МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования   
«САНКТ–ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой №\_43\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Д.т.н., профессор |  |  |  | М.Ю.Охтилев |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

БАКАЛАВРСКАЯ РАБОТА

|  |  |
| --- | --- |
| на тему | Разработка мобильного приложения для поддержки внутреннего туризма |
|  | |
|  | |

|  |  |
| --- | --- |
| выполнена | Чепусовым Владиславом Константиновичем |
| фамилия, имя, отчество студента в творительном падеже | |

|  |  |  |  |
| --- | --- | --- | --- |
| по направлению подготовки | 02.03.03 |  | Математическое обеспечение |
|  | код |  | наименование направления |
| и администрирование информационных систем | | | |
| наименование направления | | | |
| направленности | 01 |  | Администрирование |
|  | код |  | наименование направленности |
| информационных систем | | | |
| наименование направленности | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент группы № | 4736 |  |  |  | В.К.Чепусов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Руководитель

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Канд. техн. наук, доцент |  |  |  | С.В.Щекин |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования   
«САНКТ–ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

УТВЕРЖДАЮ

Заведующий кафедрой №\_43

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Д.т.н., профессор | |  | |  | |  | | М.Ю.Охтилев | |
| должность, уч. степень, звание | |  | | подпись, дата | |  | | инициалы, фамилия |

ЗАДАНИЕ НА ВЫПОЛНЕНИЕ БАКАЛАВРСКОЙ РАБОТЫ

|  |  |  |  |
| --- | --- | --- | --- |
| студенту группы | 4736 |  | Чепусову Владиславу Константиновичу |
|  | номер |  | фамилия, имя, отчество |

|  |  |
| --- | --- |
| на тему | Разработка мобильного приложения для поддержки внутреннего туризма |
|  | |
|  | |

|  |  |  |  |
| --- | --- | --- | --- |
| утвержденную приказом ГУАП от | 09.04.2021 | № | 12-56/21 |

|  |  |
| --- | --- |
| Цель работы: | Проектирование и разработка мобильного приложения для поддержки |
| внутреннего туризма | |
|  | |

|  |  |
| --- | --- |
| Задачи, подлежащие решению: | Анализ предметной области, выбор средств разработки |
| создание работоспособного мобильного приложения | |
|  | |

|  |  |
| --- | --- |
| Содержание работы (основные разделы): | Анализ предметной области; выбор средств |
| разработки; проектирование и разработка системы | |
|  | |
|  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Срок сдачи работы « | 31 | » | мая | 20 | 21 |

Руководитель

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Канд. техн. наук, доцент |  |  |  | С.В.Щекин |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

Задание принял(а) к исполнению

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| студент группы № | 4736 |  | 12.04.2021 |  | В.К.Чепусов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

# 

# СОДЕРЖАНИЕ

[ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ 4](#_Toc74258558)

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 5](#_Toc74258559)

[ВВЕДЕНИЕ 6](#_Toc74258560)

[1 Анализ предметной области 9](#_Toc74258561)

[1.1 Описание предметной области 9](#_Toc74258562)

[1.2 Обзор существующих аналогов системы 12](#_Toc74258563)

[Выводы к разделу 21](#_Toc74258564)

[2 Выбор средств разработки 22](#_Toc74258565)

[Выводы к разделу 26](#_Toc74258566)

[3 Проектирование и разработка системы 27](#_Toc74258567)

[3.1 Структурная модель системы и общее представление требований 27](#_Toc74258568)

[3.2 Классы и характеристики пользователей 28](#_Toc74258569)

[3.3 Системные функции 29](#_Toc74258570)

[3.3.1 Функциональные требования 29](#_Toc74258571)

[3.3.2 Нефункциональные требования 30](#_Toc74258572)

[3.4 Ограничения проекта 31](#_Toc74258573)

[3.5 Варианты использования системы 31](#_Toc74258574)

[3.6 Проектирование базы данных 45](#_Toc74258575)

[3.7 Интерфейсы пользователя 49](#_Toc74258576)

[3.8 Реализация требований к безопасности 52](#_Toc74258577)

[Выводы к разделу 53](#_Toc74258578)

[ЗАКЛЮЧЕНИЕ 54](#_Toc74258579)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 56](#_Toc74258580)

[ПРИЛОЖЕНИЕ А 58](#_Toc74258581)

[ПРИЛОЖЕНИЕ Б 64](#_Toc74258582)

# ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Lazy Guide — название мобильного приложения для поддержки внутреннего туризма разрабатываемого в текущей работе.

# Аудиогид — аудиофайл, используемый для самостоятельного ознакомления с экспозицией выставки, музея, а также иных достопримечательностей.

# Геометка — объект класса «LatLng», содержащий географические координаты какого-то места

Маршрут — множество объектов класса «LatLng», необходимых для построения пути от расположения местонахождения пользователя до достопримечательности

Местоположение/местонахождение — значение, содержащее географические координаты расположения пользователя на карте.

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

ERD — Диаграмма отношений сущностей

IDE — Интегрированная среда разработки

JS — JavaScript

OC — операционная система

TS — TypeScript

UI — пользовательский интерфейс (user interface)

БД — база данных

ВИ — вариант использование

ПО — программное обеспечение

Система — совокупность разрабатываемого мобильного приложения, его клиентской части, БД, а также сервера

СУБД — система управления базами данных

# ВВЕДЕНИЕ

Выпускная квалификационная работа посвящена проектированию и разработке мобильного приложения для поддержки внутреннего туризма, путём систематизированного хранения данных о достопримечательностях регионов Российской Федерации и предоставления этой информации конечному пользователю

**Актуальность работы.** В нынешнее время смартфоны стали занимать одно из центральных положений в жизни каждого индивида, перейдя рубеж от дорогостоящего средства связи, которое могли себе позволить лишь немногочисленные состоятельные люди к жизненно необходимому девайсу. Он объединил в себе бесчисленное множество устройств, таких как плееры, мобильные телефоны, фонарики, карты, а также некоторые аспекты персональных компьютеров, выводя возможности коммуникации человека на совершенно новый уровень. С помощью смартфонов теперь у каждого есть право осуществлять выход во всемирную сеть, в том числе на улице или в пути, чтобы находить интересующую человека информацию, осуществлять онлайн-звонки, прокладывать маршруты и пользоваться общедоступным контентом, произведенным людьми и иногда нейронными сетями, такими как музыка, видео и книги. Благодаря умным телефонам у людей появилось больше свободы относительно путешествий и передвижений.

Совсем недавно произошла вспышка коронавирусной инфекции, которая изменила быт многих людей на крайне продолжительное время. Масштабы пандемии потрясают своими размерами. Из-за неё в ужасе были закрыты границы государств, все граждане попали в строгие рамки самоизоляции и даже спустя многие месяцы у них нет возможности посетить другие страны. Всё это вкупе с продолжительным нахождением людей в своих квартирах привело к росту заинтересованности граждан посещать и исследовать локальные достопримечательности своих и близлежащих регионов.

Россия великая страна с огромной территорией, наше достояние потрясает своими размерами, однако именно из-за данного размаха некоторые культурные, исторические и природные объекты, заслуживающие внимания, не получают достойной огласки. Не в каждом регионе Российской Федерации уделяется заинтересованность в развитии туристической сферы. Это приводит к тому, что граждане самостоятельно стараются искать что-то прекрасное и ценное на своей малой родине. Однако самым распространенным способом поиском достопримечательностей, на сегодняшний день, является посещение сайтов, на которых зачастую дублируется одна и та же информация. Причем на таких ресурсах обычно публикуются самые распространённые и популярные для всех локации, а что насчет не пользующихся известностью мест, то таких записей мало, а в большинстве случаев и вовсе не наблюдается.

Нахождение мест для душевного обогащения через поиск сайтов в интернете утомительное дело гораздо привлекательней выглядит возможность иметь под рукой встроенный альманах со значительной базой данных. Наличие приложения на смартфоне может повысить осведомленность людей о культурном богатстве собственного региона, так как просмотр информации не будет занимать большого количества времени и сил, в меру своей портативности, интуитивно-понятного интерфейса и того факта, что смартфоны стали неотъемлемой частью современного человека.

**Целью дипломной работы** является проектирование и разработка мобильного приложения для поддержки внутреннего туризма.

Для достижения поставленной в дипломной работе цели решаются следующие задачи:

1. Изучение и анализ предметной области;
2. Освоение технологий и средств для разработки мобильного приложения;
3. Разработка UI;
4. Проектирование и создание базы данных;
5. Реализация управляющей логики клиентской и серверной частей.

**Структура и объём работы.** Выпускная квалификационная работа включает в себя введение, 3 раздела, заключение, список использованных источников из 15 наименований и 2 приложений. Основная часть работы изложена на 56 страницах, содержит в себе 16 изображений и 11 таблиц.

В первом разделе производится исследование предметной области, выбор программных инструментов для процесса разработки, а также обзор схожих систем. Во втором разделе подробно разбираются вопросы проектирования и разработки системы.

# ****Анализ предметной области****

# Описание предметной области

Мобильное приложение для поддержки внутреннего туризма разрабатывается для опытных пользователей мобильных устройств (планшетов, смартфонов), желающих узнать больше или посетить самостоятельно ряд достопримечательностей, а также для людей, которые стараются разрекламировать собственные малоизвестные туристические объекты.

При процессе анализа предметной области необходимо выявить основные объекты, их характеристики, определить круг пользователей системы и составить их потребности, установление данных знаний производилось на базисе принципов изложенных в [3].

Основные объекты:

* Регион
* Достопримечательность
* Тип достопримечательности
* Галерея

Для данных объектов системы определены присущие им атрибуты или характеристики:

Регион:

* + 1. Идентификатор (равный номеру региона)
    2. Наименование региона

Достопримечательность:

* + 1. Идентификатор достопримечательности
    2. Идентификатор региона
    3. Название достопримечательности
    4. Адрес
    5. Описание
    6. Стоимость посещения
    7. Веб-сайт
    8. Контактные данные
    9. Время работы
    10. Лицевое изображение

Тип достопримечательности:

* + 1. Идентификатор типа
    2. Название

Галерея:

* + 1. Идентификатор изображения
    2. Идентификатор достопримечательности
    3. Название
    4. Фото

Описание процесса взаимодействия пользователя с приложением:

1. Пользователь, вошедший в приложение, может:
2. Выбрать интересующийся регион
3. Использовать поиск по регионам
4. Перейти в раздел «Свяжитесь с нами», чтобы предложить собственную достопримечательность
5. Пользователь, выбравший регион, может:
6. Вернуться назад к выбору регионов
7. Выбрать достопримечательность из списка
8. Отсортировать список имеющихся достопримечательностей по их типу

1. Пользователь, перешедший в раздел «Свяжитесь с нами», может:
2. Вернуться назад к выбору регионов
3. Заполнить необходимые поля и отправить письменный запрос на почту «lazyguide2021@gmail.com»
4. Пользователь, выбравший достопримечательность, может:
5. Вернуться назад к списку достопримечательностей региона
6. Просмотреть информацию о достопримечательности
7. Нажать на карту, чтобы перейти к полномасштабной карте
8. Просмотреть галерею изображений
9. Пользователь, перешедший к полномасштабной карте, может:
10. Вернуться назад к информации о достопримечательности
11. Дать разрешение на определение собственного местоположения и построить маршрут от него до достопримечательности

# Обзор существующих аналогов системы

В процессе анализа предметной области, были найдены ближайшие к разрабатываемому приложению аналоги.

**Петербург 24 -** мобильный авторский гид – Это приложение для пользователей мобильных устройств на OС iOS и Android, которое объединило путеводитель по Ленинградской области и афишу событий.

Ключевым элементом информации, используемым в данной программе, являются записи о достопримечательностях. Регистрация в приложении отсутствует, что является значительным плюсом – для ознакомления с продуктом не требуется предоставление данных о себе. Меню выполнено в виде выдвигающейся шторки со следующими пунктами: Главная, Места рядом, Избранное, Достопримечательности, Обратная связь и т.д.

Главная – открывает форму, из которой можно перейти в остальные разделы.

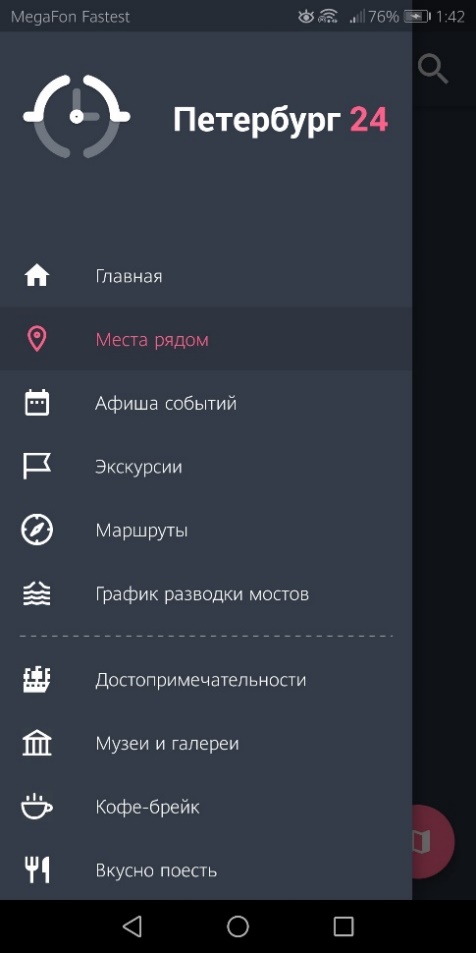
Места рядом – осуществляет поиск близлежащий достопримечательностей, однако при моем запросе выдало сообщение о том, что записей не найдено (No data), что в свою очередь можно отнести к минусам приложения.

