодняшний момент «Манекен» является одним из самых известных театров Челябинска. Основанный в 1963 году как студенческое творческое объединение, за несколько десятилетий театр вышел на профессиональный уровень. В его репертуаре есть спектакли для маленьких и больших зрителей.

Категория достопримечательности: Культурные объекты

Адрес: Россия, Челябинск, улица Пушкина, 64

[Посмотреть на карте](#)

Комментарий для пользователя

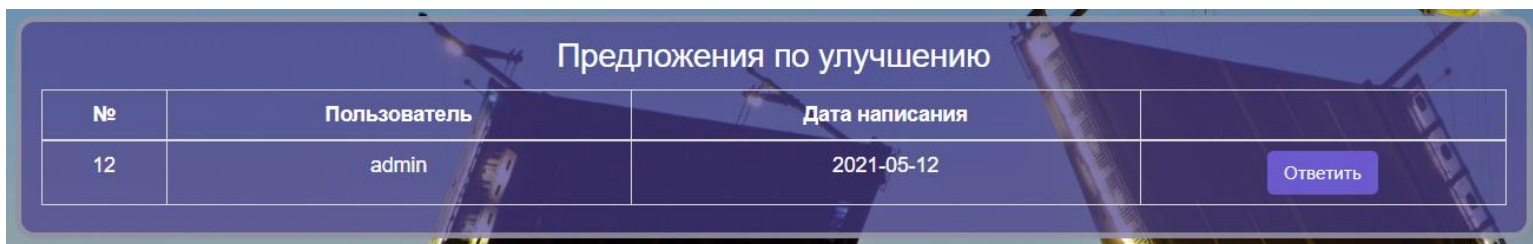
Статус: Нерассмотрена ▾

[Сохранить](#)

Рисунок 4.22 – Заявка на добавление достопримечательности

Администратор может оставить комментарий пользователю и установить статус (отклонена или одобрена). В случае одобрения администратора перенесет в форму добавления достопримечательности.

17.Список предложений по улучшению.

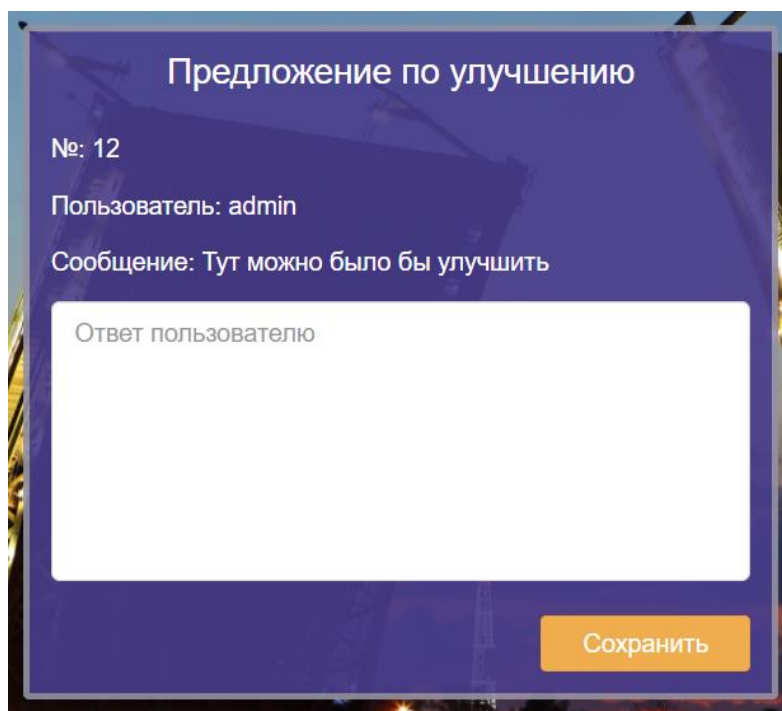


№	Пользователь	Дата написания	
12	admin	2021-05-12	Ответить

Рисунок 4.23 – Список предложений по улучшению

Здесь администратор может посмотреть нерассмотренные предложения по улучшению и выбрать одно для ответа.

18. Ответ на предложение по улучшению.



Предложение по улучшению

№: 12

Пользователь: admin

Сообщение: Тут можно было бы улучшить

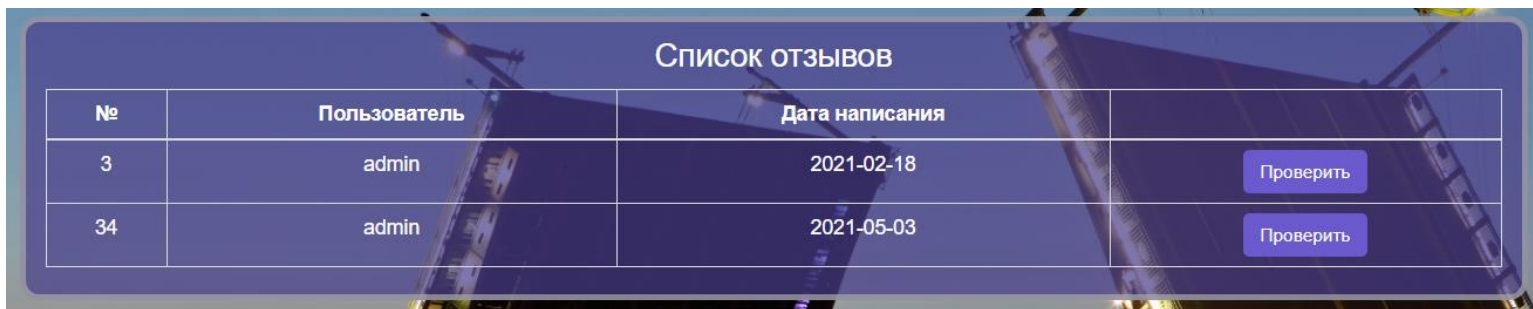
Ответ пользователю

Сохранить

Рисунок 4.24 – Ответ на предложение по улучшению

Обращения в техподдержку идентичны предложения по улучшению, поэтому далее не будут отображены.

19.Список непроверенных отзывов.

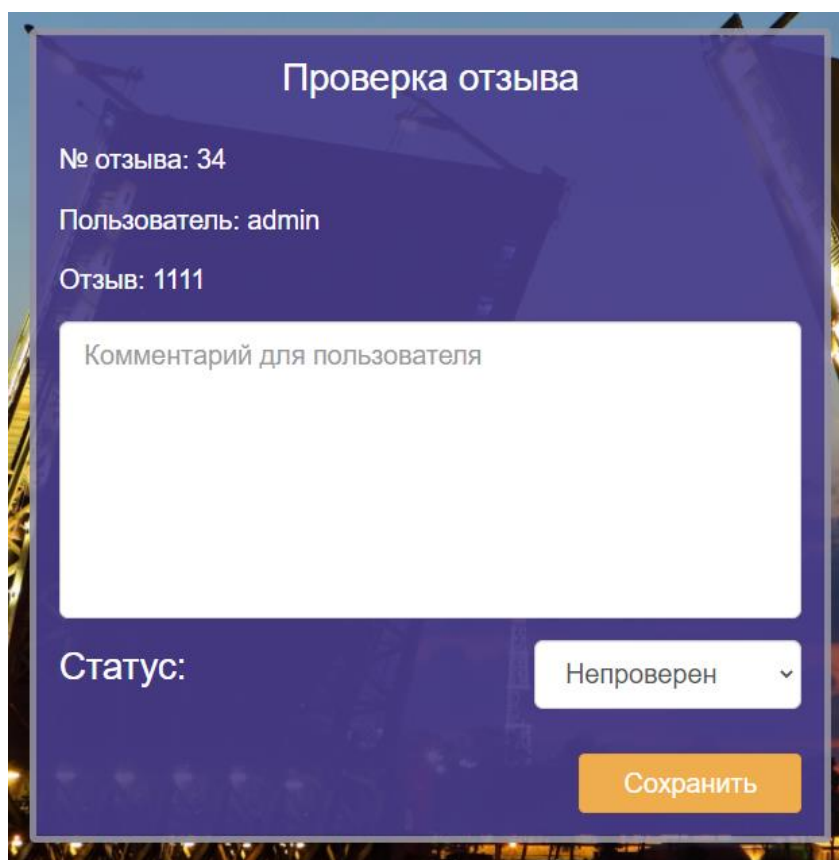


№	Пользователь	Дата написания	
3	admin	2021-02-18	Проверить
34	admin	2021-05-03	Проверить

Рисунок 4.25 – Список непроверенных отзывов

Модератор может выбрать из списка отзыв для проверки.

20. Проверка отзыва.



Проверка отзыва

№ отзыва: 34

Пользователь: admin

Отзыв: 1111

Комментарий для пользователя

Статус: Непроверен

Сохранить

Рисунок 4.26 – Проверка отзыва

Модератор устанавливает статус проверки (отклонен или одобрен), в случае отклонения отзыва требуется указать причину в комментарии для пользователя.

5. ТЕСТИРОВАНИЕ

В ходе тестирования была проверена корректная работа основных наиболее важных элементов системы, а именно: построение маршрута; добавление достопримечательностей; поиск достопримечательностей для их редактирования или удаления; написание заявок пользователем на добавление новых достопримечательностей; страница проверки заявок пользователей; страница редактирования категорий достопримечательностей.

1. Тестирование построения маршрута.

Были отобраны 5 достопримечательностей и заданы настройки без учета реального времени или местоположения.

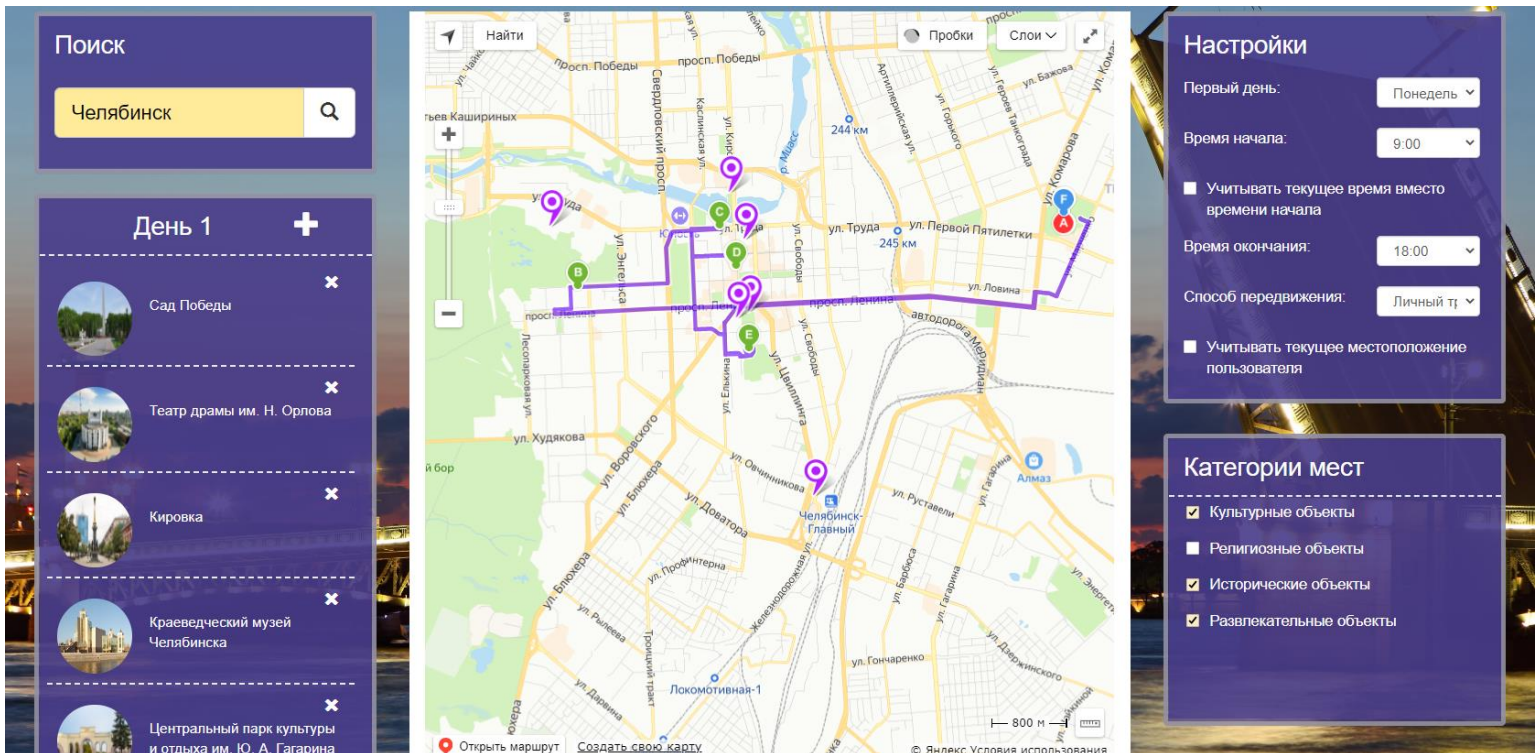


Рисунок 5.1 – Проверка построения маршрута при статических настройках.

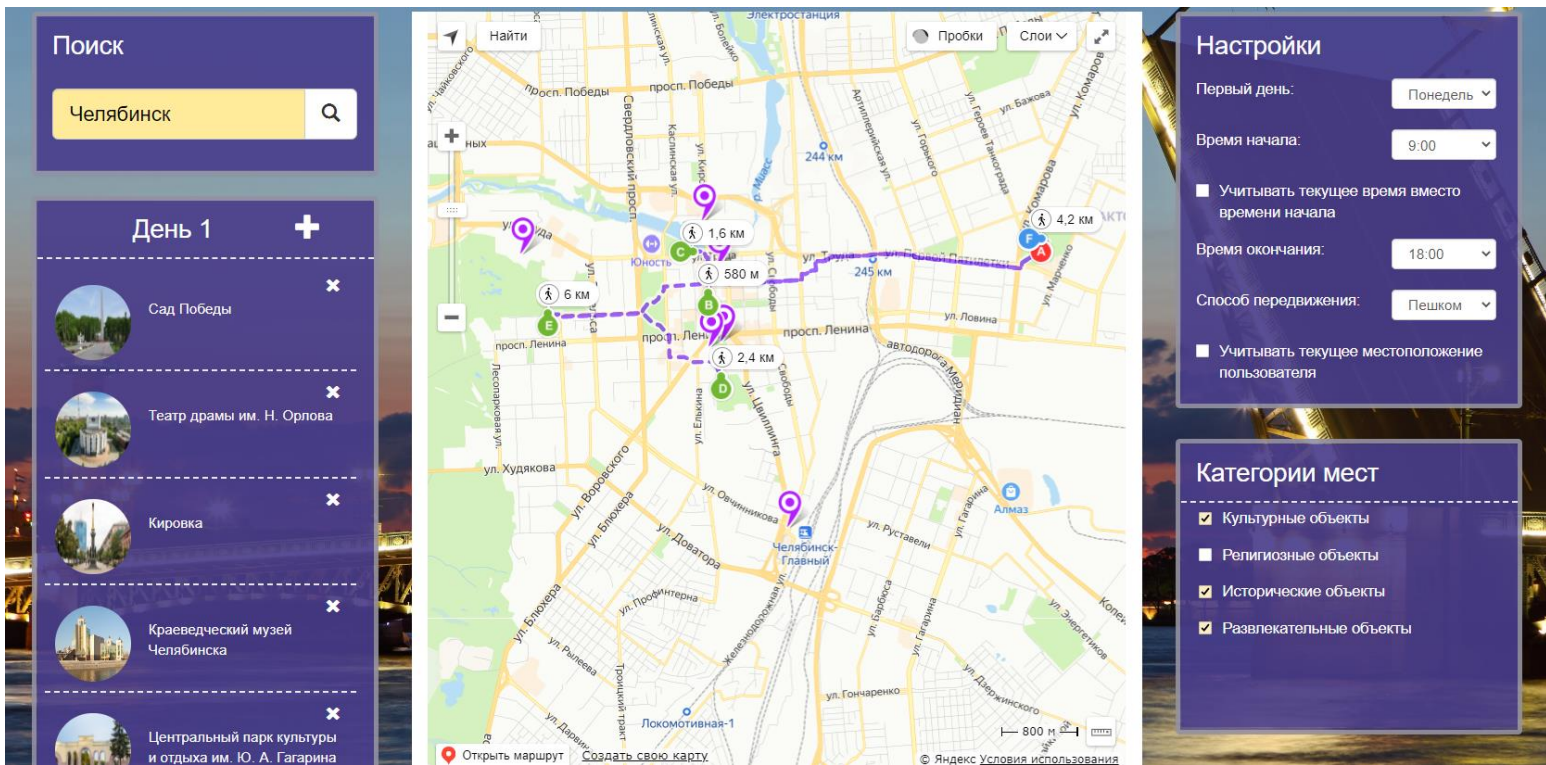


Рисунок 5.2 – Изменение способа передвижения

После смены настройки передвижения был изменен маршрут и последовательность посещения достопримечательностей.

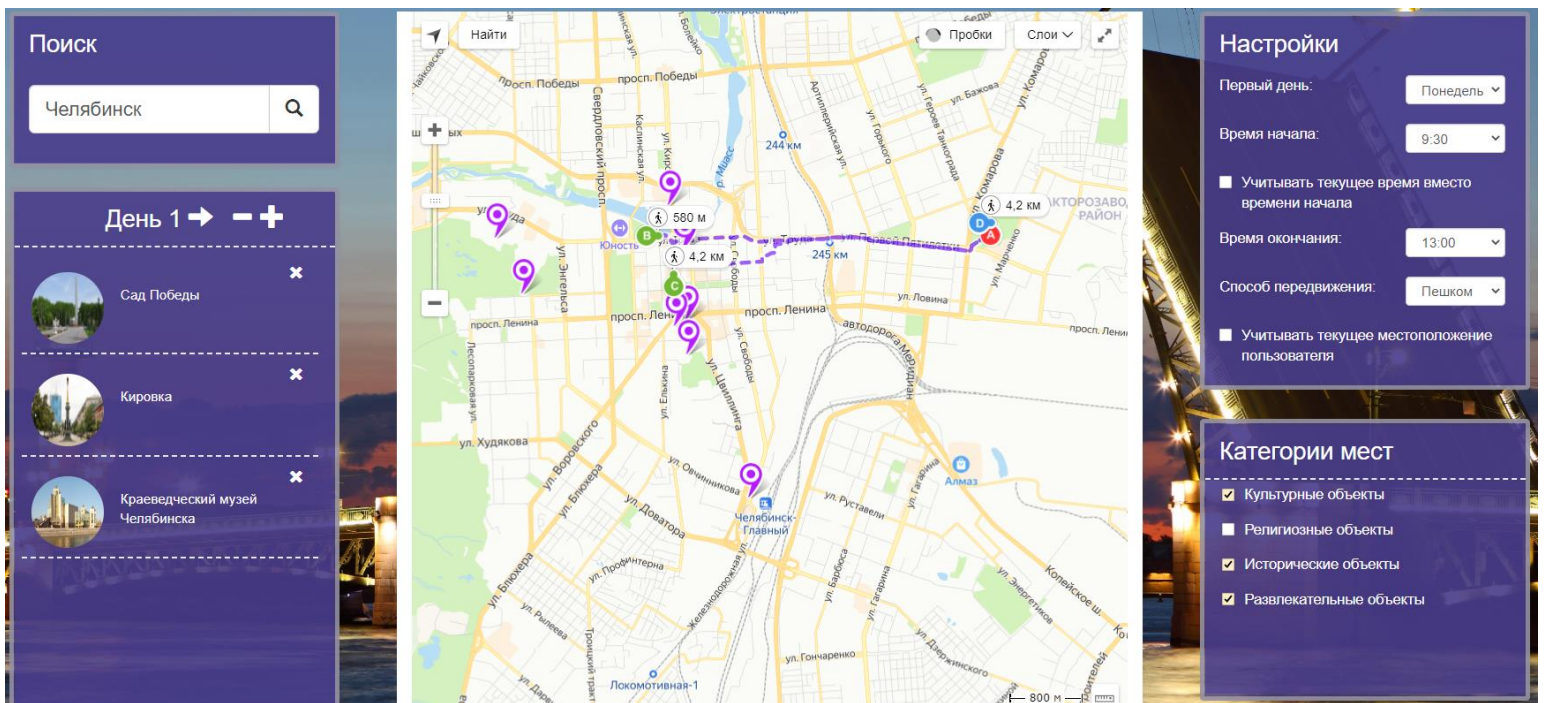


Рисунок 5.3 – Изменение настроек времени (день 1)

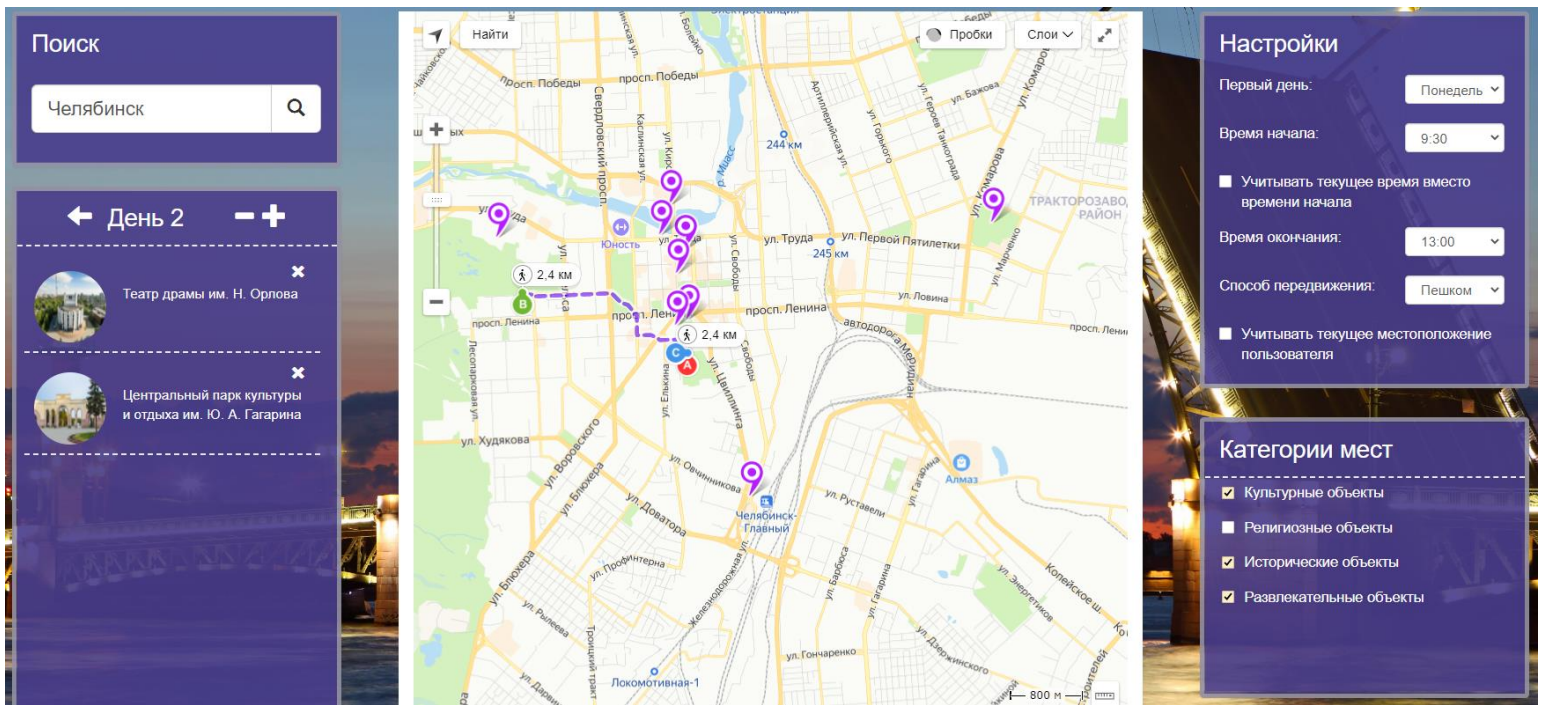


Рисунок 5.4 – Изменение настроек времени (день 2)

После уменьшения продолжительности путешествия часть достопримечательностей была перенесена на следующий день.

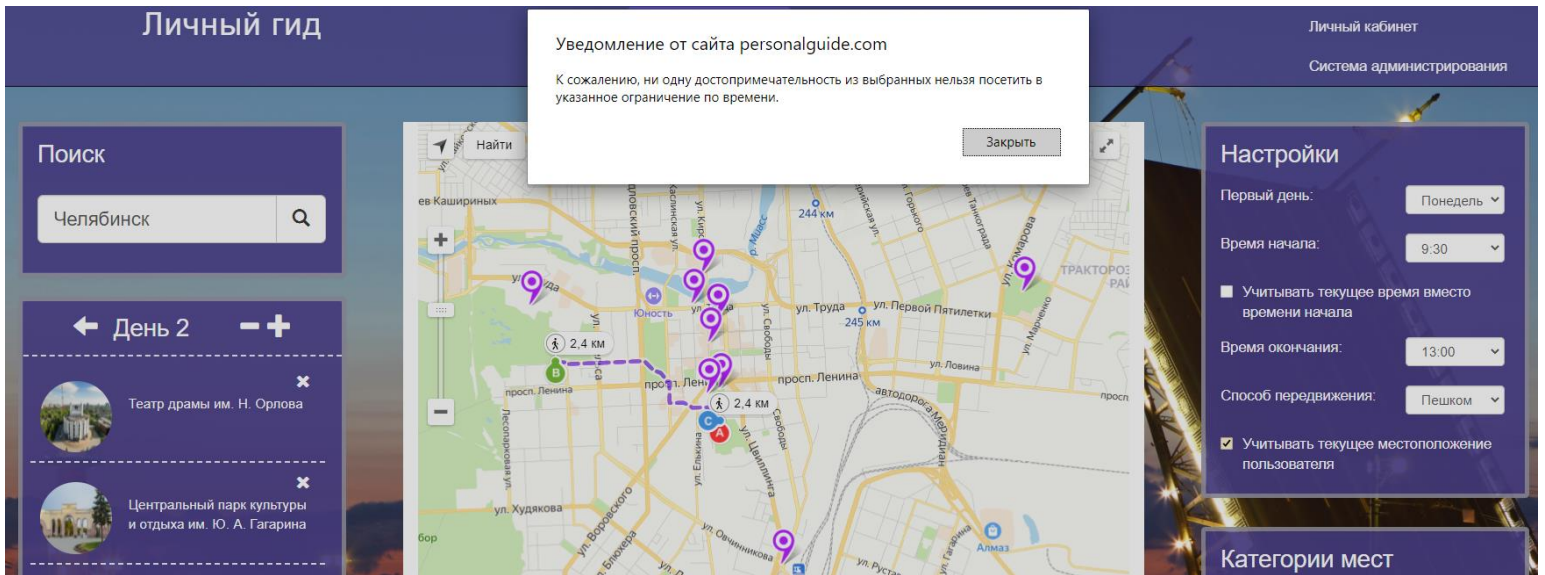


Рисунок 5.5 – Учет текущего местоположения пользователя

Поскольку пользователь находится за городом и время пути пешком до достопримечательностей будет очень долгим, то в указанные ограничения он не сможет посетить достопримечательности.