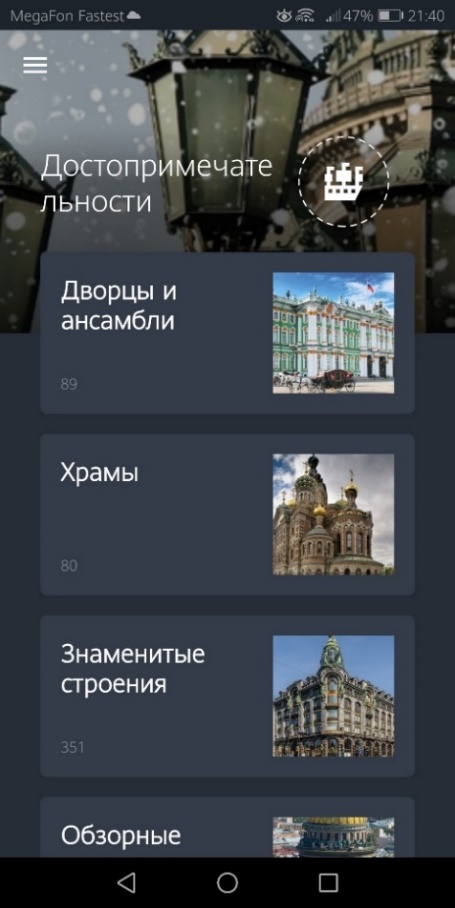
Избранное – отображает список достопримечательностей, которые пользователь отметил сердечком. Но при выходе из приложения данные не сохраняются.

Достопримечательности – перенаправляет на форму, где в виде списка представлены все культурные места Санкт-Петербурга. При выборе определённого открывается форма с соответствующей информацией.

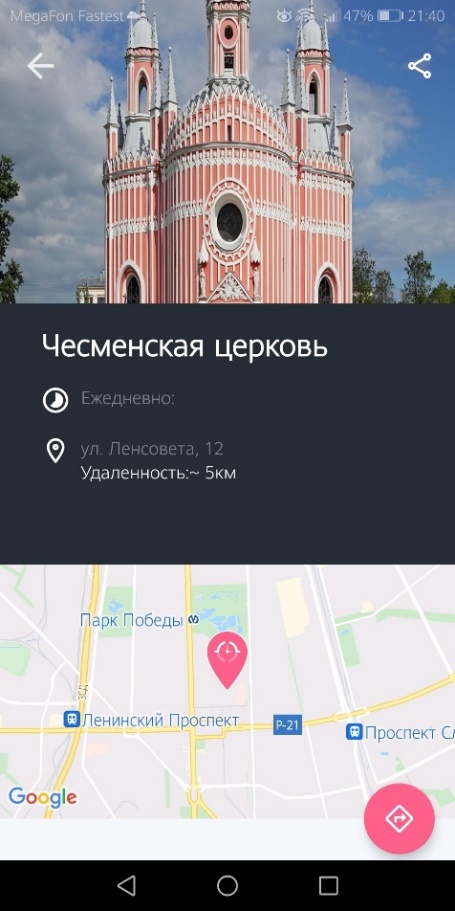
Обратная связь – перенаправляет на форму, где имеется возможность связаться с разработчиками посредством написания электронного письма прямо из приложения.

Самым значительным минусом данного приложения можно считать исключительно локальную сферу деятельности. Данные имеются только о Санкт-Петербурге и близлежащих городах, но несмотря на это в базе данных отсутствует множество культурно значимых мест данного региона. Также некорректно работает функция поиска достопримечательностей поблизости, в моем случае выдало ошибку.

  
Рисунок 1 – Внешний вид приложения Петербург 24. Меню.

  
Рисунок 2 – Внешний вид приложения Петербург 24.

Раздел «достопримечательности».

  
Рисунок 3 – Внешний вид приложения Петербург 24.

Раздел «достопримечательность».

  
Рисунок 4 – Внешний вид приложения Петербург 24.

Раздел «Места рядом»

**izi.TRAVEL** — приложение является сборником аудио-экскурсий для пользователей смартфонов на ОC iOS и Android. Ключевой информацией данного приложения является список достопримечательностей и аудиогиды по ним. Регистрация в приложении не является обязательной, однако присутствует платная подписка, которая расширяет список доступных возможностей, а также отключает рекламу в самом программном продукте.

Меню выполнено в виде открывающегося списка с несколькими пунктами: Мои истории, Покупки, Загруженное, Закладки, История, Обратная связь.

Мои истории – позволяет создать собственную экскурсию, добавлять в нее пункты остановки, с названием достопримечательностей и их описанием, но для использования необходимо предоставить личную информацию.

Покупки – позволяет приобретать платные туры, в которых вам будет зачитывать виртуальный гид информацию о доступной достопримечательности.

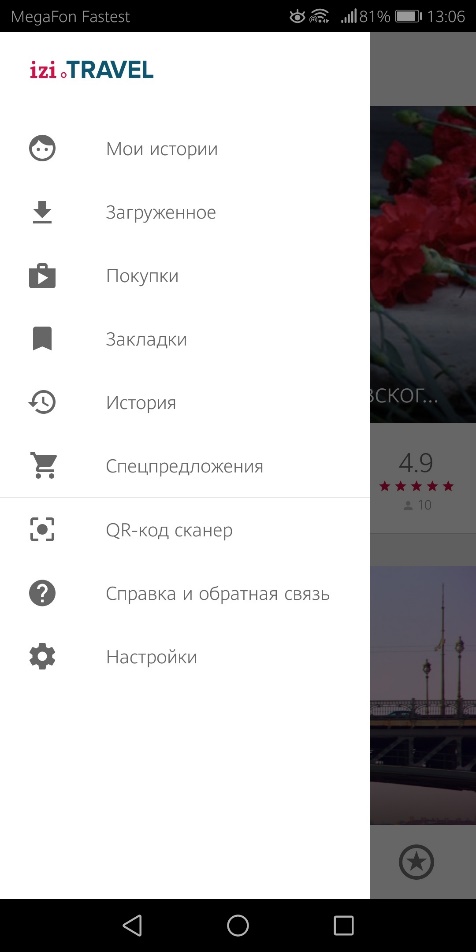
Загруженное – позволяет сохранять информацию о выбранной достопримечательности на устройство пользователя и просмотра этих данных оффлайн.

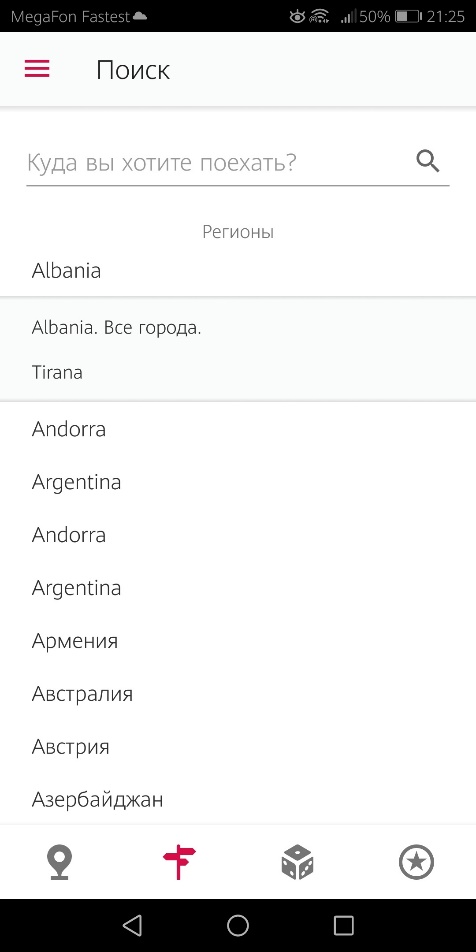
Закладки –достопримечательности, которые пользователь решил отметить в приложении.

История – ранее запущенные пользователем аудиогиды.

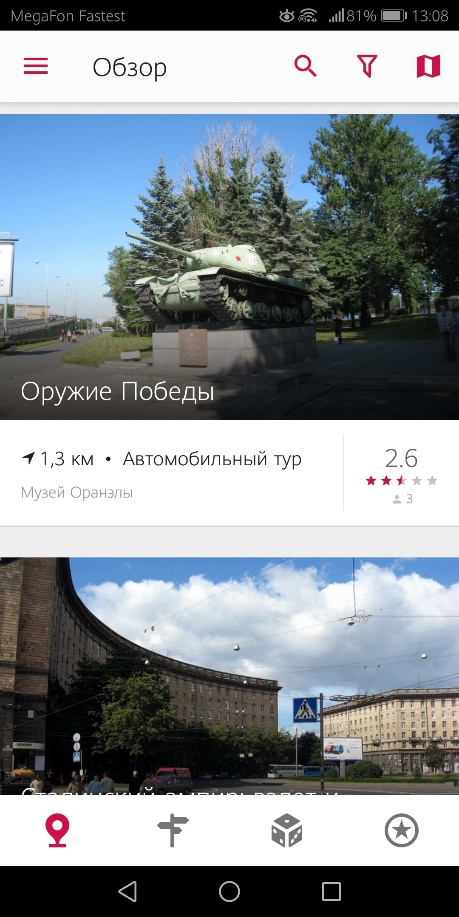
Особенностью данного информационного продукта является наличие большой базы данных с экскурсиями, которые зачитываются профессиональными дикторами. Соответственно, для конечного пользователя необходимо выбрать интересующую его достопримечательность или тур и приступить к прослушиванию информации, постепенно продвигаясь по отмеченным на карте пунктам. При достижении пользователем очередной точки включается следующая глава, которая относится именно к данной локации.

Главными минусами данного приложения являются наличие рекламы, которая мешает пользователю погрузиться с головой в историю, постоянно отвлекая от прослушивания. Также наблюдается снижение качества и количества информации при выборе пользователем регионов со слаборазвитой туристической сферой, большая часть городов России недоступна в данном приложении.

  
Рисунок 5 – Внешний вид приложения «izi.TRAVEL». Раздел «Меню».

  
Рисунок 6 – Внешний вид приложения «izi.TRAVEL».

Раздел «Выбора региона».

  
Рисунок 7 – Внешний вид приложения «izi.TRAVEL».

Раздел «Выбора достопримечательности».

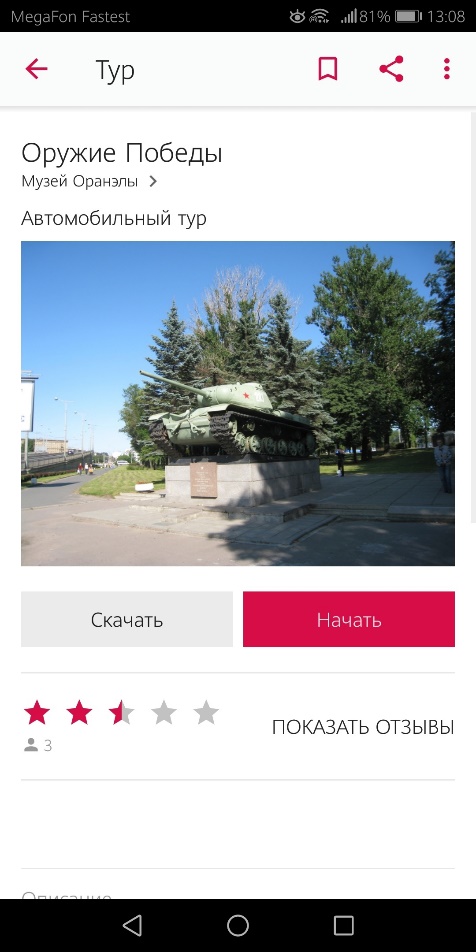


Рисунок 8 – Внешний вид приложения «izi.TRAVEL». Раздел «Тур».

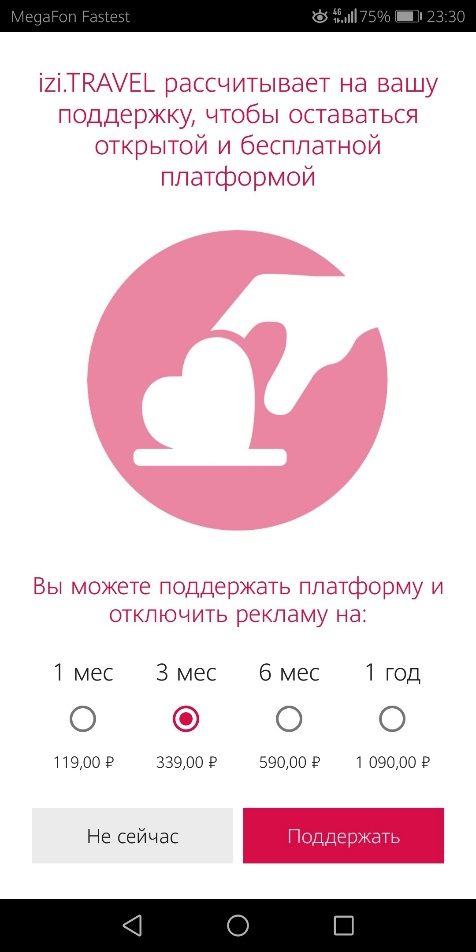


Рисунок 9 – Внешний вид приложения «izi.TRAVEL».