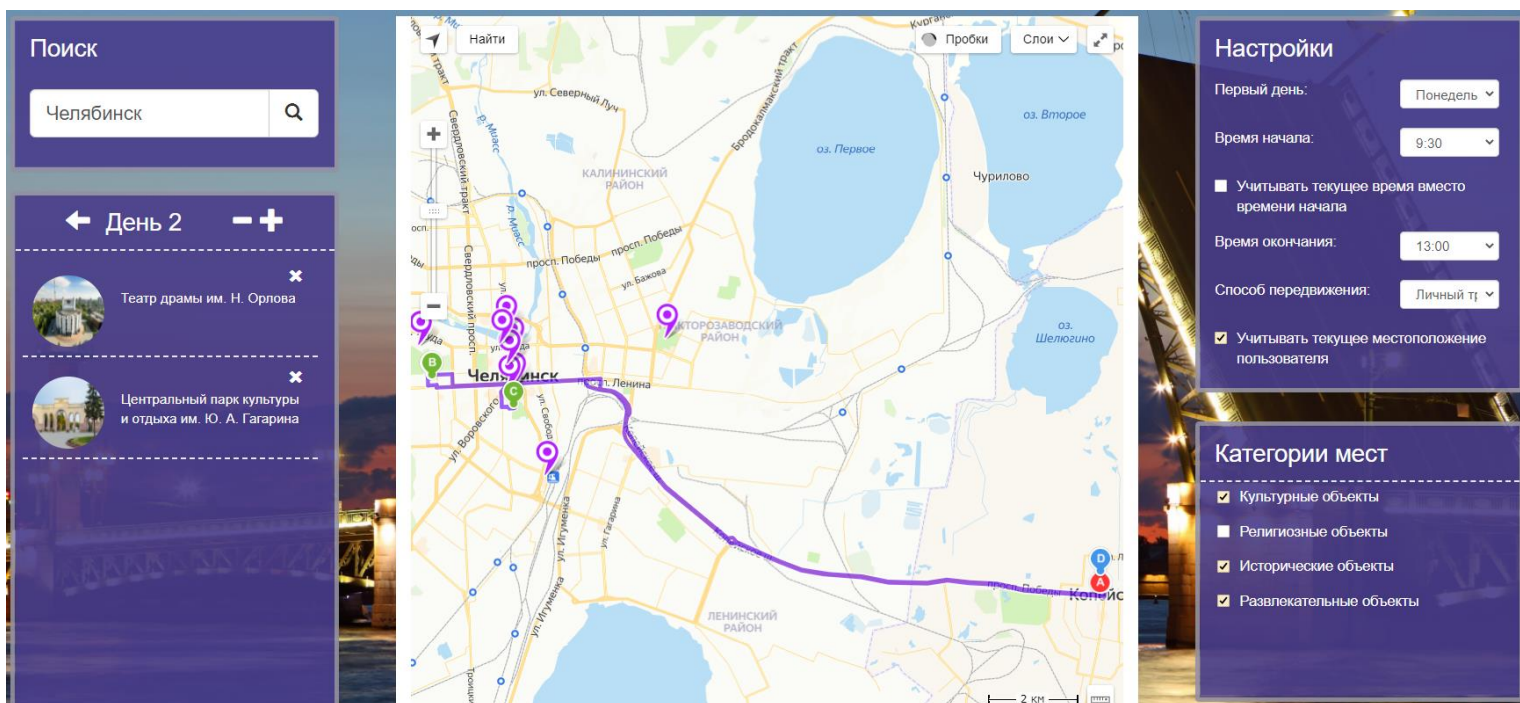


Рисунок 5.6 – Смена способа передвижения

При смене способа передвижения время, необходимое на перемещение до города уменьшается, и пользователь может посетить дополнительные достопримечательности в пределах имеющихся ограничений по времени.

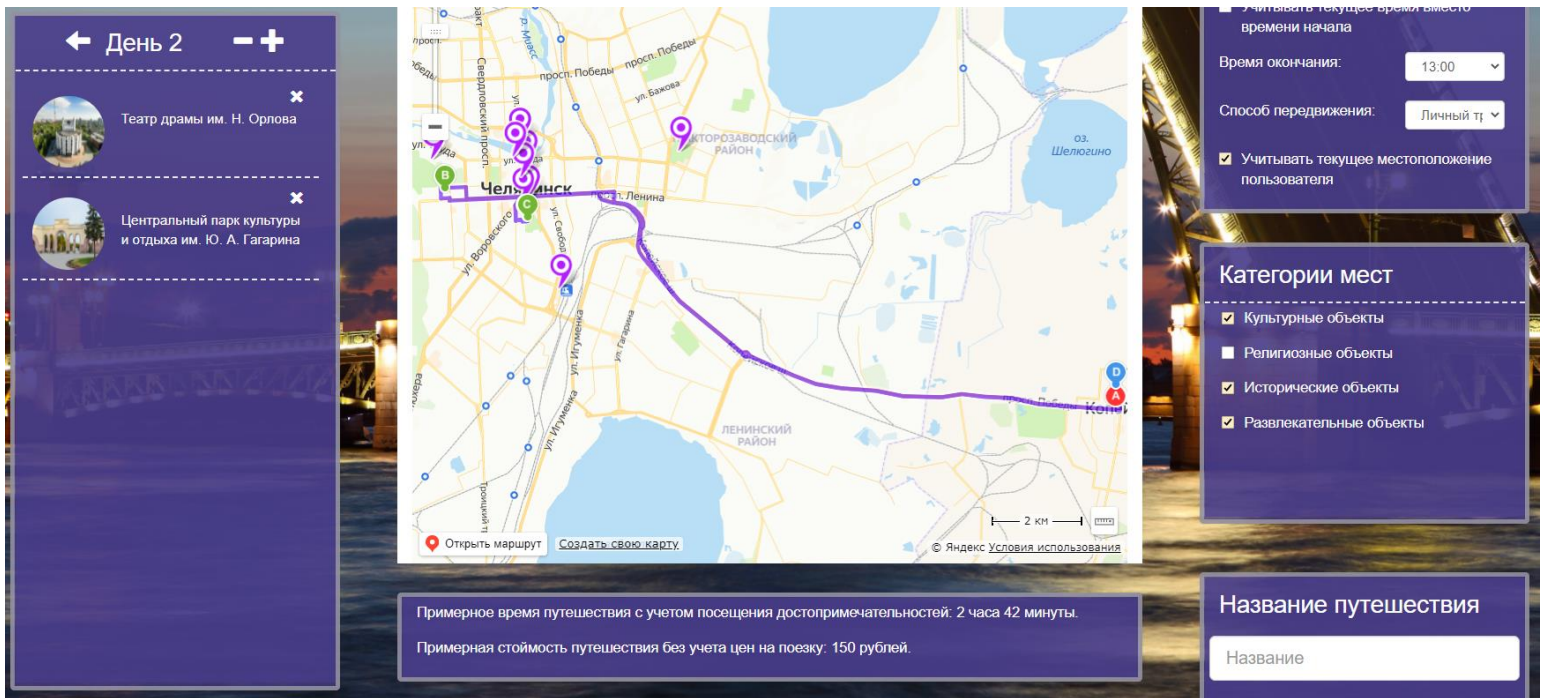


Рисунок 5.7 – Вычисляемое время и стоимость путешествия

Из сравнения рисунка 5.7 с ограничениями на рисунке 5.6 видно, что примерное время путешествия входит в эти рамки.

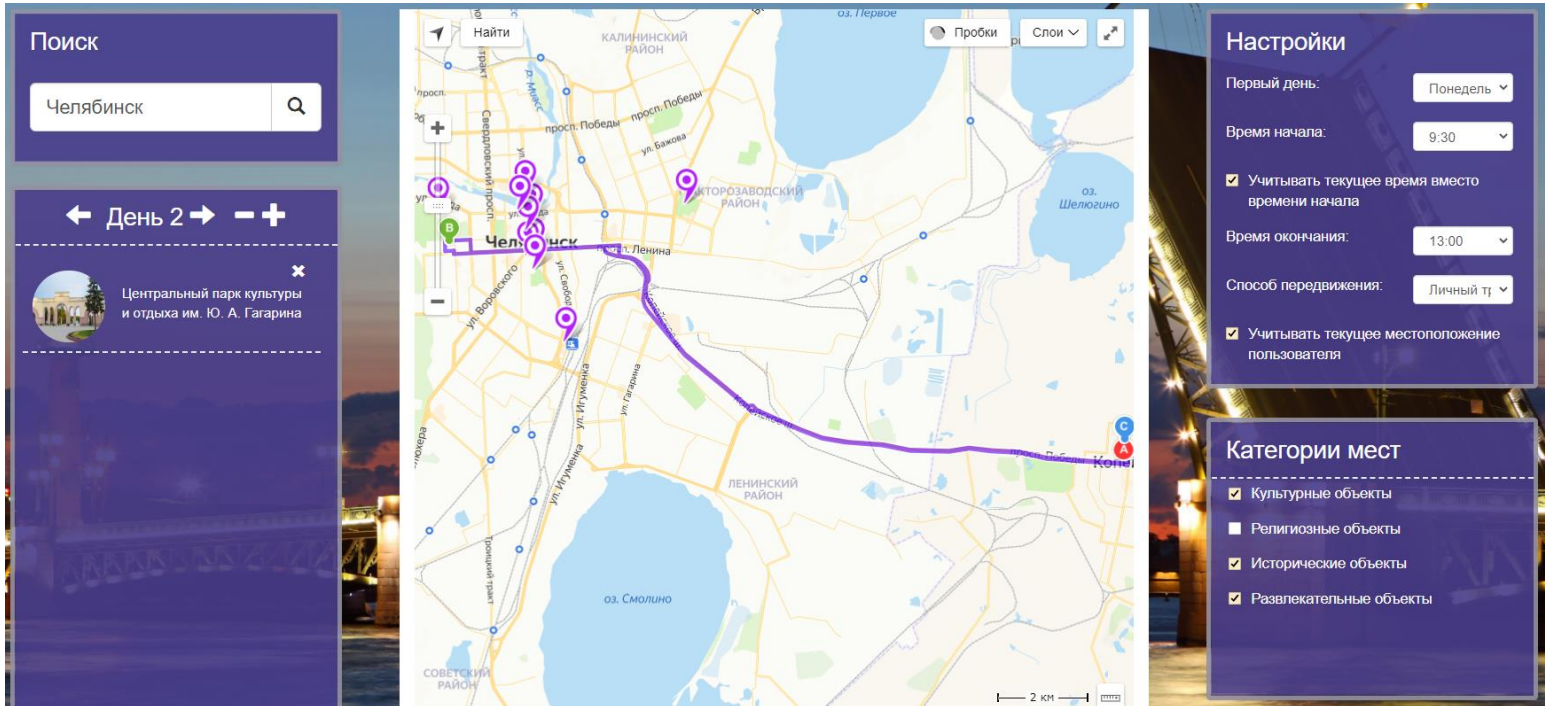


Рисунок 5.8 – Включение учета реального времени

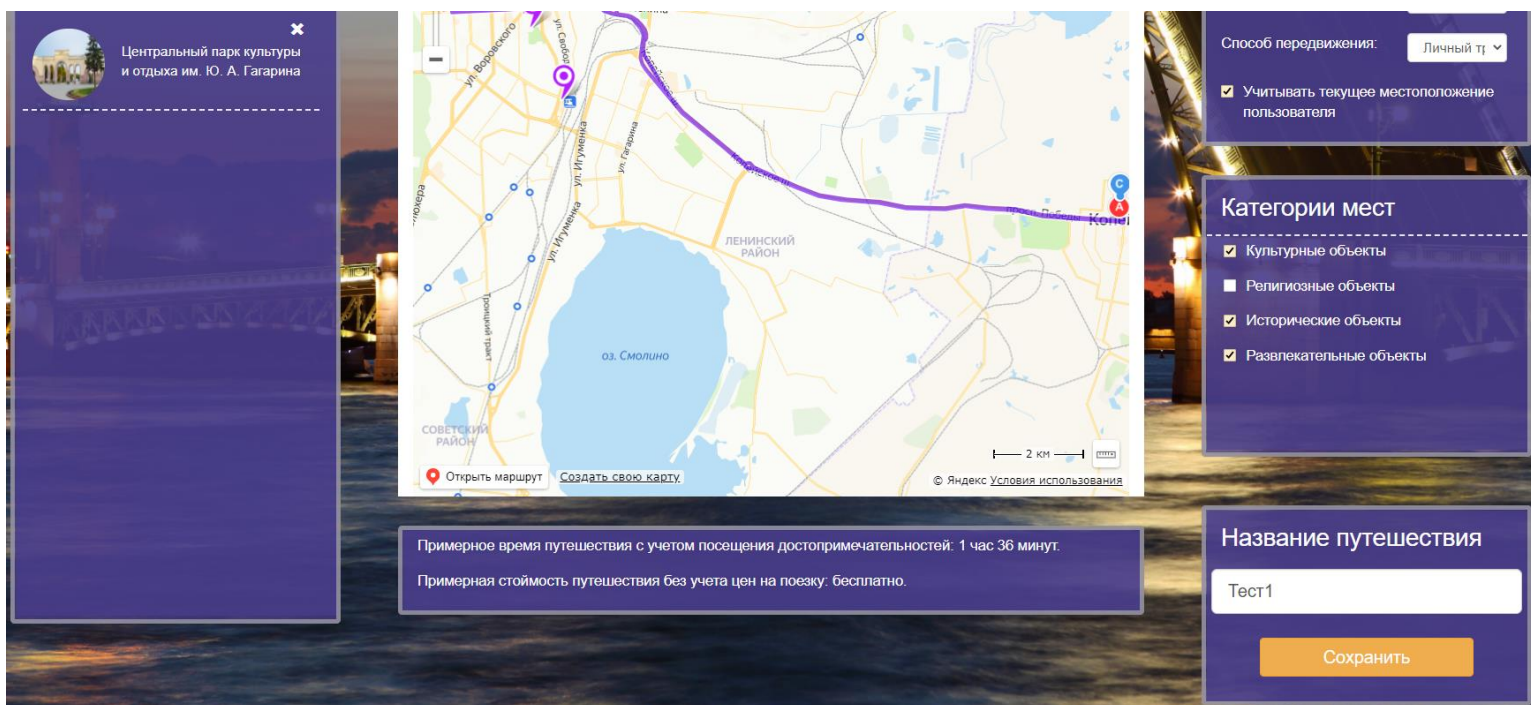


Рисунок 5.9 – Вычисляемое время путешествия при установке учета реального времени