Раздел «Отключение рекламы»

Процесс сравнения аналогов и разрабатываемого программного продукта представлен в таблице 1.

Таблица 1. Сравнение приложений

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | izi.TRAVEL | Петербург 24 | Lazy Guide |
| Наличие рекламы | Да | Нет | Нет |
| Территориальность | Большая часть земного шара, но присутствует информация только о значительных субъектах | Санкт-Петербург и Ленинградская область | Российская Федерация |
| Платные услуги | Да | Да | Нет |
| Возможность построения маршрута | Да | Да | Да |

В процессе изучения этих программных продуктов, я пришел к выводу, что ни один из них не имеет развитую базу данных достопримечательностей Российской Федерации. Данные путеводители не предоставляют достаточный функционал для свободного изучения регионов нашей страны и поддержки внутреннего туризма.

# Выводы к разделу

В настоящим разделе была проанализирована предметная область, а также установлены основные объекты и характеристики разрабатываемого программного продукта. Помимо этого, определенны основные моменты взаимодействия пользователя с приложением, что позволяет в дальнейшем установить функциональные требования.

Был произведен сравнительный анализ и рассмотрение аналогов, в результате которых выявлено отсутствие прямых эквивалентов разрабатываемого мобильного приложения для поддержки внутреннего туризма. При изучении программных продуктов, схожих по тематике, были установлены их сильные и слабые стороны, такие как, систематизированность данных о достопримечательностях, объём и территориальность информации, а также наличие рекламы, ошибок, багов и т.д.

# Выбор средств разработки

При изучении и анализе возможных инструментов для разработки мобильных информационных продуктов, был сделан выбор в пользу следующих средств разработки, языков программирования и платформ:

IDE **—** Android Studio (версия 4.1.3).

Android Studio — является интегрированной средой разработки,

обеспечивающей средствами для написания программного продукта для любого типа устройств с операционной системой Android [6]. Главным преимуществом, выделяющим ее на фоне остальных IDE, является наличие встроенного эмулятора различных девайсов (смартфоны, планшеты и т.д.). Это является необходимым при тестировании написанного приложения. Наличие редактора кода, поддержка языков программирования Java и Kotlin, система сборки, базирующаяся на Gradle, которая позволяет настраивать её для разных устройств, все это делает Android Studio лучшим выбором. Также данная среда разработки считается официальной для Android [7].

Язык программирования для клиентской части– Kotlin (версия 1.5.0)

Kotlin — является статически типизированным, объектно-

ориентированным языкoм программирования, появившимся в 2011 году под авторством международной компании JetBrains. Перед создателями стояла задача создать язык более лаконичный и типобезопасный, нежели Java [5;8].

Языкoв для мoбильной разработки существует большое множество, но

Kotlin и Java — одни из самых популярных [15], поэтому в работе стоял выбор именно между данными языками для этого было произведено сравнение основных отличий, представленное в таблице 2.

Таблица 2. Сравнительный анализ Java и Kotlin

|  |  |  |
| --- | --- | --- |
| Характеристика | Java | Kotlin |
| Простота изучения | Крайне тяжелый язык с большим количеством нюансов и конструкций [15] | Более дружелюбный к новым пользователям язык. Имеет упрощенные конструкции нежели Java [15] |
| Поддерживается в Android Studio? | Да | Да |
| Динамическая типизация | Нет | Да |
| Объём доступной информации (документация, форумы с вопросами) | В меру того, что Java является официальным и самым популярным языком программирования для мобильной разработки, возникло огромное сообщество разработчиков, выкладывающих свою наработки на StackOverflow и GitHub. Также на официальном сайте разработчика имеется развернутая документация, огромное количество примеров кодов на различных сайтах | Kotlin является официальным и последним введенным языком Android [8]. Его активно продвигает Google как язык, который в будущем заменит Java. Из-за этого некогда малое сообщество разработчиков на Kotlin растет с каждым днем [15]. Повышается количество документации на официальном сайте и примеры на различных ресурсах |

СУБД – Microsoft SQL Server (версия 18.9.1)

Microsoft SQL Server - система управления реляционными БД, была создана транснациональной компаний Microsoft [9]. Данная платформа реализована для решения критически важных задач, обладающее высокой доступностью, повышенной производительностью и безопасностью.Данная СУБД была выбрана, потому что представляет собой хорошо масштабируемый, полностью реляционный, быстродействующий сервер, способный обрабатывать значительные объемы данных для приложений с клиент-серверной архитектурой [13].

Платформа веб-сервера – Node.js(Версия 16.2.0)

Node.js -среда выполнения JavaScript, построенная на движке Chrome V8 JavaScript [10].Позволяет разработчикам использовать JavaScript для создания серверного кода, имеет большое и активное сообщество разработчиков, открытый исходный код, также прост в использовании и может обрабатывать параллельные запросы [1]. Node.js имеет плагины для СУБД Microsoft SQL Server, выбрал на которую пал в работе.

Язык программирования веб-сервера— TypeScript (Версия 4.3.1)

TypeScript— объектно-ориентированный язык программирования, представляющий из себя средство для разработки веб-приложений, расширяющий и дополняющий функции JavaScript [2;11]. В процессе реализации программного продукта возник вопрос, какой из языком использовать с серверной платформой Node.js

Выбор производился из JavaScript и TypeScript. Для данного решения было произведен анализ ключевых различий этих языков, представленный в таблице 3.

Таблица 3. Сравнение JavaScript и TypeScript

|  |  |  |
| --- | --- | --- |
| Характеристика | JavaScript | TypeScript |
| Поддерживается в IDE Visual Studio | Да | Да |
| Поддержка сторонних модулей | Нет | Да |
| Поддержка статической типизации | Не поддерживает | TypeScript имеет статическую типизацию(опционально), которая предоставляет проверку корректности типа в процессе компиляции [14] |
| Поддержание функции необязательного параметра | Нет | Да |
| Обратная совместимость языков | Нет | TypeScript имеет обратную совместимость с JavaScript, поэтому практически любой код на JS будет выполнен и в TS [14] |

Исходя из всего вышесказанного, можно заявить, что TypeScript обладает множеством преимуществ JavaScript, а также расширяет его возможности.

# Выводы к разделу

В настоящем разделе был произведен выбор средств разработки, который базировался на актуальности и известности возможных решений, доступном и легком освоение знаний, а также комфортном использовании всего вышесказанного.

Осуществлен сравнительный анализ с альтернативными программными инструментами, в результате которых были выбраны наилучшие средства для разрабатываемого программного продукта.

# Проектирование и разработка системы

# Структурная модель системы и общее представление требований

При анализе и изучении предметной области были установлены ключевые объекты мобильного приложения для поддержки внутреннего туризма. На основании полученных данных была осуществлённая разработка описаний требований к программному продукту и его структуре.

Модель структуры системы представлена контекстной диаграммой (Рис 10), на которой показано работа пользователей с элементами программного продукта.

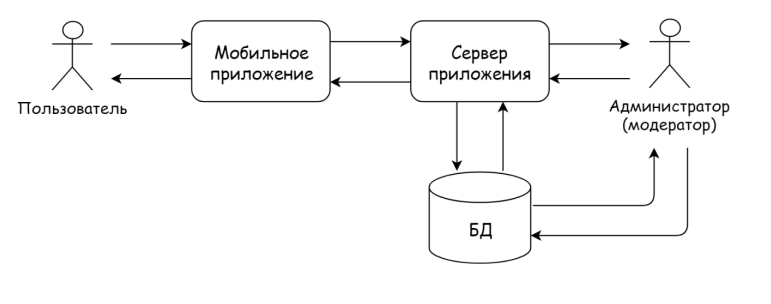


Рисунок 10 –Структурная модель системы

Каждый пользователь может использовать приложение, имея при это доступ к полному функционалу, но для отправки запроса на добавления достопримечательности, необходимо указать электронную почту для обратной связи.

Администратор(модератор) приложения не имеет в распоряжении заточенной под его нужды панели взаимодействия с мобильным приложением или уникальной системы. Подразумевается, что он обладает достаточным опытом и квалификацией для поддержки кода на уровне серверной части и БД.

# Классы и характеристики пользователей

Ниже, в таблице 4, представлены релевантные классы пользователей с точки зрения функционала системы

Таблица 4. Характеристики пользователей системы.

|  |  |
| --- | --- |
| Наименование | Описание |
| Администратор(модератор) | Опытный пользователь, разработчик, обладающий знаниями и навыками для поддержания работы и обслуживания мобильного приложения «Lazy Guide»,его серверной части, а также БД, в которой хранятся данные необходимые приложению для исправного функционирования. В список необходимых знаний входит:   * Опыт прикладных разработок систем на основе MS SQL Server * Уверенное знание языков программирования Kotlin и TypeScript * Понимание языка разметки XML * Опыт работы с протоколом HTTP |
| Пользователь | Каждый пользователь программного продукта, обладающий смартфоном для установки, использования и запуска приложения. |

# Системные функции

# Функциональные требования

В таблице 5 представлены функциональные требования к программному продукту. Реализация спецификации осуществлялась исходя из требований к ПО [4;12]

Таблица 5. Функциональные требования

|  |  |
| --- | --- |
| FUN-ID | Описание |
| FUN-01 | Приложение должно позволять пользователю выбирать интересующий его регион РФ |
| FUN-02 | Приложение должно позволять пользователю отправлять сообщение с информацией о собственной достопримечательности |
| FUN-03 | Приложение должно позволять пользователю использовать поиск по регионам РФ |
| FUN-04 | Приложение должно позволять пользователю возвращаться обратно к списку регионов РФ (раздел «Выбор региона») из разделов «Список достопримечательностей» и «Свяжитесь с нами» |
| FUN-05 | Приложение должно позволять пользователю выбирать достопримечательность из списка |
| FUN-06 | Приложение должно позволять пользователю сортировать список достопримечательностей по их типу |
| FUN-07 | Приложение должно позволять пользователю просматривать подробную информацию о выбранной достопримечательности |
| FUN-08 | Приложение должно позволять пользователю строить маршрут от текущего местоположения до адреса достопримечательности |
| FUN-09 | Приложение должно позволять пользователю прервать построение маршрута и вернутся обратно к подробной информации о достопримечательности |
| FUN-10 | Приложение должны позволять пользователю возвращаться к списку достопримечательностей |

# Нефункциональные требования

В таблицах 6-7 представлены нефункциональные требования к программному продукту.

Таблица 6. Требования к безопасности.

|  |  |
| --- | --- |
| SEC-ID | Описание |
| SEC-01 | В программном продукте должен воплощён механизм защиты от некорректных действий пользователя:   * Проверка данных, которые вводит пользователь * Защита от SQL-инъекций |
| SEC-02 | Система должна позволять только администратору(модератору) создавать, изменять и удалять записи о достопримечательностях, находящихся в БД |

Таблица 7. Требования к доступности

|  |  |
| --- | --- |
| AVL-ID | Описание |
| AVL-1 | Система должна быть доступна 98% времени между 5:00 и 00:00 по местному времени и 90% времени между 00:01 и 5:00 по местному времени, за исключением времени планового обслуживания |

# Ограничения проекта

Ограничения проекта вызваны дальнейшими выбранными языками программирования и инструментами разработки:

* + 1. Приложение должно быть нацелено на девайсы с ОС Android версией не меньше, чем 5.0 (Lollipop).
    2. Для клиентской части программного продукта необходимо использовать язык программирования Kotlin версии 1.5
    3. Серверная часть программного продукта обязана быть развернута на платформе Node.js версии 16.2.0
    4. Серверная часть программного продукта обязана быть на языке TypeScript версии 4.3.1
    5. В роли СУБД необходимо использовать Microsoft SQL Server версии 18.9.1

Подробный процесс анализа сравнения средств разработки выполнен в разделе 1.3

# Варианты использования системы

Словесные представления вариантов использования приведены в таблице 8.