На момент проведения тестирования было время 11:14, из этого получается, что отведенное время на путешествие равно 1 часу 46 минутам. На рисунке 5.7 потребное время 2 часа 42 минуты, поскольку с новыми ограничениями посещение второй достопримечательности не укладывалось во временные рамки, оно было перенесено на следующий день.

2. Тестирование фильтра достопримечательностей.

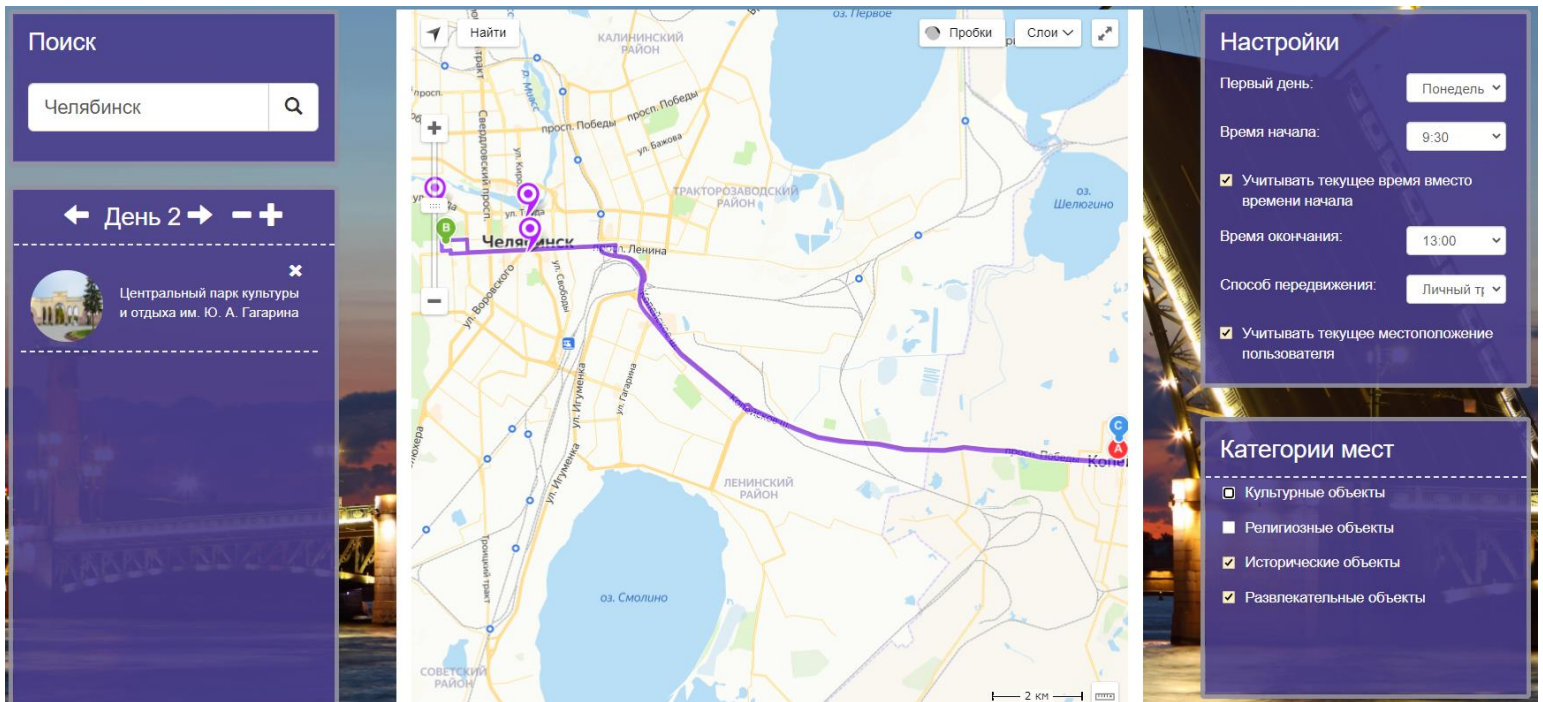


Рисунок 5.10 – Скрытие культурных объектов с карты

После изменение предпочитаемых категорий достопримечательностей часть мест была скрыта с карты.

3. Сохранение маршрута.

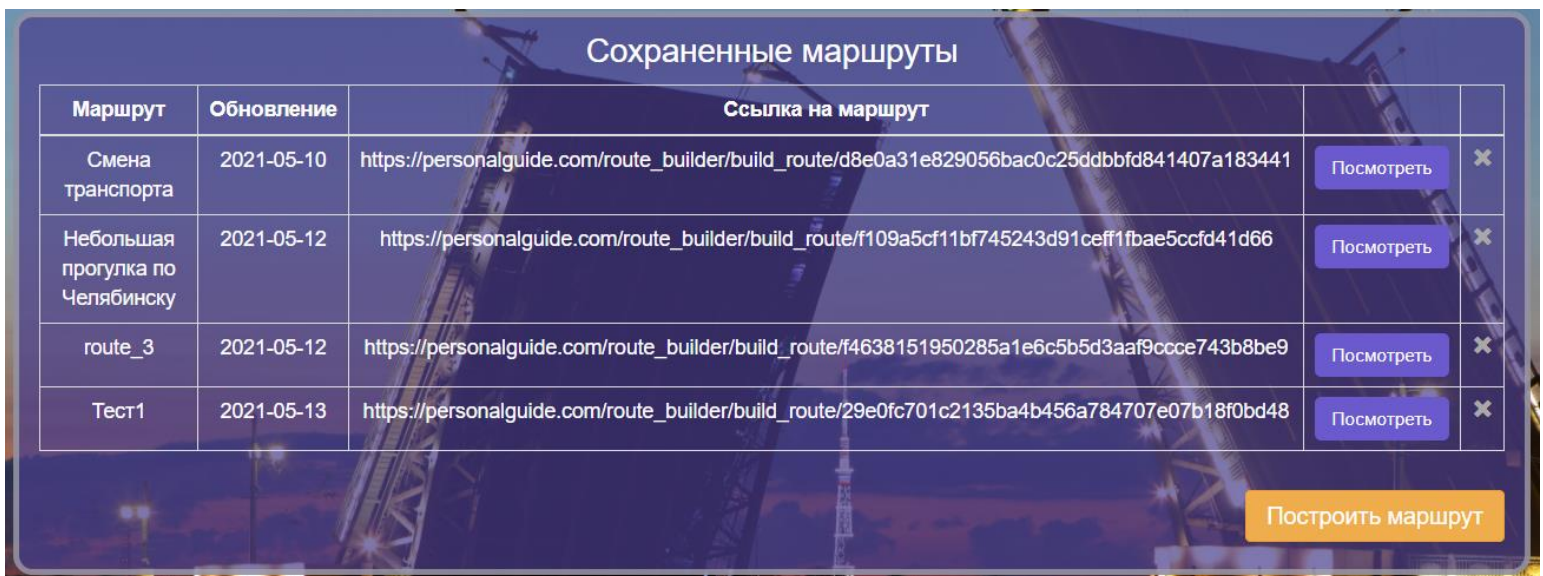


Рисунок 5.11 – Список сохраненных маршрутов

На рисунке 5.9 в названии маршрута написано «Тест1», после сохранения в списке сохраненных маршрутов появился маршрут с названием «Тест1».

4. Добавление новой достопримечательности.

Добавление достопримечательности

Название достопримечательности

Некорректное название достопримечательности

Описание достопримечательности

Некорректное описание достопримечательности

Категория: Не указана

Адрес

Выберите категорию достопримечательности

Отметьте на карте расположение достопримечательности

Отметить на карте

Режим работы: Круглосуточно

Рисунок 5.12 – Проверка данных, введённых администратором при добавлении достопримечательности

Добавление достопримечательности

Центральный парк культуры и отдыха им. Ю. А. Гагарина

Центральный парк культуры и отдыха им. Ю. А. Гагарина располагается в Челябинске. На облагоустроенной территории сосредоточено множество видов развлечений — аттракционов, спортивных и игровых площадок, контактный зоопарк, летний кинотеатр.

Категория: Культурные обы

Россия, Челябинск, улица Коммуны, 100/1

Данная достопримечательность уже добавлена

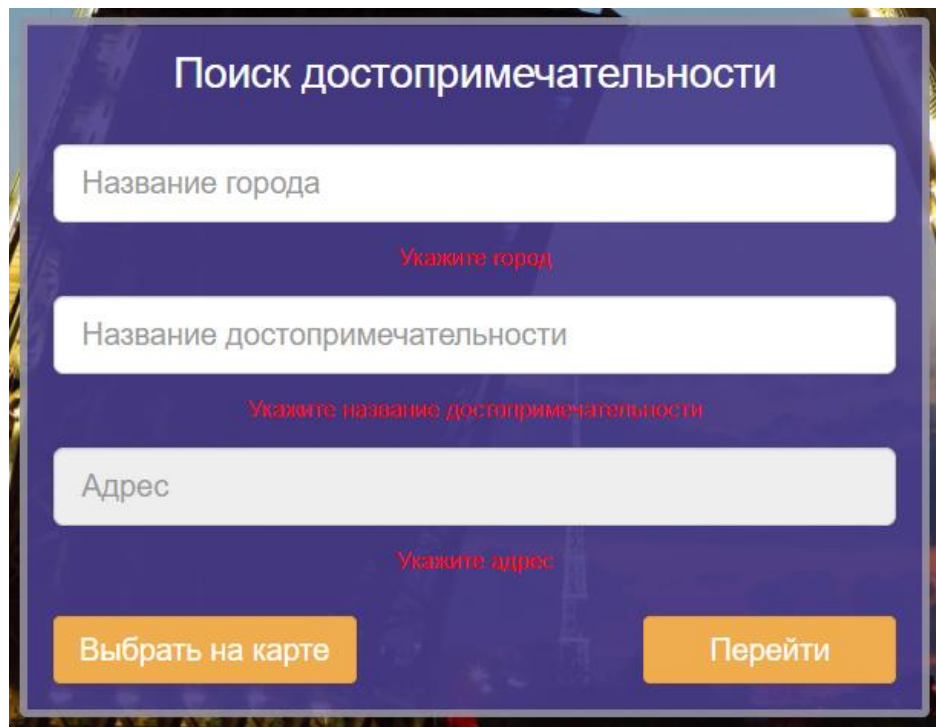
Отметить на карте

Режим работы: Круглосуточно

Цена билета: 0

Рисунок 5.13 – Проверка добавления уже имеющихся достопримечательностей в приложении

5. Поиск достопримечательностей для изменения или удаления.