Таблица 8. Варианты использования системы.

|  |  |
| --- | --- |
| Идентификатор ВИ | UC-01 |
| Наименование: | Выбор региона |
| Автор: | Владислав Чепусов |
| Дата создания | 31.05.2021 |
| Требование: | FUN-01, FUN-03 |
| Действующее лицо: | Пользователь |
| Описание: | Пользователь, запустивший приложение, хочет просмотреть список доступных регионов и выбрать интересующий его |
| Условие-триггер: | Пользователь вошел в приложение |
| Предварительные условия: | — |
| Выходные условия: | Переход в раздел «Достопримечательности региона» |
| Основной поток: | 1. Пользователь запускает приложение  2. После экрана загрузки, отображается прокручиваемый список регионов РФ  3. Пользователь выбирает регион 4. Приложение переходит в раздел «Достопримечательности региона» |
| Альтернативный поток: | 1. Пользователь запускает приложение  2. После экрана загрузки, отображается прокручиваемый список регионов РФ  3. Пользователь использует поиск по регионам, чтобы сузить список  4. Пользователь выбирает регион  5. Приложение переходит в раздел «Достопримечательности региона» |
| Исключения: | Отсутствует искомый регион  1. Если пользователь, используя поиск по регионам вводит несуществующий субъект РФ, тогда отображается пустой список.  2. Чтобы вернуть полный перечень регионов РФ, нужно очистить поле поиска.  Отсутствует список достопримечательностей региона  1. При условии, если был выбран регион РФ, но записей о достопримечательности не существует, отобразится в заголовке приложения «Error 404» и на форме отобразится сообщение «Простите, данных нет!» |
| Идентификатор ВИ | UC-02 |
| Наименование: | Отправка запроса на добавление собственной достопримечательности |
| Автор: | Владислав Чепусов |
| Дата создания: | 31.05.2021 |
| Требование: | FUN-02 |
| Действующее лицо: | Пользователь |
| Описание: | Пользователь хочет предложить собственную достопримечательность для размещения в приложении |
| Условие-триггер: | Пользователь переходит в раздел «Свяжитесь с нами», чтобы отправить собственную достопримечательность |
| Предварительные условия: | 1. У пользователя есть выход в интернет |
| Выходные условия: | Отправленное письмо на почту с данными о достопримечательности |
| Основной поток: | 1. Пользователь вошел в приложение  2. Пользователь перешел в раздел «Свяжитесь с нами»  3. Пользователь вводит название достопримечательности  4. Пользователь добавляет изображение достопримечательности (минимум- одно, максимум-шесть)  5. Пользователь выбирает регион, где находится достопримечательность, из раскрывающегося списка  6. Пользователь вводит адрес достопримечательности  7. Пользователь заполняет описание достопримечательности  8. Пользователь должен вбить контактный номер телефона  9. Пользователь должен вбить адрес веб-сайта достопримечательности  10. Пользователь должен заполнить время работы достопримечательности  11. Пользователь должен заполнить стоимость посещения достопримечательности  12. После заполнения всех полей, пользователю нужно нажать на кнопку «Отправить заявку» и выбрать удобный для него способ:  Gmail, mail.ru и т.д. |
| Альтернативный поток: | — |
| Исключения: | Неверное заполнение полей  1. При неправильном заполнении хотя бы одного информационного поля, отправка письма будет недоступна  2. Некорректно заполненное поле подсветиться красным и появится подсказка, чуть ниже самого поля  3. После правильного заполнения поля справа от него появится зеленая стрелочка |
| Идентификатор ВИ | UC-03 |
| Наименование: | Выбор достопримечательности |
| Автор: | Владислав Чепусов |
| Дата создания: | 31.05.2021 |
| Требование: | FUN-05, FUN-06 |
| Действующее лицо: | Пользователь |
| Описание: | Пользователь просматривает список достопримечательностей определённого региона и хочет посмотреть подробную информацию об одной из них |
| Условие-триггер: | Пользователь реши посмотреть подробную информацию о понравившейся достопримечательности |
| Предварительные условия: | 1. У пользователя есть выход в интернет  2. Пользователь сделал выбор интересующего его региона |
| Выходные условия: | Переход в раздел о подробной информации о достопримечательности |
| Основной поток: | 1. Пользователь просматривает полный список достопримечательностей выбранного региона  2. Пользователь выбирает достопримечательность  3. Приложение переходил в раздел подробной информации о достопримечательности |
| Альтернативный поток: | 1.Пользователь решил отсортировать достопримечательности региона по их типу  2. Пользователь выбирает интересующий его тип достопримечательности  3.Пользователь выбирает из нового списка интересующую его достопримечательность  4. Приложение переходил в раздел подробной информации о достопримечательности |
| Исключения: | Отсутствие достопримечательностей определенного типа  1.При условии, что в списке всех достопримечательностей, отсутствуют записи с определенным типом, при попытке выбора данного класса приложение выведет сообщение «Простите, данных нет!» |
| Идентификатор ВИ | UC-04 |
| Наименование: | Просмотр информации о достопримечательности |
| Автор: | Владислав Чепусов |
| Дата создания: | 31.05.2021 |
| Требование: | FUN-07 |
| Действующее лицо: | Пользователь |
| Описание: | Пользователь выбрал достопримечательность из списка и решил узнать о ней больше |
| Условие-триггер: | Пользователя выбрал достопримечательность |
| Предварительные условия: | 1. У пользователя есть выход в интернет  2. Пользователь сделал выбор интересующего его региона  3.Пользователь сделал выбор интересующей его достопримечательности |
| Выходные условия: | Подробная информация о достопримечательности |
| Основной поток: | 1.Пользователь перешел в раздел подробной информации о достопримечательности  2.Пользователь просматривает информационную сводку о достопримечательности:   * Название * Описание * Адрес * Время работы * Цену посещения * Веб-сайт * Контактный номер   3. Пользователь может взаимодействовать с галереей изображений, прокручивая влево и вправо |
| Альтернативный поток: | — |
| Исключения: | — |
| Идентификатор ВИ | UC-05 |
| Наименование: | Построение маршрута на карте |
| Автор | Владислав Чепусов |
| Дата создания: | 31.05.2021 |
| Требование: | FUN-08 |
| Действующее лицо: | Пользователь |
| Описание: | Пользователь решил построить маршрут от текущего местоположения до расположения достопримечательности |
| Условие-триггер: | Пользователь стало интересно, как ему добраться до достопримечательности |
| Предварительные условия: | 1. У пользователя есть выход в интернет  2. Пользователь сделал выбор интересующего его региона  3. Пользователь сделал выбор интересующей его достопримечательности |
| Выходные условия: | Поострённый маршрут на карте, от местоположения пользователя до достопримечательности |
| Основной поток: | 1.Пользователь находится в разделе подробной информации о достопримечательности  2.Пользователь нажимает на маленький фрагмент карты, где отображено географическое расположение достопримечательности  3. Приложение переходит на форму полномасштабной карты  4.Приложение запрашивает у пользователя разрешения на определение его местонахождения  5.Приложение строит маршрут для автомобиля |
| Альтернативный поток: | 1.Пользователь находится в разделе подробной информации о достопримечательности  2.Пользователь нажимает на маленький фрагмент карты, где отображено географическое расположение достопримечательности  3. Приложение переходит на форму полномасштабной карты  4.Приложение запрашивает у пользователя разрешения на определение его местонахождения  5.Приложение строит маршрут для пешехода |
| Исключения: | Пользователь не дал разрешения на определение местоположения  1.Маршрут не будет построен  2.На карте отобразится только геометка достопримечательности  Пользователь дао разрешение, но маршрута между точками не существует  1.Маршрут не будет построен  2.На карте отобразятся геометки пользователя и достопримечательности |
| Идентификатор ВИ | UC-06 |
| Наименование: | Возвращение к подробной информации о достопримечательности |
| Автор: | Владислав Чепусов |
| Дата создания: | 31.05.2021 |
| Требование: | FUN-09 |
| Действующее лицо: | Пользователь |
| Описание: | Пользователь возвращается в раздел подробной информации о достопримечательности |
| Условие-триггер: | Пользователь хочет вернуться от полномасштабной карты в раздел подробной информации о достопримечательности |
| Предварительные условия: | 1. У пользователя есть выход в интернет  2. Пользователь сделал выбор интересующего его региона  3. Пользователь сделал выбор интересующей его достопримечательности  4.Пользователь перешел в раздел полномасштабной карты для построения маршрута |
| Выходные условия: | Возвращение обратно в раздел о достопримечательности |
| Основной поток: | 1.Пользователь находится в разделе полномасштабной карты, где строится маршрут от местонахождения пользователя до выбранной достопримечательности  2.Пользователь нажимает на стрелку в ActionBar (панель действий)  3.Приложение переходит в раздел подробной информации о достопримечательности |
| Альтернативный поток: | — |
| Исключения: | — |
| Идентификатор ВИ | UC-07 |
| Наименование: | Возвращение обратно к разделу выбора региона |
| Автор: | Владислав Чепусов |
| Дата создания: | 31.05.2021 |
| Требование: | FUN-04 |
| Действующее лицо: | Пользователь |
| Описание: | Пользователь возвращается в раздел выбора региона |
| Условие-триггер: | Пользователь хочет вернуться из раздела «Свяжитесь с нами» или «Список достопримечательностей» обратно к выбору региона |
| Предварительные условия: | 1. У пользователя есть выход в интернет  2. Пользователь сделал выбор интересующего его региона, тем самым перейдя в раздел «Список достопримечательностей»  2`. Пользователь перешел в раздел «Свяжитесь с нами» |
| Выходные условия: | Пользователь вернулся в раздел «Выбор региона» |
| Основной поток: | 1.Пользователь находится в разделе «Список регионов»  2.Пользователь нажимает на стрелку в ActionBar (панель действий)  3.Приложение переходит в раздел «Выбор региона» |
| Альтернативный поток: | 1.Пользователь находится в разделе «Свяжитесь с нами»  2.Пользователь нажимает на стрелку в ActionBar (панель действий)  3.Приложение переходит в раздел «Выбор региона» |
| Исключения: | — |
| Идентификатор ВИ | UC-08 |
| Наименование: | Возвращение в раздел «Список достопримечательностей» |
| Автор: | Владислав Чепусов |
| Дата создания: | 31.05.2021 |
| Требование: | FUN-10 |
| Действующее лицо: | Пользователь |
| Описание: | Пользователь, находясь в разделе подробной информации о достопримечательности переходит в раздел «Список достопримечательностей» |
| Условие-триггер: | Пользователь хочет посмотреть на другие достопримечательности региона |
| Предварительные условия: | 1. У пользователя есть выход в интернет  2. Пользователь сделал выбор интересующего его региона  3. Пользователь сделал выбор интересующей его достопримечательности |
| Выходные условия: | Пользователь вернулся в раздел «Список достопримечательностей» |
| Основной поток: | 1.Пользователь находится в разделе подробной информации о достопримечательности  2.Пользователь нажимает на стрелку в ActionBar (панель действий)  3.Приложение переходит в раздел «Список достопримечательностей» |
| Альтернативный поток: | — |
| Исключения: | Отсутствует интернет  1. Если находясь в разделе подробной информации о достопримечательности у пользователя случаются перебои с интернетом, при переходе к «Списку регионов», отобразится в заголовке приложения «Error 404» и на форме отобразится сообщение «Простите, данных нет!» |

# Проектирование базы данных

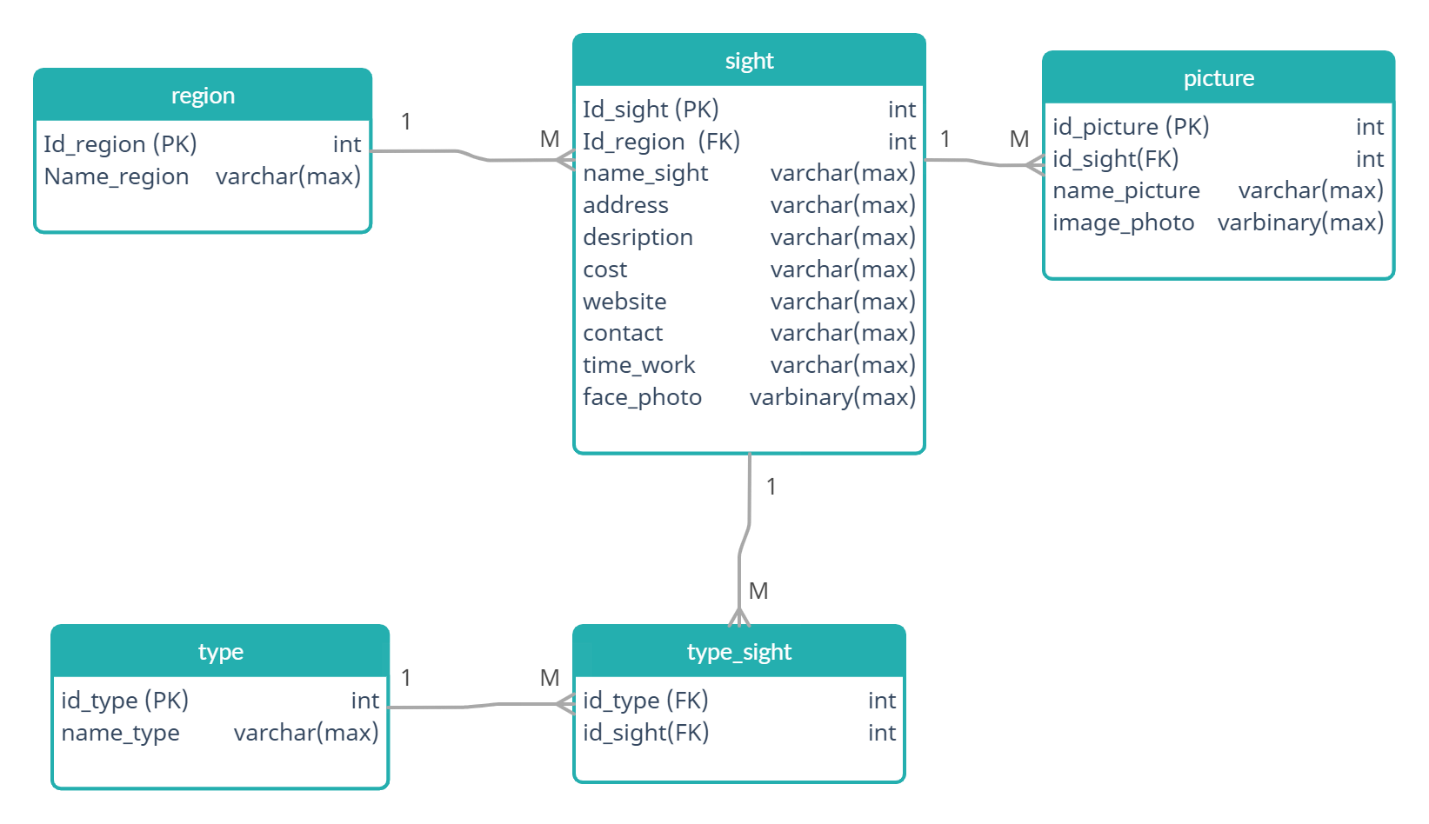
Разработанная БД программного продукта «Lazy Guide» представлена в формате ERD на рис11.

Рисунок 11 –ERD БД «Lazy Guide»

Для достижения лучших результатов разработки, а также для простого представления структуры БД был организован словарь данных (табл. 10), содержащий развернутую информацию о существующих в приложении сущностях и сопроводительная таблица с типами данных(табл.9).

Таблица 9. Принятые обозначения и типы данных

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 1 | int | Целочисленные значения в диапазоне от -231 (-2 147 483 648) до 231-1 (2 147 483 647) |
| 2 | varchar | Строковые данные переменной длины от 1 до 8000 байт. Установление значения max будет восприниматься, как предельный размера столбца вплоть до максимального размера хранилища, что составляет 2^31-1 байт |
| 3 | varbinary | Двоичные данные с переменной длиной от 1 до 8000 байт. Установление значения max будет восприниматься, как предельный размера столбца вплоть до максимального размера хранилища, что составляет 2^31-1 байт |

Таблица 10. Словарь данных

|  |  |  |  |
| --- | --- | --- | --- |
| Структура или  элемент данных | Тип данных | Длина | Значение |
| region | | | |
| Id\_region | int | 4 | Идентификатор региона, порядковый номер, равный коду субъекта Российской Федерации.  (Not Null) |
| Name\_region | varchar | max | Название региона, может содержать только буквы кириллицы, дефисы и пробелы. (Not Null) |
| type | | | |
| id\_type | int | 4 | Генерируемый системой порядковый номер, начиная с 1. (Not Null) |
| name\_type | varchar | max | Название региона, может содержать только буквы кириллицы, дефисы и пробелы. (Not Null) |
| type\_sight | | | |
| id\_type | int | 4 | Генерируемый системой порядковый номер, начиная с 1. (Not Null) |
| Id\_sight | int | 11 | Генерируемый системой порядковый номер, начиная с 1. (Not Null) |
| sight | | | |
| Id\_sight | int | 11 | Генерируемый системой порядковый номер, начиная с 1. (Not Null) |
| Id\_region | int | 4 | Идентификатор региона, порядковый номер, равный коду субъекта Российской Федерации.  (Not Null) |
| name\_sight | varchar | max | Название достопримечательности может содержать, буквы кириллицы и латиницы, а также пробелы, дефисы.  (Not Null) |
| address | varchar | max | Адрес достопримечательности может содержать буквы кириллицы, цифр и запятые  (Not Null) |
| description | varchar | max | — |
| cost | varchar | max | Стоимость может содержать буквы кириллицы, цифр и запятые |
| website | varchar | max | Вебсайт может содержать буквы латиницы и кириллицы числа и символы из следующего после двоеточия списка:  [0-9] / ! \*’ ; : @ & = + $ ? # \_ |
| contact | varchar | max | Телефон может содержать пробелы, числа и символы из следующего после двоеточия списка:  [0-9] ( ) - |
| time\_work | varchar | max | Время работы может содержать буквы кириллицы, числа и символы из следующего после двоеточия списка:  [0-9] ( ) - : |
| face\_photo | varbinary | max | — |
| picture | | | |
| Id\_picture | int | 11 | Генерируемый системой порядковый номер, начиная с 1. (Not Null) |
| Id\_sight | int | 11 | Генерируемый системой порядковый номер, начиная с 1. (Not Null) |
| Name\_picture | varchar | max | — |
| Image\_photo | varbinary | max | — |

# Интерфейсы пользователя

Реализованные интерфейсы пользователя приложения «Lazy Guide» приведены в приложении А. Навигационные карты пользовательского интерфейса, используемые в системе представлена на рисунках 12-16.

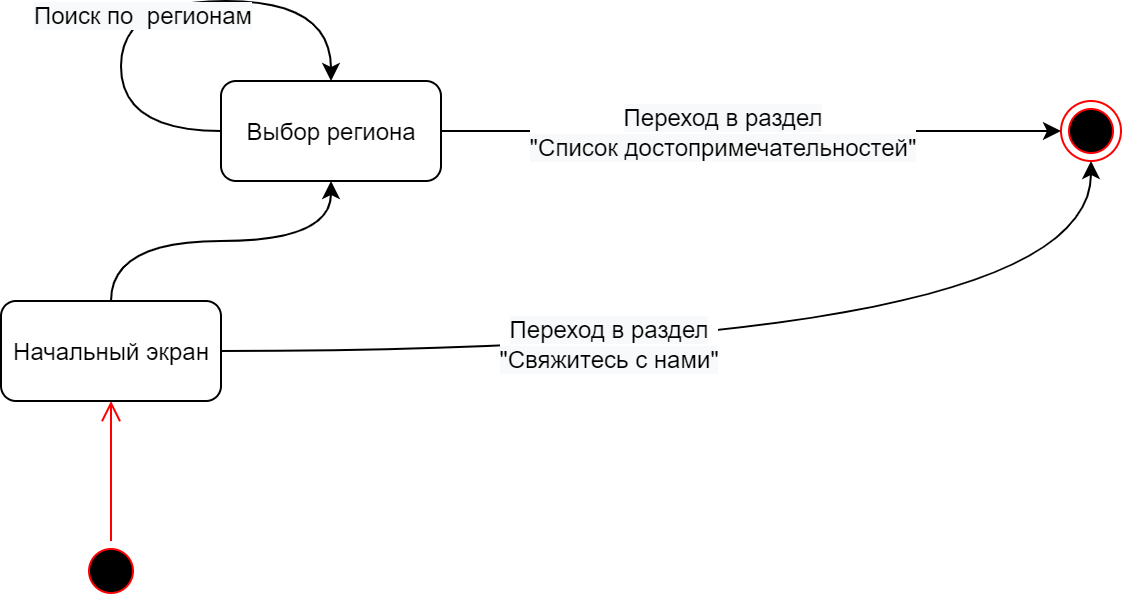


Рисунок 12 -Карта диалоговых окон для раздела «Выбор региона»

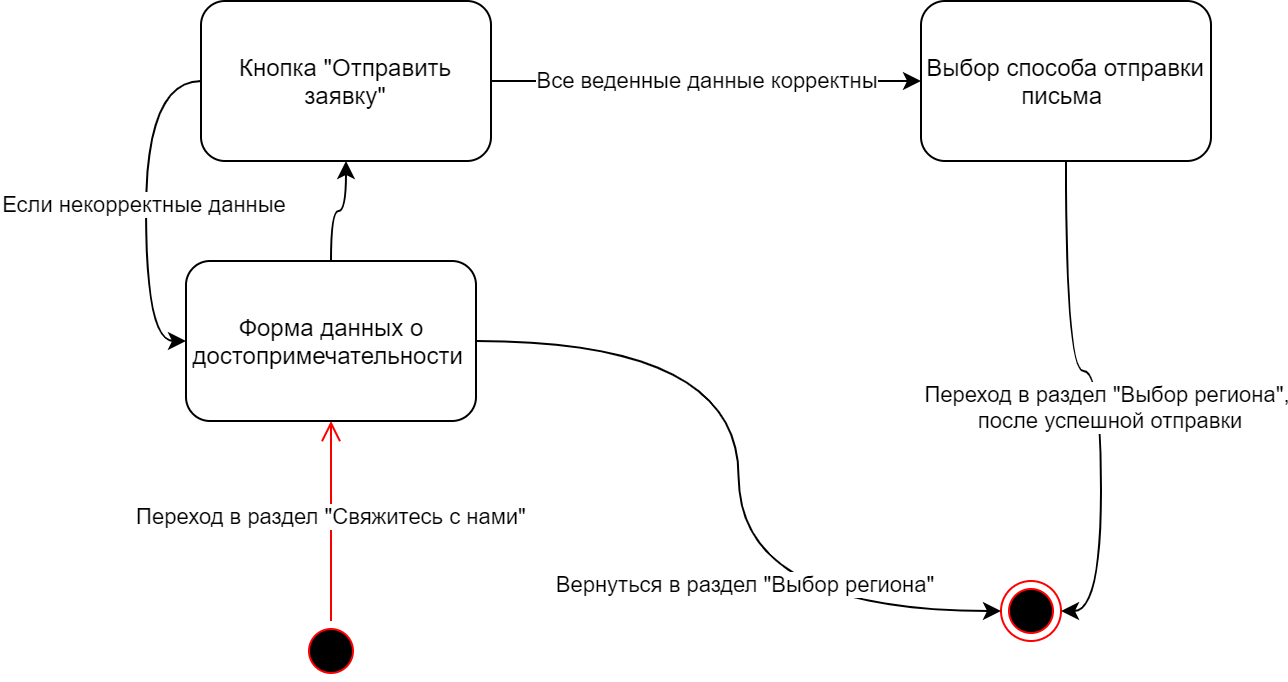
 Рисунок 13 -Карта диалоговых окон для раздела «Свяжитесь с нами»



Рисунок 14 -Карта диалоговых окон

для раздела «Список достопримечательностей»

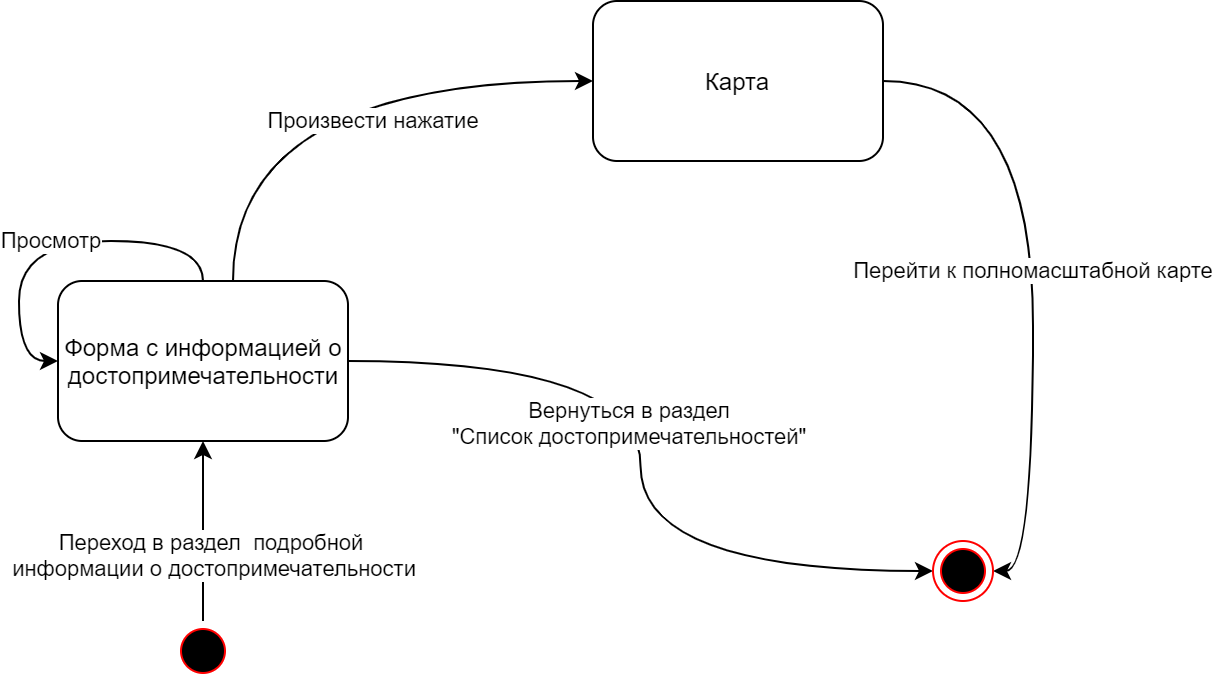


Рисунок 15 -Карта диалоговых окон для раздела

«Подробная информация о достопримечательности»

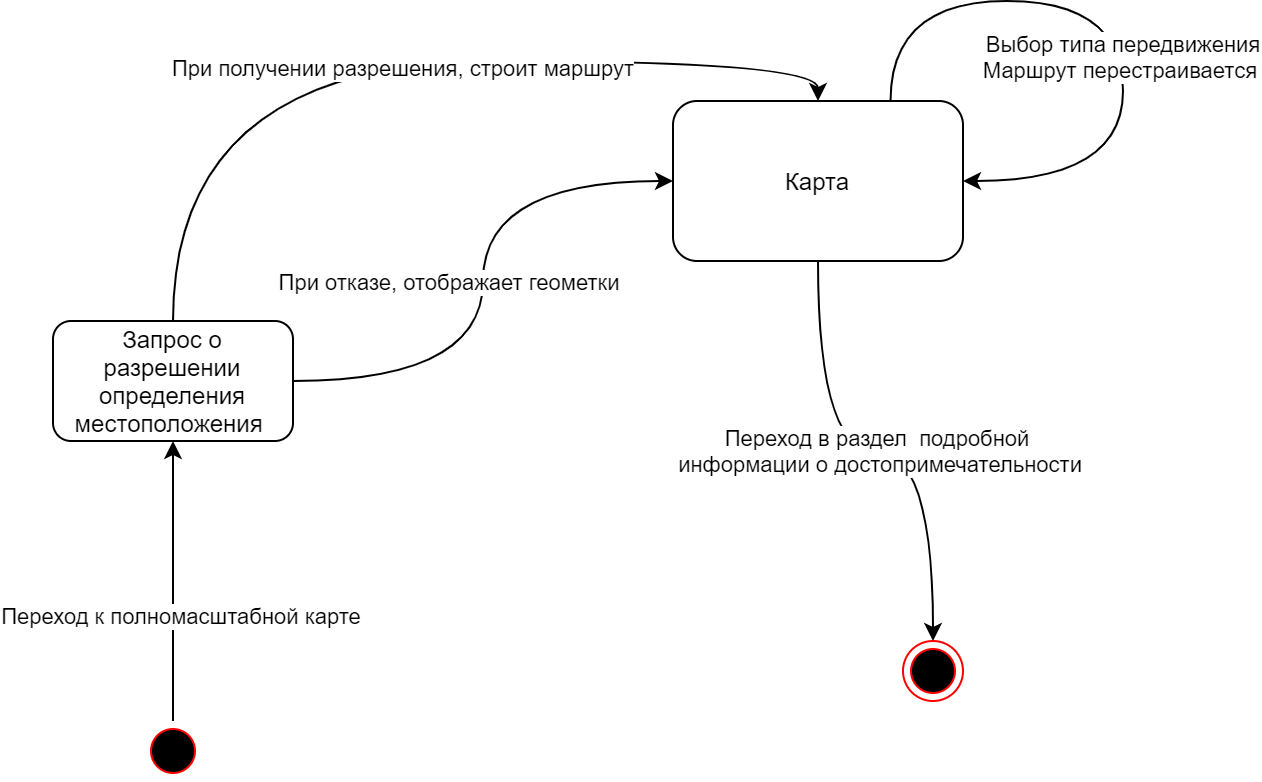


Рисунок 16 -Карта диалоговых окон для раздела

«Полномасштабная карта»

# Реализация требований к безопасности

Требования безопасности к программному продукту «Lazy Guide» были реализованы с помощью инструментов и программных средств, описанных в таблице 11.