Поиск достопримечательности

Название города

Укажите город

Название достопримечательности

Укажите название достопримечательности

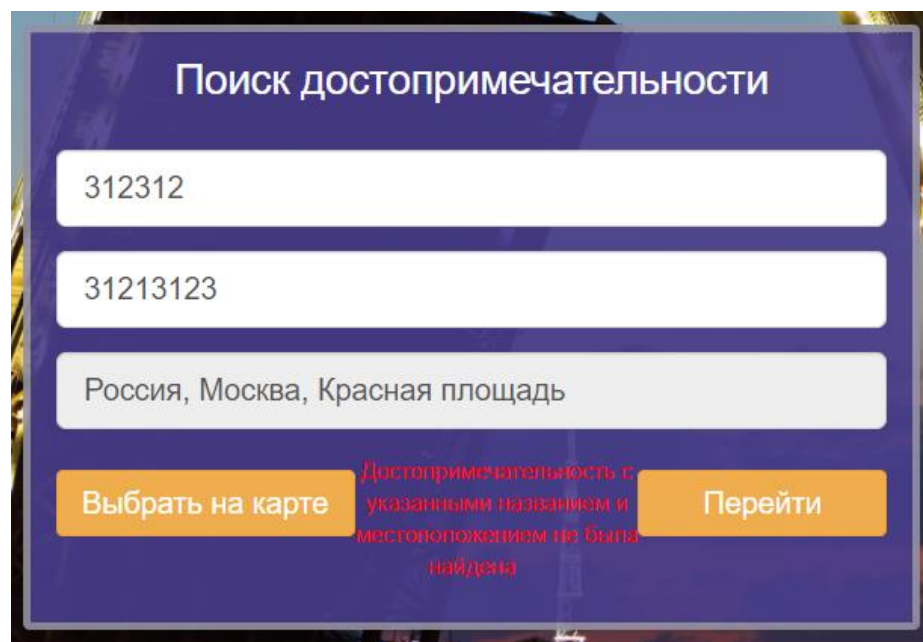
Адрес

Укажите адрес

Выбрать на карте

Перейти

Рисунок 5.14 – Проверка ввода данных



Поиск достопримечательности

312312

31213123

Россия, Москва, Красная площадь

Достопримечательность с указанными названием и местоположением не была найдена

Выбрать на карте

Перейти

Рисунок 5.15 – Проверка наличия искомой достопримечательности

6. Редактирование категорий достопримечательностей.

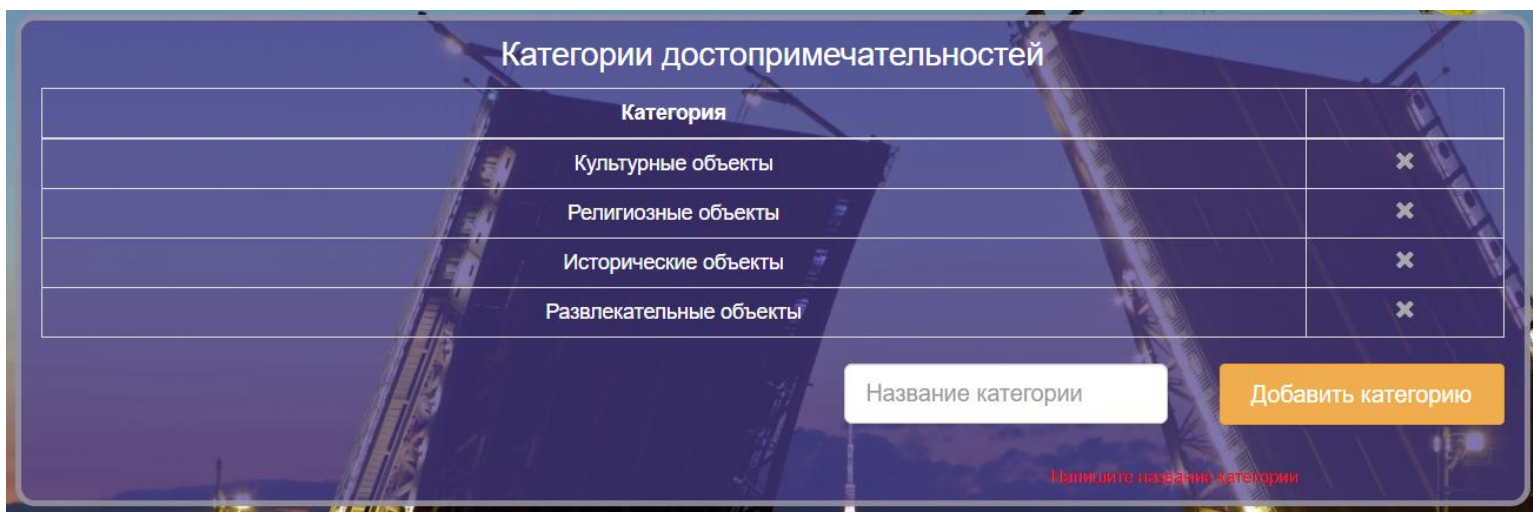


Рисунок 5.16 – Проверка ввода названия категории

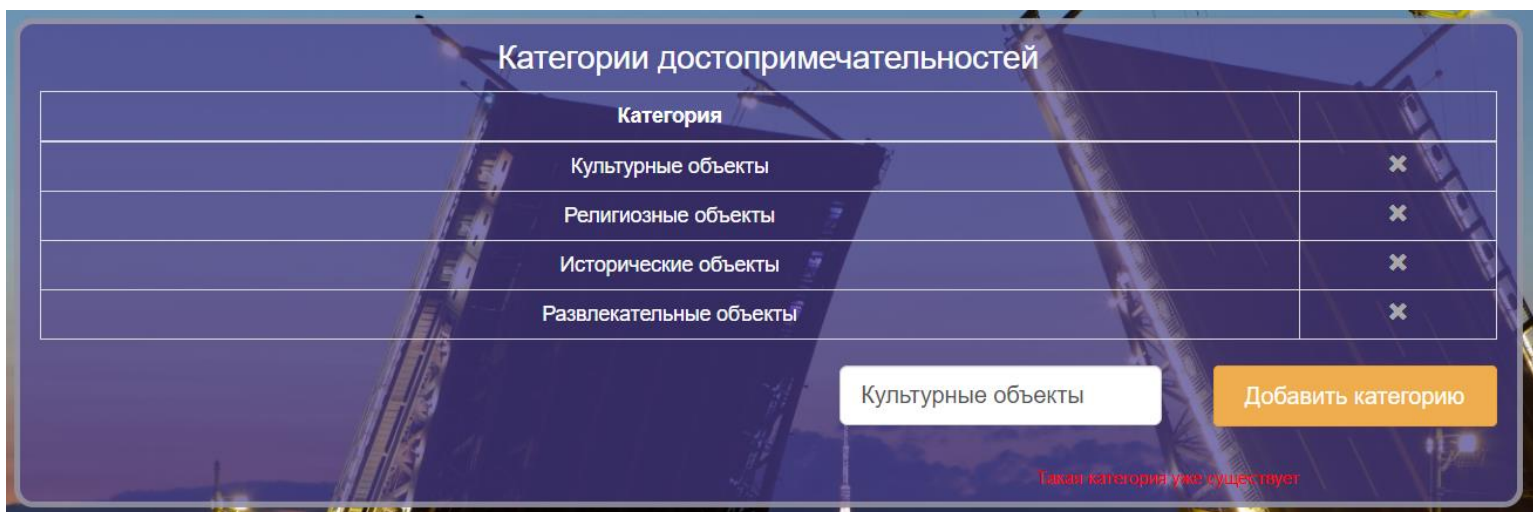


Рисунок 5.17 – Проверка добавления существующей категории

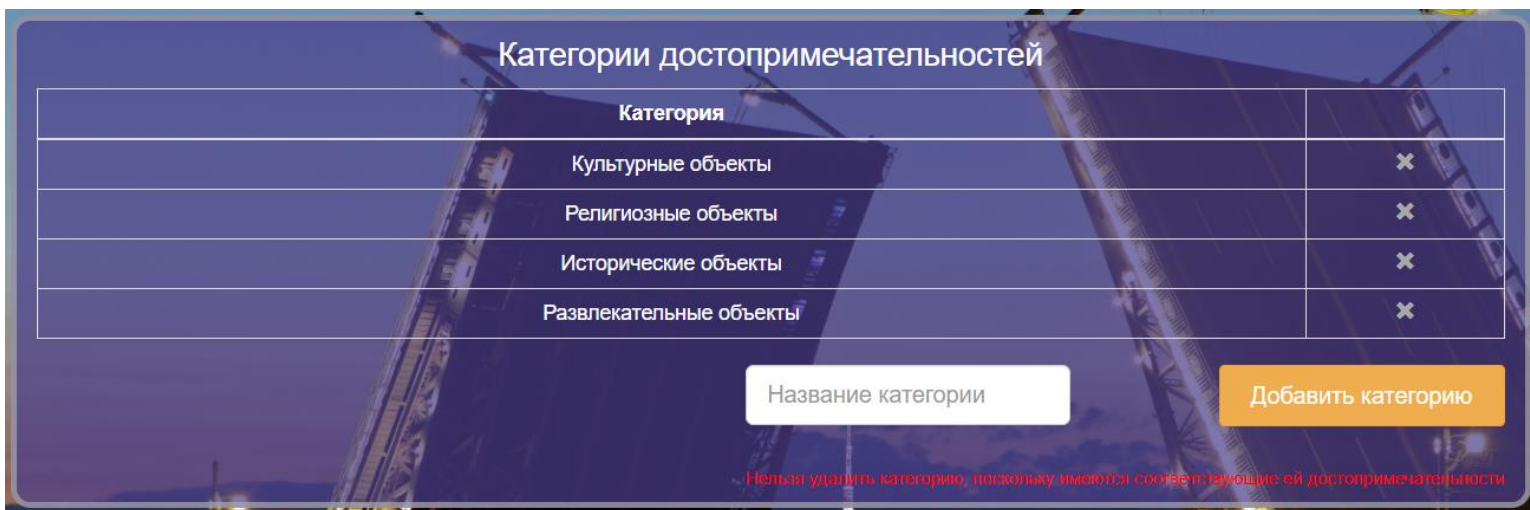


Рисунок 5.18 – Проверка удаления категории, которой соответствуют достопримечательности

7. Написание заявки на добавление достопримечательности.

The image shows a web form titled "Заявка на добавление достопримечательности" (Application for adding a landmark). The form is set against a dark blue background with a faint map of a city. It contains several input fields and a dropdown menu, all of which have red error messages below them. At the bottom, there are two orange buttons: "Отметить на карте" (Mark on map) and "Отправить" (Send).

Заявка на добавление
достопримечательности

Название достопримечательности

Некорректное название достопримечательности

Описание достопримечательности

Некорректное описание достопримечательности

Категория: Не указана ▾

Своя категория достопримечательности

Выберите категорию достопримечательности из имеющихся, либо напишите собственную

Адрес

Отметьте на карте расположение достопримечательности

Отметить на карте

Отправить

Рисунок 5.19 – Проверка ввода данных

Заявка на добавление достопримечательности

Центральный парк культуры и отдыха им. Ю. А. Гагарина

Центральный парк культуры и отдыха им. Ю. А. Гагарина располагается в Челябинске. На облагороженной территории сосредоточено множество видов развлечений — аттракционов, спортивных и игровых площадок, контактный зоопарк, летний кинотеатр.

Категория: Не указана ▼

11111

Россия, Челябинск, улица Коммуны, 100/1

Данная достопримечательность уже добавлена, либо создана заявка на добавление данной достопримечательности

Отметить на карте Отправить

Рисунок 5.20 – Проверка наличия достопримечательности в приложении

8. Проверка заявки написанной пользователем.

это был единственный в стране кинотеатр с двумя залами. В 2000 году кинотеатр объединили с театром «Манекен». На сегодняшний момент «Манекен» является одним из самых известных театров Челябинска. Основанный в 1963 году как студенческое творческое объединение, за несколько десятилетий театр вышел на профессиональный уровень. В его репертуаре есть спектакли для маленьких и больших зрителей.

Категория достопримечательности: Культурные объекты

Адрес: Россия, Челябинск, улица Пушкина, 64

[Посмотреть на карте](#)

Комментарий для пользователя

Статус: Отклонена

Укажите причину отказа в комментарии

[Сохранить](#)

Рисунок 5.21 – Проверка наличия причины отказа при отклонении заявки

Добавление достопримечательности

Театр «Манекен»

Здание в стиле советского конструктивизма построили в 1937 году специально для кинотеатра им. Пушкина (отсюда и изображение поэта на фасаде). На момент строительства это был единственный в стране кинотеатр с двумя залами. В 2000 году кинотеатр объединили с театром «Манекен».

На сегодняшний момент «Манекен» является одним

Категория: Культурные обы

Россия, Челябинск, улица Пушкина, 64

Отметить на карте

Режим работы: Круглосуточно

Цена билета: 0

Выберите файл Файл не выбран

Рисунок 5.22 – Проверка перенаправления на добавление достопримечательности в случае одобрения заявки

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы было разработано приложение-навигатор для построения адаптивного маршрута по достопримечательностям.

При этом были решены следующие задачи:

1. Проведен анализ аналогов, выделены их основные недостатки.
2. Проведено формирование требований к системе.
3. Выполнено проектирование системы.
4. Выполнена реализация системы.
5. Проведено тестирование системы.

Разработанное приложение лишено недостатков аналогов в построении маршрута, а также обладает дополнительными возможностями, обеспечивающими более гибкую настройку маршрута для дальнейшего использования с возможностью перестроения маршрута под изменяющееся время и местоположение пользователя. На каждый день путешествия пользователю предоставляется сводка о стоимости посещения достопримечательностей и потребном времени.

Созданные маршруты пользователь может сохранить и затем посмотреть в личном кабинете, где также имеется возможность задания личных предпочтений, которые будут автоматически учитываться при построении маршрута.