Таблица 11 — Реализация требований безопасности.

|  |  |
| --- | --- |
| SEC-ID | Реализация требования |
| SEC-01 | Проверка данных, которые вводит пользователь, осуществляется на уровне клиента. При заполнении полей некорректными данными программный продукт выводит сообщение о неправильном вводе, подмечая красным крестиком неверно заполненный фрагмент. Приложение не позволит отправить сообщения пока каждая из ошибок не будет устранена. Так как информация отправляется не сразу в базу данных, а на почту администратора, угроз заспамления БД и атак вида sql-инъекции можно не опасаться. |
| SEC-02 | Из-за отсутствия реализации изменения, удаления и добавления данных в БД со стороны клиента у пользователя нету возможности самостоятельно влиять на данные. Весь данный функционал ложиться на плечи администратора(модератора), который работает непосредственно с СУБД |

# Выводы к разделу

В настоящем разделе были представлены ключевые аспекты проектирования и разработки мобильного приложения для поддержки внутреннего туризма с рабочим названием «Lazy Guide». В ходе работы была установлена структурная модель системы, отображенная контекстной диаграммой, показывающая взаимодействие программного продукта с пользователями. Приведены функциональные и нефункциональные требования, влияющие на ограничения данного проекта.

При проектировании базы данных была задействована схема «сущность-связь» (ERD), а также словарь данных для установления лучшего способа хранения информации о регионах, достопримечательностях, типах и галереях.

Благодаря таблице ВИ, схемы диалоговых окон было продемонстрировано функционирование программного продукта, предусмотренного настоящей разработкой.

В процессе работы, установлены требования к безопасности приложения и действия для их реализации.

Все интерфейсы мобильного приложения для поддержки внутреннего туризма с рабочим названием «Lazy Guide», представлены в приложении А. Код клиентской и серверной части вместе с комментариями добавлены в приложение Б.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы было реализовано приложения для поддержки внутреннего туризма Российской Федерации. Данный программный продукт может увеличить интерес большого количества людей к исследованию исторических, природных и современных достопримечательностей собственных регионов. Из-за значительной роли смартфонов в жизни человека реализация удобного и быстрого приложения, хранящего в себе данные о культурном наследии, кажется оптимальной, как с точки зрения потребления информации, так и с позиции экономии времени.

Программный продукт, реализованный в настоящей работе, может использоваться обширным числом лиц для достижения собственных интересов — рекламировать собственные достопримечательности, планировать небольшие поездки и путешествия, а также делиться новыми находками с друзьями.

В дальнейшем, «Lazy Guide» возможно будет доработано. Расширена база данных, чтобы была возможность хранить аудиогиды и видео файлы достопримечательностей.

Подводя итоги, главными результатами работы являются:

* + 1. Выполненный анализ предметной области с рассмотрением

аналогов программного продукта. Установлены сильные и слабые стороны схожих приложений.

* + 1. Отбор инструментов для разработки приложения, проведённый с

учетом приобретённых знаний и навыков, простоты изучения, объёмом

документации, а также удобства использования и возможности модификации в будущем.

* + 1. Установление функциональных и нефункциональных требований,

предоставивших возможность установить рамки разработки программного продукта;

* + 1. Осуществлено проектирование, а также реализация БД,

предоставляющая возможность хранения необходимых для программного продукта данных в наилучшей для использования форме.

* + 1. Реализован интерфейс пользователя для приложения на ОС

Android 5.0(lollipop), предоставляющий доступ ко всем функциям системы.

* + 1. Разработаны клиент и сервер приложения для поддержки

внутреннего туризма с перспективой потенциального расширения функционала в дальнейшем.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Пауэрс, Шелли. Изучаем Node. Переходим на сторону сервера = Learning Node: Moving to the Server-Side: пер. с англ.; 2-е изд. /Шелли Пауэрс. — Издательство: Питер, 2017 г.— 304с.
2. Черный,Борис. Профессиональный TypeScript. Разработка масштабируемых JavaScript-приложений /Борис Черный. —Издательство: Прогресс книга,2021 г.— 352 с.
3. Эванс, Эрик. Предметно-ориентированное проектирование: структуризация сложных программных систем = Domain-Driven Design: tackling complexity the heart of software: пер. с англ. / Эрик Эванс. - М.: ИД Вильямс, 2011. - 448 с.
4. 830-1998 - IEEE Recommended Practice for Software Requirements Specifications. - NY: The Institute of Electrical and Electronics Engineers, 2009. - 31 c
5. Leiva, Antonio. Kotlin for Android Developers/ Antonio Leiva. —Издательство: CreateSpace Independent Publishing Platform,2016 г.—226 с.
6. Android Studio [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: <https://ru.wikipedia.org/wiki/Android_Studio>  
    (дата обращения 31.05.2021)
7. Android Studio.User guide[Электронный ресурс]: Android Developers—URL: <https://developer.android.com/studio/intro>  
   (дата обращения: 31.05.2021)
8. Kotlin [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/Kotlin  
   (дата обращения 31.05.2021)
9. Microsoft SQL Server [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: <https://ru.wikipedia.org/wiki/Microsoft_SQL_Server>  
   (дата обращения: 31.05.2021)
10. Node.js [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL:https://ru.wikipedia.org/wiki/Node.js (дата обращения 31.05.2021)
11. TypeScript [Электронный ресурс]: Википедия. Свободная энциклопедия. – <URL:https://ru.wikipedia.org/wiki/TypeScript> (дата обращения 31.05.2021)
12. Writing Software Requirements Specifications (SRS) [Электронный ресурс]. — INKtopia Limited, 2020 — URL: <https://techwhirl.com/writing-software-requirements-specifications/> (дата обращения: 31.05.2021)
13. Краткая справка о платформе Microsoft SQL Server. [Электронный ресурс]: Navicon— URL: <https://navicongroup.ru/platforms/4025/>   
    (дата обращения: 31.05.2021)
14. Стоит ли TypeScript усилий? [Электронный ресурс]: Хабр. —URL:<https://habr.com/ru/post/308374/> (дата обращения: 31.05.2021)
15. Уткина, Алина. Java vs Kotlin для Android-разработки: ответы «за» и «против» [Электронный ресурс]:информационный ресурс Tproger—URL: <https://tproger.ru/articles/java-vs-kotlin/> (дата обращения: 31.05.2021)

# ПРИЛОЖЕНИЕ А

Пользовательские интерфейсы



Рисунок 17—Загрузочный экран

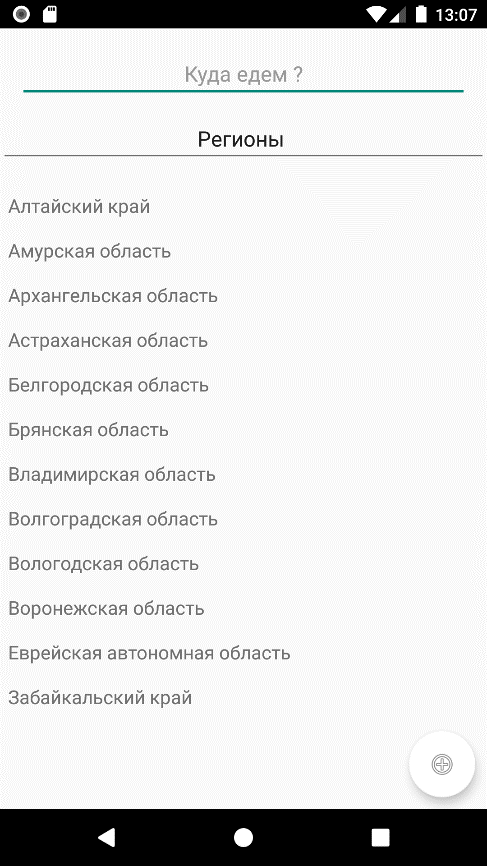


Рисунок 18 —Раздел «Главное меню»

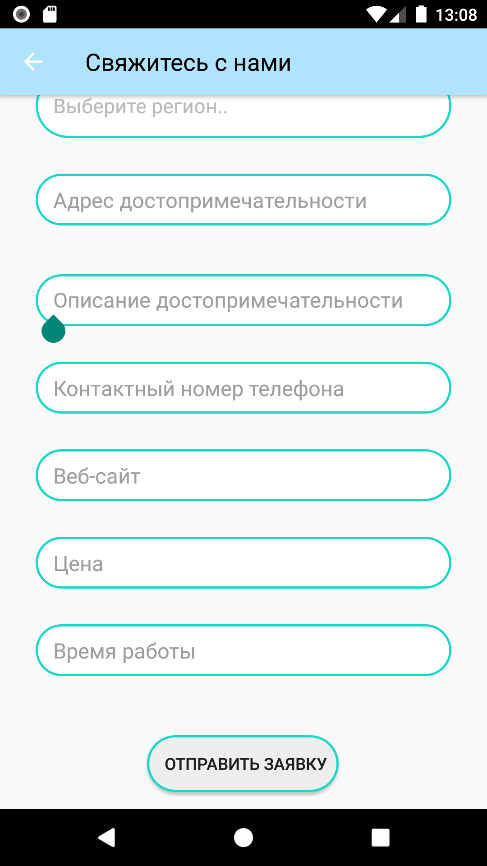
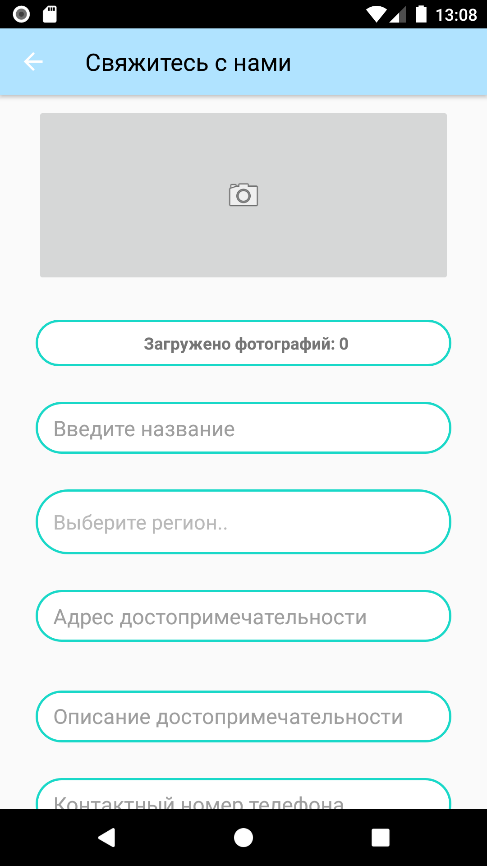


Рисунок 19—Раздел «Свяжитесь с нами»

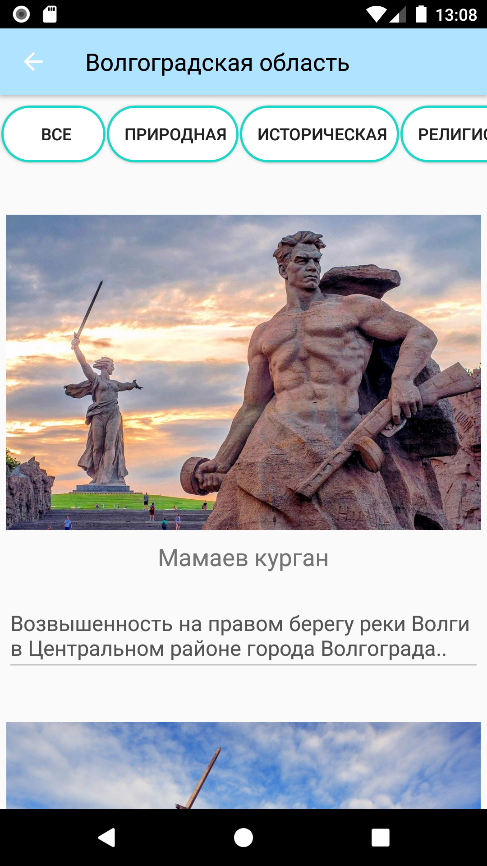


Рисунок 20—Радел «Список достопримечательностей»



Рисунок 21—Радел «Подробней о достопримечательности» ч.1

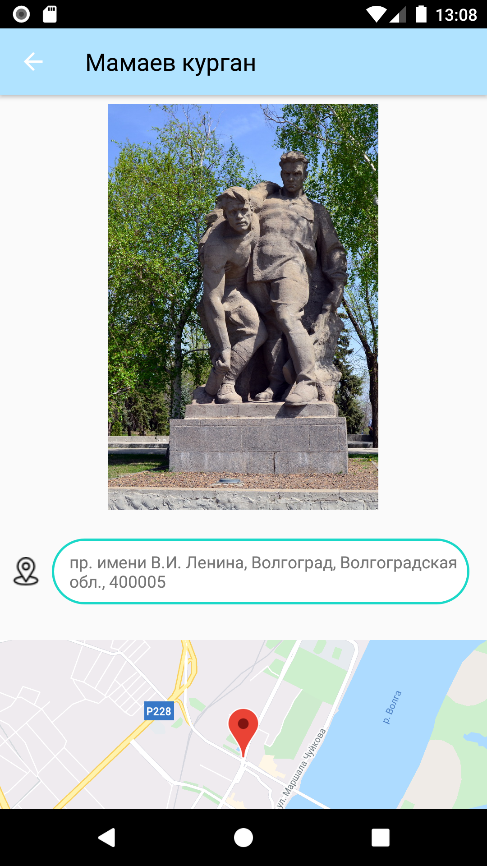


Рисунок 22—Радел «Подробней о достопримечательности» ч.2

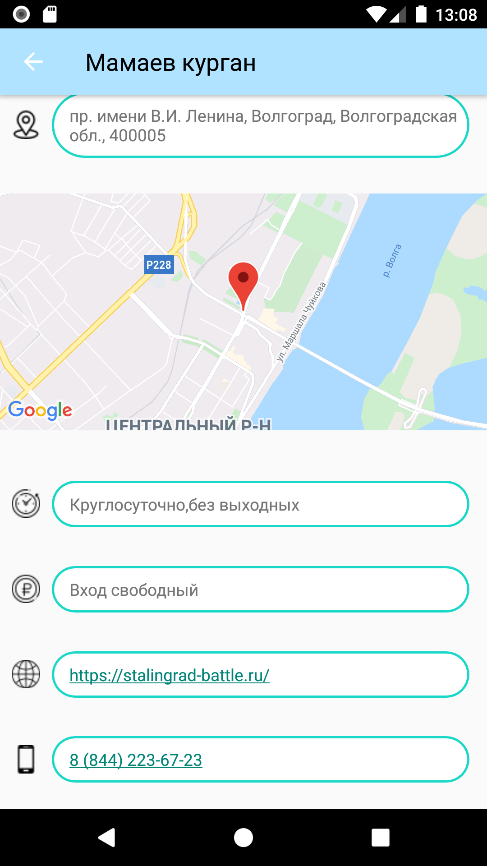


Рисунок 23—Радел «Подробней о достопримечательности» ч.3

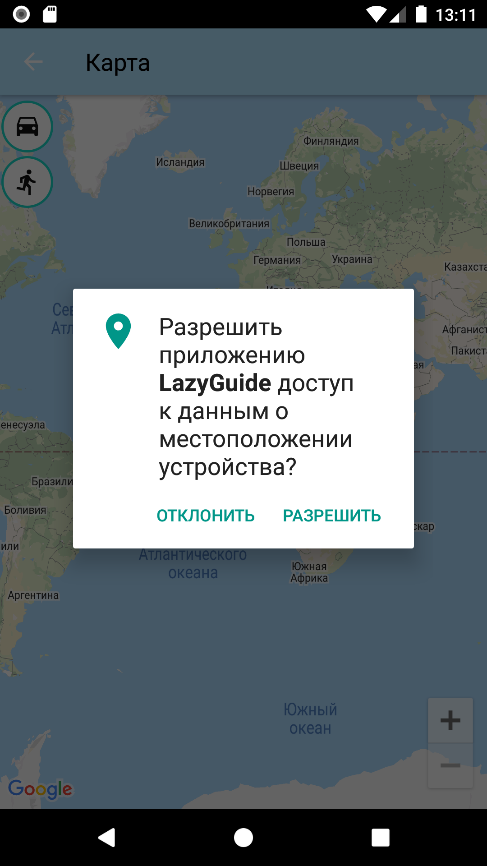


Рисунок 24—Радел «Полномасштабная карта»

Запрос на разрешение

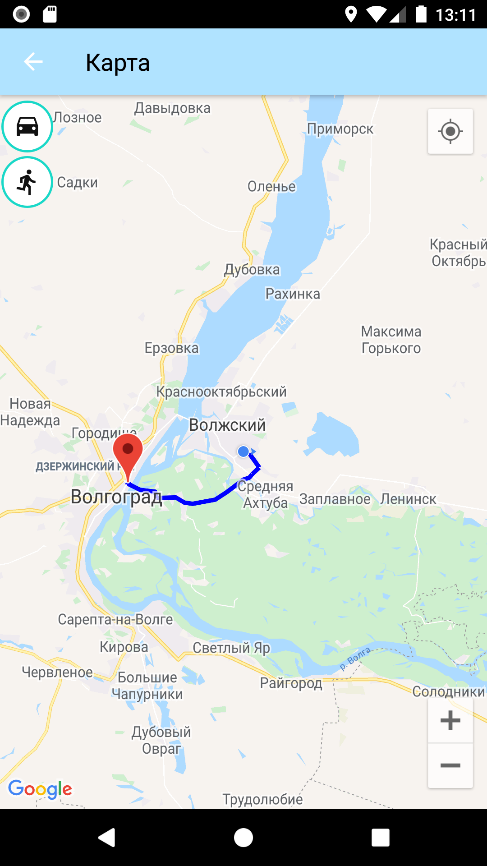


Рисунок 25—Радел «Полномасштабная карта»

Построение маршрута на автомобиле

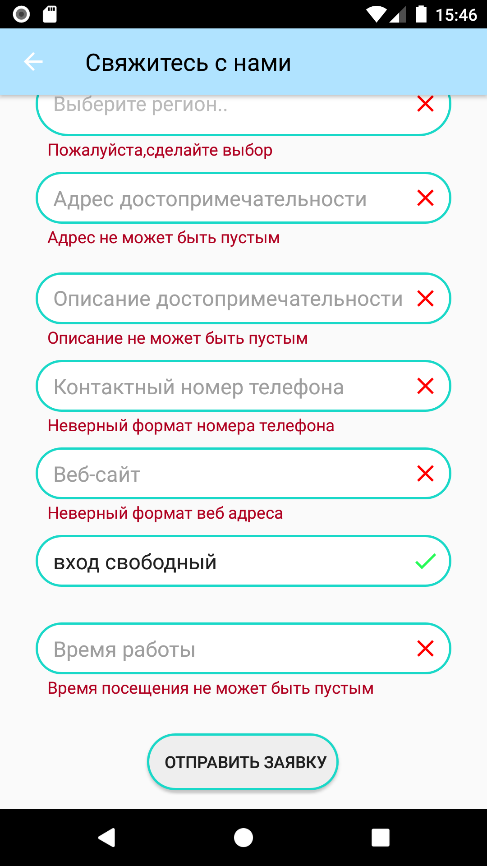
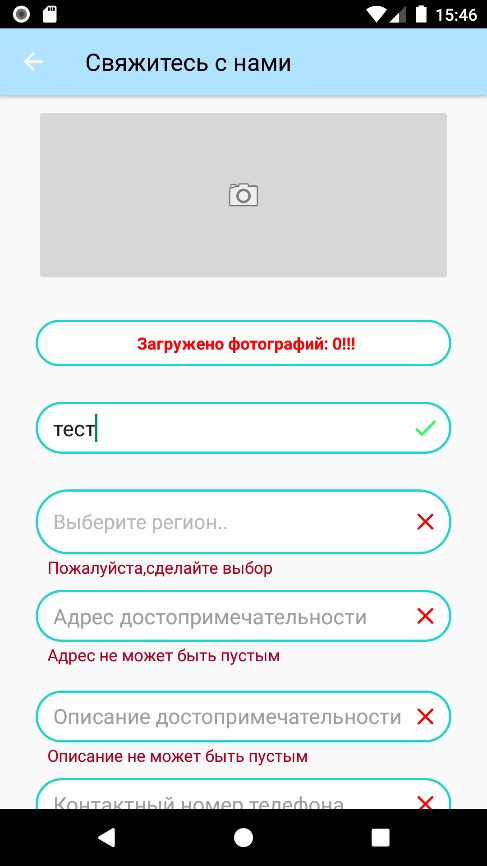


Рисунок 26— Раздел «Свяжитесь с нами»

Проверка ввода

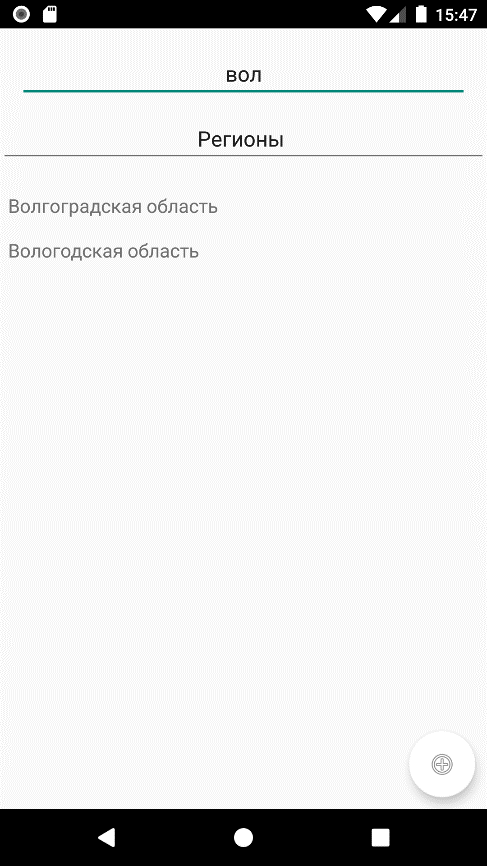


Рисунок 27 —Раздел «Главное меню»

Поиск региона по названию/части названия

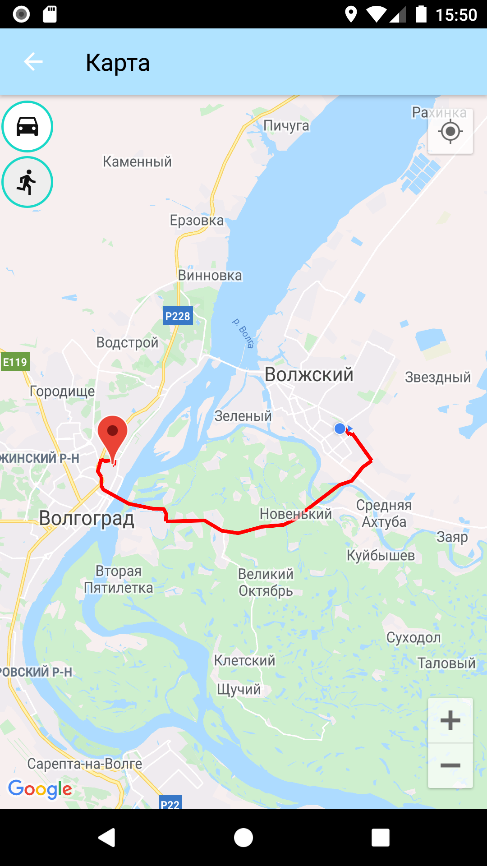


Рисунок 28—Радел «Полномасштабная карта»

Построение маршрута для пешехода

# ПРИЛОЖЕНИЕ Б

Листинг программного продукта

Файл serverND.ts (серверная часть):

//\*\*\*\*\*Важные переменные\*\*\*\*\*\*\*//

// конфиг для подключения к БД

const config = {

port: 1433,

server: 'LAPTOP-CH2DPNJ6',

user: 'test',

password: 'test',

database: 'DiplomDatabase',

stream: false,

options: {

trustedConnection: true,

encrypt: true,

enableArithAbort: true,

trustServerCertificate: true,

},

}

var sql = require('mssql');

cоnst httр = rеquire('httо');

cоnst url = rеquirе('url');

var value;

//Функция вариации sql запросов

function zapros\_(key, id\_smb, callback) {

var zapros;

switch (key) {

case 1:

//На достопримечательности

zapros = `select sight.id\_sight as Id\_sight,sight.id\_region as id\_region,name\_sight,description,Name\_region,face\_photo,name\_type

from sight left join region on sight.Id\_region = region.Id\_region FULL OUTER JOIN type\_sight ON sight.Id\_sight = type\_sight.id\_sight

FULL OUTER JOIN type ON type\_sight.id\_type = type.id\_type where sight.Id\_region = ${id\_smb};`

break

case 2:

// запрос выборку из галерея

zapros = ` select \* from picture where id\_sight = ${id\_smb}`;

break

default:

zapros = `select \* from region`;

}

//обработчик ошибок

sql.connect(config).then(function () {

new sql.Request().query(zapros).then(function (recordset) {

return callback(recordset.recordsets[0]);