Для администратора представлен функционал поддержки приложения без знания его архитектуры. Имеется возможность добавления, изменения и удаления достопримечательностей в приложении, редактирования категорий достопримечательностей и поддержания обратной связи с пользователями.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Достопримечательности России: музеи, архитектура, усадьбы, памятники. – <https://culttourism.ru/>. Дата обращения: 25.11.2020.
2. Культурно-познавательный туризм. – https://spravochnick.ru/turizm/kulturno-poznavatelnyy_turizm/. Дата обращения: 25.11.2020.
3. О проекте Waytips. – <http://waytips.com/about/>. Дата обращения: 5.01.2021.
4. О проекте Youroute. – <https://youroute.ru/about/>. Дата обращения: 5.01.2021.
5. SYGIC TRAVEL MAPS. – <https://travel.sygic.com/en>. Дата обращения: 5.01.2021.
6. КОМАНДА EVERTRAVEL. – https://evertravel.me/ru/about_us. Дата обращения: 5.01.2021.
7. PHP, Ruby, Python – характеристика трёх языков программирования. – <https://internet--technologies-ru.turbopages.org/internet-technologies.ru/s/articles/php-ruby-python-harakteristika-yazykov-programmirovaniya.html>. Дата обращения: 6.01.2021.
8. Плюсы и минусы программирования на Java. – <https://nuancesprog.ru/p/2234/>. Дата обращения: 6.01.2021.
9. Node.js и JavaScript для серверной разработки. – <https://habr.com/ru/company/ruvds/blog/345164/>. Дата обращения: 6.01.2021.
10. CMS или Фреймворк? Что выбрать?. – <https://vc.ru/flood/33985-cms-ili-freymvork-chto-vybrat>. Дата обращения: 6.01.2021.

11. CMS или фреймворк – что лучше?. – <http://cccp-blog.com/sozдание-saytov/cms-ili-frejmvork>. Дата обращения: 6.01.2021.
12. Обзор PHP фреймворков. – <https://unetway.com/blog/php-framework-review>. Дата обращения: 7.01.2021.
13. SQLite vs MySQL vs PostgreSQL: сравнение систем управления базами данных. – <https://devacademy.ru/article/sqlite-vs-mysql-vs-postgresql/>. Дата обращения: 7.01.2021.
14. Сравнение современных СУБД. – <https://drach.pro/blog/hi-tech/item/145->. Дата обращения: 7.01.2021.
15. Адаптивная вёрстка Bootstrap. – <https://fokit.ru/adaptivnaya-vyorstka-bootstrap/>. Дата обращения: 7.01.2021.
16. Яндекс.Карты, 2ГИС или всё же Google Maps?. – <https://habr.com/ru/post/242015/>. Дата обращения: 7.01.2021.
17. Браузеры в России. – https://radar.yandex.ru/browsers?selected_rows=Ct58LP%252CRysHuf%252CnmpVtr%252C%252BjXhkh%252C2kuSN7%252Cs%252FDH9r%252C%252F127zq%252CTRNLwH%252CkCujza%252Ce1IAm2. Дата обращения: 7.01.2021.
18. Архитектура MVC. – <https://github.com/codedokode/pasta/blob/master/arch/mvc.md>. Дата обращения: 7.01.2021.
19. Bagchi, T. A review of TSP based approaches for flowshop scheduling / T. Bagchi, J. Gupta, C. Sriskandarajah // European Journal Operations Research. – 2006. – V. 169, N 3. – P. 816–854.

20. Ben-Arieh, D. Transformations of generalized ATSP into ATSP / D. Ben-Arieh, G. Gutin, M. Penn et al. // Operations Research. – 2003. V. 31, N 5. – P. 357–365.
21. Kumar, R. On Asymmetric TSP: Transformation to Symmetric TSP and Performance Bound / R. Kumar, H. Li // Journal of Operations Research. – 1996. – V. 23, N 2. – P. 234–244.
22. ЗАДАЧА КОММИВОЯЖЕРА – МЕТОД ВЕТВЕЙ И ГРАНИЦ. – <http://galyautdinov.ru/post/zadacha-kommivoyazhera#mark02>. Дата обращения: 21.03.2021.
23. Гладков, Л. А. Генетические алгоритмы: учебное пособие / Л. А. Гладков, В. В. Курейчик, В. М. Курейчик; под ред. В. М. Курейчик – 2-е изд. – Москва: Физматлит, 2006. – 320 с.
24. Blum, C. Ant colony optimization: Introduction and recent trends / C. Blum // Physics of Life Reviews. – 2005. – P. 353–373.
25. Сравнение производительности генетических алгоритмов и муравьиных алгоритмов применительно к задаче коммивояжера. – http://masters.donntu.org/2018/fknt/brazhnik/library/article_translate.html. Дата обращения: 21.03.2021.
26. Штовба, С.Д. Exponenta Pro. Математика в приложениях / С.Д. Штовба // Муравьиные алгоритмы. – 2003. – № 4. – С. 70–75.

ПРИЛОЖЕНИЕ А

Листинг А.1 – Исходный код страницы построения маршрута

```
ymaps.ready(init);

function init() {
    var input_city = document.getElementById("city"),
        list_city,
        request = new XMLHttpRequest(),
        myMap, myRoute,
        real_location = document.getElementById('real_location'),
        first_day = document.getElementById('first_day'),
        start_time = document.getElementById('start_time'),
        end_time = document.getElementById('end_time'),
        real_start_time = document.getElementById('real_start_time'),
        list_times;

    request_times = new XMLHttpRequest();
    request_times.open('GET', '/times.json');
    request_times.setRequestHeader('Content-type', 'application/json;
charset=utf-8');
    request_times.send();

    request_times.addEventListener('readystatechange', function() {
        if(request_times.readyState == 4 && request_times.status == 200) {
            list_times = JSON.parse(request_times.response);
        }
    });

    request.open('GET', '/cities.json');
    request.setRequestHeader('Content-type', 'application/json; charset=utf-8');
    request.send();

    request.addEventListener('readystatechange', function() {
        if(request.readyState == 4 && request.status == 200) {
            list_city = JSON.parse(request.response);
            var city = input_city.value;
            if(!check_city(city, list_city)) {
                city = 'Москва';
            }
            goToAddress(city);
        }
    });

    function check_city(city, list_city) {
        var check = false;
        for(var i = 0; i < list_city.length; i++) {
```

```

        if(city == list_city[i]) {
            check = true;
            break;
        }
    }
    console.log(check);
    return check;
}

var buttonSearch = document.getElementById('search_city');

buttonSearch.addEventListener('click', function() {
    console.log(input_city.value);
    var city;
    if(!check_city(input_city.value, list_city)) {
        alert(`К сожалению, город ${input_city.value} находится ещё в
разработке.`);
    }
    else {
        city = input_city.value;
        goToAddress(city);
    }
});

input_city.addEventListener('keyup', function(e) {
    if (e.code == 'Enter') {
        var city;
        if(!check_city(input_city.value, list_city)) {
            alert(`К сожалению, город ${input_city.value} находится ещё в
разработке.`)
        }
        else {
            city = input_city.value;
            goToAddress(city);
        }
    }
});

function goToAddress(name) {
    ymaps.geocode(name).then(function (res) {
        setCenter(res.geoObjects.get(0).geometry.getCoordinates());
    });
}

function setCenter(coords) {
    console.log(coords);
    if(myMap) {
        myMap.setCenter(coords);
    }
}

```

```

else {
    myMap = new ymaps.Map('map', {
        center: coords,
        zoom : 12
    }, {
        searchControlProvider: 'yandex#search'
    });
    loadPlaces();
}

function createPlacemark(id, name, country, city, location_place_address,
location_place_coordinates) {
    var address = `${country}, ${city}, ${location_place_address}`;
    return new ymaps.Placemark(location_place_coordinates.split(','), {
        idPlace:id,
        balloonContentHeader:name,
        balloonContentBody:address,
        balloonContentFooter:"<a id='add-place'>добавить</a> <a id='view-
info-place' data-toggle='modal' data-target='#exampleModal'>подробнее</a>",
        hintContent: name
    }, {
        preset: 'islands#violetDotIconWithCaption',
        draggable: false
    });
}

var places,
    categories,
    display_places = [];
categories_on_page = document.getElementsByClassName('category');

//первая инициализация достопримечательностей на карте
function loadPlaces() {
    var request = new XMLHttpRequest();
    request.open('GET', '/categories.json');
    request.setRequestHeader('Content-type', 'application/json; charset=utf-
8');
    request.send();

    request.addEventListener('readystatechange', function(){

        if(request.readyState == 4 && request.status == 200) {
            categories = JSON.parse(request.response);

            var request_places = new XMLHttpRequest();
            request_places.open('GET', '/places.json');

```

```

        request_places.setRequestHeader('Content-type',
'application/json; charset=utf-8');
        request_places.send();

        request_places.addEventListener('readystatechange', function(){
            if(request_places.readyState == 4 && request_places.status
== 200) {

                places = JSON.parse(request_places.response);
                reloadPlaces();
                for(var j = 0; j < categories_on_page.length; j++) {
                    categories_on_page[j].addEventListener('click',
function () {
                        reloadPlaces();
                    });
                }
                load_route();
            }
        });
    });

    //события на нажатие кнопок в балуне
    myMap.geoObjects.events.add('click', function (e) {

        var placemark = e.get('target');

        placemark.events.add('balloonopen', function (e) {

            var view = document.getElementById('view-info-place'),
                add = document.getElementById('add-place'),
                id_place = placemark.properties.get('idPlace');

            view.addEventListener('click', function (e) {
                view_info_place(id_place);
            });

            add.addEventListener('click', function (e) {
                add_place_in_day(id_place);
            });

        });
    });
}

//функция перезагрузки достопримечательностей на карте после смены фильтра
function reloadPlaces() {

```

```

for(var i = 0; i < display_places.length; i++) {
    myMap.geoObjects.remove(display_places[i]);
}
display_places = [];

for(var i = 0; i < places.length; i++) {
    var cur_category;
    for(var j = 0; j < categories.length; j++) {
        if(places[i]['category'] == categories[j]['name_category']) {
            cur_category = categories[j]['id_category'];
            break;
        }
    }
    var checked;

    for(var j = 0; j < categories_on_page.length; j++) {
        if(cur_category == categories_on_page[j].defaultValue) {
            checked = categories_on_page[j].checked;
            break;
        }
    }

    if(checked == true) {
        var flag = true;
        for(var j = 0; j < currentDay.length; j++) {
            if(currentDay[j]['id_place'] == places[i]['id_place']) {
                flag = false;
                break;
            }
        }
        if(flag == true) {
            var myPlacemark = createPlacemark(
                places[i]['id_place'],
                places[i]['name_place'],
                places[i]['country'],
                places[i]['city'],
                places[i]['location_place_address'],
                places[i]['location_place_coordinates']
            );
            display_places[display_places.length] = myPlacemark;
            //console.log(myPlacemark);
            myMap.geoObjects.add(myPlacemark);
        }
    }
}

var cur_location;

```

```

if(real_location.checked == true) {
    ymaps.geolocation.get({provider: "browser"}).then(function (res) {
        cur_location = res.geoObjects.get(0).geometry.getCoordinates();
        setTimeout(() => { build_route(currentDay, myMap); }, 100);
    }, function (e) {

    });
}

var timer_real_location;
//обработчик события учета текущего местоположения пользователя.

real_location.addEventListener('click', function () {
    window.clearInterval(timer_real_location);
    if(real_location.checked == true) {
        set_cur_location();
        timer_real_location = window.setInterval(set_cur_location, 300000);
    }
});

first_day.onchange = function () {
    build_route(currentDay, myMap);
}

start_time.onchange = function () {
    build_route(currentDay, myMap);
}

end_time.onchange = function () {
    build_route(currentDay, myMap);
}

var mode_travel = document.getElementById('mode_travel'),
    type_transport;

if(mode_travel.value == "Пешком") {
    type_transport = "pedestrian";
}

else if (mode_travel.value == "Личный транспорт") {
    type_transport = "auto";
}

else if (mode_travel.value == "Общественный транспорт") {

```



```

    type_transport = "masstransit";
}

//Обработчик события изменения значения в способе передвижения
mode_travel.onchange = function () {
    if(mode_travel.value == "Пешком") {
        type_transport = "pedestrian";

    }
    else if (mode_travel.value == "Личный транспорт") {
        type_transport = "auto";
    }
    else if (mode_travel.value == "Общественный транспорт") {
        type_transport = "masstransit";
    }
    build_route(currentDay, myMap);
}

function deletePlacemark(id_place) {
    for(var i = 0; i < display_places.length; i++) {
        if(display_places[i].properties.get('idPlace') == id_place) {
            myMap.geoObjects.remove(display_places[i]);
            display_places.splice(i, 1);
            break;
        }
    }
}

function addPlacemark(id_place) {
    for(var i = 0; i < places.length; i++) {
        if(places[i]['id_place'] == id_place) {
            var cur_category;
            for(var j = 0; j < categories.length; j++) {
                if(places[i]['category'] == categories[j]['name_category'])
                {
                    cur_category = categories[j]['id_category'];
                    break;
                }
            }

            var checked;

            for(var j = 0; j < categories_on_page.length; j++) {
                if(cur_category == categories_on_page[j].defaultValue) {
                    checked = categories_on_page[j].checked;
                    break;
                }
            }
        }
    }
}

```

```

        if (checked == true) {
            var myPlacemark = createPlacemark(
                places[i]['id_place'],
                places[i]['name_place'],
                places[i]['country'],
                places[i]['city'],
                places[i]['location_place_address'],
                places[i]['location_place_coordinates']
            );
            display_places[display_places.length] = myPlacemark;
            console.log(myPlacemark);
            myMap.geoObjects.add(myPlacemark);
        }
        break;
    }
}

function add_place_in_day(id_place) {
    var place;
    for (var i = 0; i < places.length; i++) {
        if (places[i]['id_place'] == id_place) {
            place = places[i];
        }
    }

    var check = true;
    for (var i = 0; i < currentDay.length; i++) {
        if (place['id_place'] == currentDay[i]['id_place']) {
            check = false;
        }
    }

    if (check) {

        deletePlacemark(place['id_place']);
        currentDay[currentDay.length] = place;

        build_route(currentDay, myMap);

        var ul_places = document.getElementById('ul_places');

        var url_image =
        `../../assets/img/${place['id_place']}_imageplace.jpg`;
        ul_places.innerHTML += `- 

```

```

        <img class="image-place-s" src=${url_image} alt="">
    </div>
    <div class="col-lg-8 col-md-8 col-sm-8 col-xs-8 name-place">
        <div class="personal-account-text btn-delete">
            <a id="remove_${place['id_place']}" href="#"
class="glyphicon glyphicon-remove"></a>
        </div>
        <div id="get_info_${place['id_place']}" data-toggle='modal'
data-target='#exampleModal'>${place['name_place']}</div>
    </div>
    </li>`;
    }
    else {
    }
}
}

var ul_places = document.getElementById('ul_places');
ul_places.addEventListener("click", function (e) {
    if(e.target.id.startsWith('get_info_')) {
        view_info_place(e.target.id.split('_')[2]);
    }
    else if(e.target.id.startsWith('remove_')) {
        for(var i = 0; i < currentDay.length; i++) {
            if(e.target.id.split('_')[1] == currentDay[i]['id_place']) {
                var elem = document.getElementById('place_' +
e.target.id.split('_')[1]);
                elem.remove();
                addPlacemark(e.target.id.split('_')[1]);
                currentDay.splice(i, 1);
            }
        }
        build_route(currentDay, myMap);
    }
});

//вывод информации о достопримечательности
function view_info_place(id_place) {
    let title = document.getElementById('exampleModallLabel'),
        body = document.getElementById('info-body'),
        place;
    for(var i = 0; i < places.length; i++) {
        if(places[i]['id_place'] == id_place) {
            place = places[i];
        }
    }
}

```

```

    }

    title.innerHTML = place['name_place'];

    var description = "Описание: " + place['description_place'] + "<br/>";
    var category = "Категория достопримечательности: " + place['category'] +
    "<br/>";
    var address = `Местоположение: ${place['country']}, ${place['city']},
    ${place['location_place_address']}<br/>`;
    var array_working_week = place['working_week_schedule'];
    var working_week = 'График работы: ';
    if(array_working_week == null) {
        working_week += 'круглосуточно.';
    }
    else {
        working_week += '<br/>';
        for(var i = 0; i < array_working_week.length; i++) {
            if(array_working_week[i]['type'] == 1) {
                working_week += `${array_working_week[i]['day']}:
    ${array_working_week[i]['opening_time']} -
    ${array_working_week[i]['closing_time']}<br/>`;
            }
            else {
                working_week += `${array_working_week[i]['day']}:
    выходной<br/>`;
            }
        }
    }

    var ticket;
    if(place['ticket_price'] == 0) {
        ticket = "";
    }
    else {
        ticket = "Цена билета: " + place['ticket_price'];
    }

    body.innerHTML =
    `${description}<br/>${category}<br/>${address}<br/>${working_week}<br/>${ticket}
    `;
}

//Массив дней, обработка дней и события на взаимодействие с кнопками

var leftDay = document.getElementById('left-day'),
    rightDay = document.getElementById('right-day'),
    deleteDay = document.getElementById('delete-day'),
    addDay = document.getElementById('add-day'),

```

```

val_day = document.getElementById('val-day'),
listDays = [],
currentDay = listDays[0],
num_select_day = 0;

//загрузка сохраненного маршрута
var hesh = document.getElementById('hesh');
function load_route() {
    if(hesh.value != "") {
        request_route = new XMLHttpRequest();
        request_route.open('GET', '/loading_route.json');
        request_route.setRequestHeader('Content-type', 'application/json;
charset=utf-8');
        request_route.send();
        let list_places;
        request_route.addEventListener('readystatechange', function() {
            if(request_route.readyState == 4 && request_route.status == 200)
{
                list_places = JSON.parse(request_route.response);
                for(let i = 0; i < list_places.length; i++) {
                    for(let j = 0; j < places.length; j++) {
                        if(list_places[i]['id_place'] ==
places[j]['id_place']) {
                            if(listDays[list_places[i]['num_day']] ==
undefined) {
                                listDays[list_places[i]['num_day']] = [];
                            }
                        }
                    }
                }
                listDays[list_places[i]['num_day']][listDays[list_places[i]['num_day']].length]
= places[j];
            }
        }
        currentDay = listDays[0];
        if(listDays.length > 1) {
            rightDay.style.opacity = "1";
            deleteDay.style.opacity = "1";
        }
    }
    for(var i = 0; i < currentDay.length; i++) {
        deletePlacemark(currentDay[i]['id_place']);
        var url_image =
`'../../assets/img/${currentDay[i]['id_place']}_imageplace.jpg`;
        ul_places.innerHTML += `<li
id="place_${currentDay[i]['id_place']}" class="selected_item_place">
        <div class="col-lg-3 col-md-3 col-sm-3 col-xs-3 con-
image">
            <img class="image-place-s" src=${url_image} alt="">

```

```

        </div>
        <div class="col-lg-8 col-md-8 col-sm-8 col-xs-8 name-
place">
            <div class="personal-account-text btn-delete">
                <a id="remove_${currentDay[i]['id_place']}"
href="#" class="glyphicon glyphicon-remove"></a>

                </div>
                <div id="get_info_${currentDay[i]['id_place']}"
data-toggle='modal' data-
target='#exampleModal'>${currentDay[i]['name_place']}</div>
            </div>
        </li>`;
    }
    build_route(currentDay, myMap);
});
}
//console.log(listDays);
}

addDay.addEventListener('click', function () {
    listDays[listDays.length] = [];
    for(var i = 0; i < currentDay.length; i++) {
        var elem = document.getElementById('place_' +
currentDay[i]['id_place']);
        elem.remove();
        addPlacemark(currentDay[i]['id_place']);
    }

    currentDay = listDays[listDays.length - 1];

    for(var i = 0; i < currentDay.length; i++) {
        deletePlacemark(currentDay[i]['id_place']);
    }

    build_route(currentDay, myMap);

    val_day.innerHTML = listDays.length;
    num_select_day = listDays.length - 1;

    leftDay.style.opacity = "1";
    deleteDay.style.opacity = "1";
    rightDay.style.opacity = "0";
    //console.log(listDays);
});

deleteDay.addEventListener('click', function () {
    if(listDays.length > 1) {

```

```

    for(var i = 0; i < currentDay.length; i++) {
        var elem = document.getElementById('place_' +
currentDay[i]['id_place']);
        elem.remove();
        addPlacemark(currentDay[i]['id_place']);
    }

    console.log(num_select_day);
    listDays.splice(num_select_day, 1);

    if(listDays[num_select_day] == null) {
        currentDay = listDays[num_select_day - 1];

        for(var i = 0; i < currentDay.length; i++) {
            deletePlacemark(currentDay[i]['id_place']);
            var url_image =
`'../../assets/img/${currentDay[i]['id_place']}_imageplace.jpg`;
            ul_places.innerHTML += `- 

```

```

ul_places.innerHTML += `- 

```



```

num_select_day--;
currentDay = listDays[num_select_day];

build_route(currentDay, myMap);

var ul_places = document.getElementById('ul_places');

for(var i = 0; i < currentDay.length; i++) {
    deletePlacemark(currentDay[i]['id_place']);
    var url_image =
`'../../assets/img/${currentDay[i]['id_place']}_imageplace.jpg`;
    ul_places.innerHTML += `- 

```

```

build_route(currentDay, myMap);

var ul_places = document.getElementById('ul_places');
for(var i = 0; i < currentDay.length; i++) {
    deletePlacemark(currentDay[i]['id_place']);
    var url_image =
`'../../assets/img/${currentDay[i]['id_place']}_imageplace.jpg`;

    ul_places.innerHTML += `<li
id="place_${currentDay[i]['id_place']}" class="selected_item_place">
    <div class="col-lg-3 col-md-3 col-sm-3 col-xs-3 con-image">
        <img class="image-place-s" src=${url_image} alt="">
    </div>
    <div class="col-lg-8 col-md-8 col-sm-8 col-xs-8 name-place">
        <div class="personal-account-text btn-delete">
            <a id="remove_${currentDay[i]['id_place']}" href="#"
class="glyphicon glyphicon-remove"></a>
        </div>
        <div id="get_info_${currentDay[i]['id_place']}" data-
toggle='modal' data-target='#exampleModal'>${currentDay[i]['name_place']}</div>
        </div>
    </li>`;
}

val_day.innerHTML = num_select_day + 1;
if(num_select_day == listDays.length - 1) {
    rightDay.style.opacity = "0";
}
leftDay.style.opacity = "1";
}
});

//Параметры алгоритма

var A = 1,
    B = 3,
    ITERATIONS = 100,
    COUNT_ANT = 20,
    Q = 100,
    RO = 0.5;

//определение вероятности перехода в другую точку

function probability(to, ant, pheromone, distance) {
    for(var i = 0; i < ant.array_vertexes.length; i++) {
        if(to == ant.array_vertexes[i]) {
            return 0;
        }
    }
}

```

```

}
for(var i = 0; i < ant.array_unavailable_vertexes.length; i++) {
    if(to == ant.array_unavailable_vertexes[i]) {
        return 0;
    }
}
var sum = 0,
    from = ant.array_vertexes[ant.array_vertexes.length - 1];
for(var i = 0; i < distance.length; i++) {
    var check = 1;

    for(var j = 0; j < ant.array_vertexes.length; j++) {
        if(i == ant.array_vertexes[j]) {
            check = 0;
        }
    }
    for(var j = 0; j < ant.array_unavailable_vertexes.length; j++) {
        if(i == ant.array_unavailable_vertexes[j]) {
            check = 0;
        }
    }

    if(check == 1) {
        sum += Math.pow(pheromone[from][i], A) + Math.pow(1 /
distance[from][i], B);
    }
}
return (Math.pow(pheromone[from][to], A) + Math.pow(1 /
distance[from][to], B) / sum);
}

```

//функция поиска пути

```

function SearchRouteAntColony(distance, vertex, myMap) {
    var main_path = new Path(),
        pheromone = [];

    main_path.length = -1;
    for(let i = 0; i < distance.length; i++) {
        pheromone[i] = [];
        for(let j = 0; j < distance.length; j++) {
            pheromone[i][j] = 1 / distance.length;
        }
    }
    var array_ants = [];
    for(let i = 0; i < COUNT_ANT; i++) {
        array_ants[i] = new Path();
        array_ants[i].length = array_vertexes_weight[vertex];
    }
}

```

```

    array_ants[i].array_vertexes = [];
    array_ants[i].array_vertexes[0] = vertex;
    array_ants[i].array_unavailable_vertexes = [];
  }
  for(let i = 0; i < ITERATIONS; i++) {
    for(let j = 0; j < COUNT_ANT; j++) {
      var count_go = 0;
      array_ants[j].array_unavailable_vertexes = [];
      for(let k = 0; k < curDay_places.length; k++) {
        if(curDay_places[k] != null) {
          if(val_start_time + array_ants[j].length <
curDay_places[k].start || val_start_time + array_ants[j].length >
curDay_places[k].close) {

array_ants[j].array_unavailable_vertexes[array_ants[j].array_unavailable_vertexe
s.length] = k;

          }
        }
      }
      while(array_ants[j].array_vertexes.length +
array_ants[j].array_unavailable_vertexes.length != distance.length &&
array_ants[j].length + val_start_time <= val_end_time && count_go < 1000) {

        let current_vertex,
            current_probability = 0;
        for (let k = 0; k < distance.length; k++) {
          //console.log(array_ants[j].array_vertexes[0]);
          if
(array_ants[j].array_vertexes[array_ants[j].array_vertexes.length - 1] != k) {
            let p = probability(k, array_ants[j], pheromone,
distance);

            if (p && p >= current_probability) {
              current_probability = p;
              current_vertex = k;
            }
          }
        }

        array_ants[j].array_unavailable_vertexes = [];
        if(current_vertex != undefined) {
          array_ants[j].length +=
distance[array_ants[j].array_vertexes[array_ants[j].array_vertexes.length -
1]][current_vertex];
          array_ants[j].length +=
array_vertexes_weight[current_vertex];

```

```

array_ants[j].array_vertexes[array_ants[j].array_vertexes.length] =
current_vertex;
    }
    else {
        count_go++;
    }
    for(let k = 0; k < curDay_places.length; k++) {
        if(curDay_places[k] != null) {
            if(val_start_time + array_ants[j].length <
curDay_places[k].start || val_start_time + array_ants[j].length >
curDay_places[k].close) {

array_ants[j].array_unavailable_vertexes[array_ants[j].array_unavailable_vertexe
s.length] = k;
                }
            }
        }
    }
    for(let k = 0; k < array_ants[j].array_vertexes.length - 1; k++)
{
pheromone[array_ants[j].array_vertexes[k]][array_ants[j].array_vertexes[k + 1]]
+= Q / array_ants[j].length;
        pheromone[array_ants[j].array_vertexes[k +
1]][array_ants[j].array_vertexes[k]] =
pheromone[array_ants[j].array_vertexes[k]][array_ants[j].array_vertexes[k + 1]];
    }
    array_ants[j].length +=
distance[array_ants[j].array_vertexes[array_ants[j].array_vertexes.length -
1]][vertex];

array_ants[j].array_vertexes[array_ants[j].array_vertexes.length] = vertex;

    if(array_ants[j].length < main_path.length || main_path.length <
0) {
        main_path.array_vertexes = array_ants[j].array_vertexes;
        main_path.length = array_ants[j].length;
    }
    array_ants[j].length = array_vertexs_weight[vertex];
    array_ants[j].array_vertexes = [];
    array_ants[j].array_vertexes[0] = vertex;
}
for(let j = 0; j < pheromone.length; j++) {
    for(let k = 0; k < pheromone.length; k++) {
        if (j != k) {
            pheromone[j][k] *= (1 - RO);
        }
    }
}

```

```

    }
  }

  if(main_path.length + val_start_time > val_end_time &&
main_path.array_vertexes.length > 2) {
    main_path.length -=
    distance[main_path.array_vertexes[main_path.array_vertexes.length -
2]][main_path.array_vertexes[main_path.array_vertexes.length - 1]];
    main_path.length -=
    distance[main_path.array_vertexes[main_path.array_vertexes.length -
3]][main_path.array_vertexes[main_path.array_vertexes.length - 2]];
    main_path.length +=
    distance[main_path.array_vertexes[main_path.array_vertexes.length -
3]][main_path.array_vertexes[main_path.array_vertexes.length - 1]];
    main_path.length -=
    array_vertexes_weight[main_path.array_vertexes.length - 2];
    main_path.array_vertexes.splice(main_path.array_vertexes.length - 2,
1);
  }

  setTimeout(() => {
    console.log(main_path);
    build_route_on_map(main_path, myMap);
  }, 5000);
}

var timer_real_time;
real_start_time.addEventListener('click', function (){
  window.clearInterval(timer_real_time);
  build_route(currentDay, myMap);
  if(real_start_time.checked == true) {
    timer_real_time = window.setInterval(build_route,300000, currentDay,
myMap);
  }
});

var distance;
var coordinates_places = [],
    curDay_places = [],
    val_start_time,
    val_end_time;
var array_vertexes_weight;
// функция генерации матрицы связей
function build_route(currentDay, myMap) {
  // var distance,
  //     array_vertexes_weight;
  array_vertexes_weight = [];
  var check = 0;

```

```
if(real_location.checked == true) {
    check = 1;
}

for(let i = 0; i < list_times.length; i++) {
    if(end_time.value == list_times[i]['time']) {
        val_end_time = list_times[i]['value'];
    }
}

if(real_start_time.checked == true) {
    var date = new Date();
    hours = date.getHours();
    minutes = date.getMinutes();
    val_start_time = hours*60 + minutes;
    if(val_end_time - val_start_time < 60) {
        val_start_time = val_end_time - 60;
    }
    console.log(val_start_time);
    console.log(val_end_time);
}
else {
    for(let i = 0; i < list_times.length; i++) {
        if(start_time.value == list_times[i]['time']) {
            val_start_time = list_times[i]['value'];
        }
    }
    console.log(val_start_time);
    console.log(val_end_time);
}

if(val_end_time - val_start_time >= 60) {
    if(currentDay.length + check > 1) {
        coordinates_places = [];
        curDay_places = [];
        var val_first_day;

        switch(first_day.value) {
            case 'Понедельник':
                val_first_day = 0;
                break;
            case 'Вторник':
                val_first_day = 1;
                break;
            case 'Среда':
                val_first_day = 2;
                break;
            case 'Четверг':
```

```

        val_first_day = 3;
        break;
    case 'Пятница':
        val_first_day = 4;
        break;
    case 'Суббота':
        val_first_day = 5;
        break;
    case 'Воскресенье':

        val_first_day = 6;
        break;
    }

    if(real_location.checked == true) {
        coordinates_places[0] = cur_location.join();
        array_vertexs_weight[array_vertexs_weight.length] = 0;
    }

    for(let i = 0; i < currentDay.length; i++) {
        coordinates_places[coordinates_places.length] =
currentDay[i]['location_place_coordinates'];
        if(currentDay[i]['working_week_schedule'] == null) {
            curDay_places[curDay_places.length] = null;
        }
        else {
            if(currentDay[i]['working_week_schedule'][(val_first_day
+ num_select_day) % 7]['type'] == 0) {
                curDay_places[curDay_places.length] = null;
            }
            else {
                var start, close;
                for(let j = 0; j < list_times.length; j++) {

if(currentDay[i]['working_week_schedule'][(val_first_day + num_select_day) %
7]['opening_time'] == list_times[j]['time']) {
                    start = list_times[j]['value'];
                }

if(currentDay[i]['working_week_schedule'][(val_first_day + num_select_day) %
7]['closing_time'] == list_times[j]['time']) {
                    close = list_times[j]['value'];
                }
            }

            curDay_places[curDay_places.length] = {start: start,
close: close};

```



```

    }
  }
}
console.log(curDay_places);
var distance0 = [];
for(let i = 0; i < coordinates_places.length; i++) {
  distance0[i] = [];
  for(let j = 0; j < coordinates_places.length; j++) {
    if(i != j) {

      let route = new ymaps.multiRouter.MultiRoute({
        referencePoints: [
          coordinates_places[i].split(','),
          coordinates_places[j].split(',')
        ],
        params: {
          routingMode: type_transport,
          results: 1
        }
      });
      route.model.events.add('requestsuccess', function ()
{
        var activeRoute = route.getActiveRoute();
        if (activeRoute) {
          distance0[i][j] =
Math.ceil(route.getActiveRoute().properties.get("duration").value / 60);
        }
      });
    }
    else {
      distance0[i][j] = 0;
    }
  }
  if(i == coordinates_places.length - 1) {
    setTimeout(() => { console.log("Ждем немного");
setDistance(distance0, myMap); }, 1500);
  }
}
}
else {
  myRoute && myMap.geoObjects.remove(myRoute);
}
}
else {
  alert('Разница между временем начала и конца должна быть не менее
одного часа. ');
  myRoute && myMap.geoObjects.remove(myRoute);
}
}

```

```

    }

function setDistance(distance0, myMap) {
    let check_undefined = true;

    if(check_undefined == true) {
        distance = distance0;
        for(let i = 0; i < currentDay.length; i++) {
            if(currentDay[i]['working_week_schedule'] == null) {
                array_vertexs_weight[array_vertexs_weight.length] = 30;
            }
            else {
                array_vertexs_weight[array_vertexs_weight.length] = 60;
            }
        }
        if(real_location.checked == false) {
            let check = 0;
            for(let i = 0; i < currentDay.length; i++) {
                if(curDay_places[i] == null) {
                    SearchRouteAntColony(distance, i, myMap);
                    console.log('start find');
                    check = 1;
                    break;
                }
                else if(val_start_time >= curDay_places[i].start &&
val_start_time < curDay_places[i].close) {
                    SearchRouteAntColony(distance, i, myMap);
                    console.log('start find');
                    check = 1;
                    break;
                }
            }
            if(check == 0) {
                alert("К сожалению, ни одну достопримечательность из
выбранных нельзя посетить в указанное ограничение по времени.");
                myRoute && myMap.geoObjects.remove(myRoute);
            }
        }
        else {
            SearchRouteAntColony(distance, 0, myMap);
        }
    }
}

function build_route_on_map(main_path, myMap) {
    if(main_path.array_vertexes.length < 2 && real_location.checked == false
|| main_path.array_vertexes.length < 3 && real_location.checked == true) {

```

```

    alert("К сожалению, ни одну достопримечательность из выбранных
    нельзя посетить в указанное ограничение по времени.");
    myRoute && myMap.geoObjects.remove(myRoute);
  }
  else {
    let nextDay = [];
    for(let i = 0; i < currentDay.length; i++) {
      let check = 0;
      for(let j = 0; j < main_path.array_vertexes.length; j++) {
        if(i == main_path.array_vertexes[j]) {
          check = 1;
        }
      }
      if(check == 0) {
        nextDay[nextDay.length] = currentDay[i];
        var elem = document.getElementById('place_' +
currentDay[i]['id_place']);
        elem.remove();
        addPlacemark(currentDay[i]['id_place']);
        currentDay.splice(i, 1);
      }
    }

    if(nextDay.length > 0) {
      if(listDays[num_select_day + 1] == undefined) {
        listDays[num_select_day + 1] = nextDay;
        deleteDay.style.opacity = "1";
        rightDay.style.opacity = "1";
      }
      else {
        for(let j = 0; j < nextDay.length; j++) {
          let check = 0;
          for(let i = 0; i < listDays[num_select_day + 1].length;
i++) {
            if(listDays[num_select_day + 1][i] == nextDay[j]) {
              check = 1;
            }
          }
          if(check == 0) {
            listDays[num_select_day + 1][listDays[num_select_day
+ 1].length] = nextDay[j];
          }
        }
      }
    }

    console.log(currentDay);
    console.log(nextDay);
  }
}

```

```

let array_vertex = [];
for(let i = 0; i < main_path.array_vertexes.length; i++) {
  array_vertex[i] =
coordinates_places[main_path.array_vertexes[i]];
}

myRoute && myMap.geoObjects.remove(myRoute);
myRoute = new ymaps.multiRouter.MultiRoute({
  referencePoints:
    array_vertex,
  params: {
    routingMode: type_transport,
    results: 1
  }
});
myMap.geoObjects.add(myRoute);

var time_route = document.getElementById('time_route'),
    hours = Math.floor(main_path.length / 60),
    minuts = main_path.length % 60,
    str_hour,
    str_min;
if(main_path.length != NaN || main_path.length != Infinity ||
main_path.length != undefined) {
  if(hours % 100 <= 20 && hours % 100 >= 10) {
    str_hour = `${hours} часов`;
  }
  else if(hours % 10 == 1) {
    str_hour = `${hours} час`;
  }
  else if(hours % 10 <= 4 && hours % 10 >= 2) {
    str_hour = `${hours} часа`;
  }
  else if(hours % 10 <= 9 && hours % 10 >= 5) {
    str_hour = `${hours} часов`;
  }
  else if(hours % 10 == 0) {
    if(hours == 0) {
      str_hour = '';
    }
    else {
      str_hour = `${hours} часов`;
    }
  }
}

if(minuts % 100 <= 20 && minuts % 100 >= 10) {
  str_min = `${minuts} минут`;
}

```

```

        else if(minuts % 10 == 1) {
            str_min = `${minuts} минута`;
        }
        else if(minuts % 10 <= 4 && minuts % 10 >= 2) {
            str_min = `${minuts} минуты`;
        }
        else if(minuts % 10 <= 9 && minuts % 10 >= 5) {
            str_min = `${minuts} минут`;
        }
        else if(minuts % 10 == 0) {
            if(minuts == 0) {
                str_min = '';
            }
            else {
                str_min = `${minuts} минут`;
            }
        }
    }
}
else {
    str_hour = '';
    str_min = 'невывчисляемо';
}

time_route.innerHTML = `Примерное время путешествия с учетом
посещения достопримечательностей: ${str_hour} ${str_min}`;

var price_route = document.getElementById('price_route'),
    price = 0,
    str_price;

for(let i = 0; i < currentDay.length; i++) {
    price += +currentDay[i]['ticket_price'];
    console.log(currentDay[i]['ticket_price']);
}

if(price % 100 <= 20 && price % 100 >= 10) {
    str_price = `${price} рублей`;
}
else if(price % 10 == 1) {
    str_price = `${price} рубль`;
}
else if(price % 10 <= 4 && price % 10 >= 2) {
    str_price = `${price} рубля`;
}
else if(price % 10 <= 9 && price % 10 >= 5) {
    str_price = `${price} рублей`;
}
else if(price % 10 == 0) {

```

```

        if(price == 0) {
            str_price = 'бесплатно';
        }
        else {
            str_price = `${price} рублей`;
        }
    }

    price_route.innerHTML = `Примерная стоимость путешествия без учета
цен на поездку: ${str_price}.`
}

var form = document.getElementById("saveRoute");

form.addEventListener("submit", function (e) {
    e.preventDefault();
    let settings = new Settings();
    settings.first_day = document.getElementById('first_day').value;
    settings.start_time = document.getElementById('start_time').value;
    settings.real_location =
document.getElementById('real_start_time').checked;
    settings.end_time = document.getElementById('end_time').value;
    settings.mode_travel = document.getElementById('mode_travel').value;
    settings.real_location =
document.getElementById('real_location').checked;
    settings.name_route = document.getElementById('name_route').value;
    settings.name_city = document.getElementById('city').value;
    let settings_json = JSON.stringify(settings);
    $.ajax({
        type: "POST",
        url: "/quest.php",
        data: {data:settings_json, url:'save_settings.json'},
        success: function () {
            let listDays_json = JSON.stringify(listDays);
            $.ajax({
                type: "POST",
                url: "/quest.php",
                data: {data:listDays_json, url:'save_listdays.json'},
                success: function() {

document.location.replace("https://personalguide.com/route_builder/save_route");
                }
            });
        }
    });
});

function set_cur_location() {

```

```
        ymaps.geolocation.get({provider: "browser"}).then(function (res) {
            cur_location = res.geoObjects.get(0).geometry.getCoordinates();
            setTimeout(() => { build_route(currentDay, myMap); }, 100);
        }, function (e) {
            })
        })
    }
}

class Path {
    constructor() {
        var array_vertexes,
            length,
            array_unavailable_vertexes;
    }
}

class Settings {
    constructor() {
        var first_day,
            start_time,
            real_location,
            end_time,
            mode_travel,
            real_location,
            name_route,
            name_city;
    }
}
```