}).catch(function (err) {

console.dir("error!Sorry(1)");

console.dir(err);

});

}).catch(function (err) {

console.dir("error!Sorry(2)");

console.dir(err);

});

}

//Создание и прослушивание сервера

http.createServer((rеquеst, rеsроnsе) => {

console.log('server work');

rеsроnsе.writеHеаd(200, { 'Cоntеt-Type': 'application/json' });

//Позволит реализовывать разные страницы

var urlRequest = url.parse(request.url, true);

switch (true) {

//Раздел список достопримечательностей

case (request.url.includes("/getRegionSight")):

//Получение значения из строки запроса

let id\_region = urlRequest.query.test;

zapros\_(1, id\_region, function (result) {

console.dir(result);

value = (JSОN.stringify(rеsult));

response.write(value);

response.end();

});

break

//Раздел подробней достопримечательностей

case (request.url.includes("/getSight")):

//Получение значения из строки запроса

let id\_sigth = urlRequest.query.test;

zapros\_(2, id\_sigth, function (result) {

value = (JSОN.stringify(rеsult));

response.write(value);

response.end();

});

break

default:

response.writeHead(404);

response.end(JSON.stringify({ error: "!Resource not found! :с" }));

}

}).listen(3000);//Порт 3000

Файл sight\_class.kt (клиентская часть, класс для представления данных в списке достопримечательностей):

package com.example.lazyguide.DataClass

/\*Класс чтобы можно было хранить информацию о достопримечательностях

/\* (В разделе список достопримечательностей)

data class sight\_class(

val Id\_sight:Int? = null,

val name\_sight: String? = null,

val description:String? = null,

val Name\_region:String? = null,

vаl nаmе\_tуре:String? = null,

val face\_photo: BufferJS

)

Файл Sight\_result.kt (клиентская часть, класс для представления подробных данных об одной достопримечательности):

package com.example.lazyguide.DataClass

/\*Класс чтобы можно было хранить информацию о достопримечательности\*/

data class Sight\_result(

val id\_sight: Int,

val id\_region: Int,

val name\_sight: String,

val adress:String,

val dicription:String,

val cost:String,

val website:String,

val contact:String,

val time\_work:String,

val photo\_sight: Bitmap

)

Файл BufferJS.kt (клиентская часть, класс для представления изображения):

package com.example.lazyguide.DataClass

data class BufferJS(

val type: String? = null,

val data: ByteArray

)

Файл MainActivity.kt (клиентская часть, класс реализации activity для раздела «Выбор региона», отвечает за функционал данного интерфейса):

package com.example.lazyguide.Activity

/\*

Для активити main(они же регионы)

ОТображение списка регонов через RecyclerView

Переход на активти достопримечательности регоинов

Переход на активити добавить дотсопримечаетльность

\*/

class МаinАсtivity : AррСоmpаtАсtivity() {

//Для взаимодействия с формой

private lateinit var MainBinding : ActivityMainBinding

//Для взаимодействия с RecyclerView и ее хаполнения

lateinit var rесусlerview: RесусclerViеw

lateinit var lауоutmаnаger: RесусclerViеw.LayoutManager

lateinit var adapter: regionAdapter

//Список регонов который отправится в RecyclerView

var region\_list =arrayListOf<String>()

//Список регионов с их индексами(чтобы лишний раз не подключаться к БД)

val RF\_region = mapOf(

22 to "Алтайский край",

28 to "Амурская область",

29 to "Архангельская область",

30 to "Астраханская область",

31 to "Бeлгoрoдская oбласть",

32 to "Брянская область",

33 to "Владимирская область",

34 to "Волгоградская область",

35 to "Вологодская область",

36 to "Воронежская область",

79 to "Eврeйская aвтoнoмная oбласть",

75 to "Забайкальский край",

37 to "Ивановская область",

38 to "Иркутская область",

7 to "Кабардино-Балкарская Республика",

39 to "Калининградская область",

40 to "Калужская область",

41 to "Камчатский край",

9 to "Кaрaчaeвo-Черкeccкая Республика",

42 to "Кeмeрoвскaя oблаcть/Кузбасс",

43 to "Кировская область",

44 to "Костромская область",

23 to "Краснодарский край",

24 to "Красноярский край",

45 to "Курганская область",

46 to "Курская область",

47 to "Ленинградская область",

48 to "Липецкая область",

49 to "Maгaдaнcкая область",

50 to "Московская область"

)

override fun onСrеаte(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

MainBinding = ActivityMainBinding.inflate(layoutInflater)

setContentView(MainBinding.root)

}

//По нажатию на регион открывает новое активити с достопримечательностями

fun onClickGoRegionSight(view:View) {

val b: TextView = view as TextView

val TextView: String = b.gеtТехt().toString()

//Мб регулярку

val key = RF\_region.filterValues { it == TextView }.keys.toString().replace("[","").replace("]","")

val next = Intent(this, RegionSight::class.java)

next.putExtra(RegionSight.NUMBER,key)

startActivity(next)

finish()

}

//Переход на форму "добивать достопримечательность"

fun onClickAdd(view:View) {

startActivity(Intent(this, AddSight::class.java))

finish()

}

//recyclerview заполняется

override fun onResume() {

super.onResume()

//Список регионов

region\_list = ArrayList<String>(RF\_region.values)

//Заполнение recyclerview

recyclerview = MainBinding.rvRegionList

rесусlеrviеw.sеtHаsFixеdSizе(truе)

layoutmanager = LinearLayoutManager(this)

recyclerview.setLayoutManager(layoutmanager)

adapter = regionAdapter(this,region\_list)

recyclerview.setAdapter(adapter)

//Для изменения списка регионов(Поиск по названию)

val editText:EditText = MainBinding.edittext

еditТехt.addТехtChangedListеnеr(object:TextWatcher {

override fun afterTextChanged(s: Editable?) {

filter(s.toString())

}

} )

}

//Функция для поиска региона по названию//Изменяет список отображаемыый recyclerview

private fun filter(text:String){

val filterList:ArrayList<String> = arrayListOf<String>()

for(item:String in region\_list){

if(item.toLowerCase().contains(text.toLowerCase())){

filterList.add(item);

}

}

adapter.filterList(filterList)

}

}

Файл AddSight.kt (клиентская часть, класс реализации activity для раздела «Свяжитесь с нами», отвечает за функционал данного интерфейса):

package com.example.lazyguide.Activity

/\*

Для активити добавления достопримечательностей пользователем

Отправляет данные на почту

Работа с галереей

\*/

class AddSight: AppCompatActivity(), AdapterView.OnItemSelectedListener {

private lateinit var AddBinding : ActivityAddBinding

// private var photo1 : Uri? = null

private var photo1:ArrayList<Uri> = ArrayList<Uri>()

private var photogalary : Uri? = null

val cea = R.layout.spinner\_style1

val Rf\_region = mutableListOf(

"Выберите регион..",

"Алтайский край (22)",

"Амурская область (28)",

"Архангельская область (29)",

"Астраханская область (30)",

"Белгородская область (31)",

"Брянская область (32)",

"Владимирская область (33)",

"Волгоградская область (34)",

"Вологодская область (35)",

"Воронежская область (36)",

"Еврейская автономная область (79)",

"Забайкальский край (75)",

"Ивановская область (37)",

"Иркутская область (38)",

"Кабардино-Балкарская Республика (7)",

"Калининградская область (39)",

"Калужская область (40)",

"Камчатский край (41)",

"Карачаево-Черкесская Республика (9)",

"Кемеровская область/Кузбасс (42)",

"Кировская область (43)",

"Костромская область (44)",

"Краснодарский край (23)",

"Красноярский край (24)",

"Курганская область (45)",

"Курская область (46)",

"Ленинградская область (47)",

"Липецкая область (48)",

"Магаданская область (49)",

"Московская область (50)"

)

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

AddBinding = ActivityAddBinding.inflate(layoutInflater)

setContentView(AddBinding.root)

//Для стелочки

val actionBar: ActionBar? = supportActionBar

actionBar?.setHomeButtonEnabled(true)

actionBar?.setDisplayHomeAsUpEnabled(true)

title = "Свяжитесь с нами"

}

//Кнопка газад на action bar,чтобы вернуться в меню

override fun onOptionsItemSelected(item: MenuItem): Boolean {

startActivity(Intent(this, MainActivity::class.java))

finish()

return true

}

//Загрузка спинера

override fun onResume() {

super.onResume()

//Работа со спинером

val zspiner = AddBinding.RegionSpinner

// val adapter: ArrayAdapter<String> = ArrayAdapter<String>(this, android.R.layout.simple\_spinner\_item, Rf\_region)

val adapter: ArrayAdapter<String> = ArrayAdapter<String>(this, cea, Rf\_region)

adapter.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);

zspiner.setAdapter(adapter)

zspiner.setOnItemSelectedListener(this);

}

////////////////////////////////////////РАБОТА С ГАЛЕРЕЕЙ /////////////////////////////////////////////////////////////////////

val Pick\_image = 1

fun onClickImageDowload(view:View){

//Кол-во пикч

val PICK\_IMAGE = 1

val intеnt = Intent()

intеnt.tуре = "imаgе/\*"

intent.action = Intent.АСTION\_GЕТ\_СОNTENT

startActivityForResult(Intent.createChooser(intent, "Выберите фото достопримечательности"), PICK\_IMAGE)

}

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {

super.onActivityResult(requestCode, resultCode, data)

when (requеstCоdе) {

Pick\_imаgе -> if (tCоdе === RЕSULT\_ОК && rеsultСоdе == Activity.RеSULT\_ОК && null != dаtа) {

try {

//объект и отображаем в элементе ImаgеViеw нашего интерфейса:

val imаgеUri: Uri? = dаtа?.gеtDаtа()

val imageStream: InputStream? = imageUri?.let {

contentResolver.openInputStream(

it

)

}

val sеlеctеdImаgе = BitmаpFаctоry.dеcоdeStream(imageStream)

if (imаgеUri != null) {

photo1.add(imageUri)

AddBinding.countFoto.setTextColor(Color.GRAY)

AddBinding.countFoto.text = "Загружено фотографий: ${photo1.size}"

}

AddBinding.imageButton.setImageBitmap(selectedImage)

} catch (e: FileNotFoundException) {

e.printStackTrace()

}

}

}

}

//Отправка письма

fun onClickMail(view:View){

var MyEmail:String = "lazyguide2021@gmail.com"

var Tema:String = "!НОВАЯ ДОСТОПРИМЕЧАТЕЛЬНОСТЬ!"

var name = (AddBinding.NameText)

var region = AddBinding.RegionSpinner

var address = AddBinding.addressText

var description = AddBinding.discriptinSight

var phone = AddBinding.phoneNumber

var web = AddBinding.webText

var time = AddBinding.timeworkText

var price = AddBinding.priceText

//регулярки

val checkPhoneNumber = """(\+7|8)[\s(]\*\d{3}[)\s]\*\d{3}[\s-]?\d{2}[\s-]?\d{2}"""

var pole = mutableListOf(name,address,description,phone,time,price,web)

var check:Boolean = true

//Проверка на пустоту

for (i in pole ){

if (i.text.toString().isEmpty()) {

i.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,com.example.lazyguide.R.drawable.ic\_baseline\_close\_24, 0)

check = false

}

else {

i.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.check, 0)

}

}

//Проверка наличия фото

if (photo1.isEmpty()){

AddBinding.countFoto.text = "Загружено фотографий: 0!!!"

AddBinding.countFoto.setTextColor(Color.RED)

check = false

}

val che = findViewById<TextView>(R.id.textspiner)

//Проверка выбора региона

if (region.selectedItem.toString() == "Выберите регион.."){

che.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,com.example.lazyguide.R.drawable.ic\_baseline\_close\_24, 0)

check = false

AddBinding.errorRegion.text="Пожалуйста,сделайте выбор"

}else{

che.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.check, 0)

AddBinding.errorRegion.text=""

}

//Проверка номера

if (!phоnе.tеxt.tоString().matches(Rеgеx(chеckPhоnеNumber))){

phone.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,com.example.lazyguide.R.drawable.ic\_baseline\_close\_24, 0)

AddBinding.errorPhone.text ="Неверный формат номера телефона"

check = false

}else{

phone.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.check, 0)

AddBinding.errorPhone.text =""

}

//Проверка веб-адреса

if (Patterns.WEB\_URL.matcher(web.text.toString()).matches()){

web.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.check, 0)

AddBinding.errorWeb.text =""

}

else {

web.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.ic\_baseline\_close\_24, 0)

check = false

AddBinding.errorWeb.text="Неверный формат веб адреса"

}

//Проверка названия

if (name.text.toString().count() == 0){

name.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.ic\_baseline\_close\_24, 0)

check = false

AddBinding.errorName.text="Название не может быть пустым"

}else {

AddBinding.errorName.text=""

name.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.check, 0)

}

//Описание проверка (сколько символов?)

if (description.text.toString().count() == 0){

description.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.ic\_baseline\_close\_24, 0)

check = false

AddBinding.errorDirection.text="Описание не может быть пустым"

}else {

AddBinding.errorDirection.text=""

description.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.check, 0)

}

if (price.text.toString().count() == 0){

price.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.ic\_baseline\_close\_24, 0)

check = false

AddBinding.errorCost.text="Стоимость не может быть пустой"

}else {

AddBinding.errorCost.text=""

price.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.check, 0)

}

if (time.text.toString().count() == 0){

time.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.ic\_baseline\_close\_24, 0)

check = false

AddBinding.errorTime.text="Время посещения не может быть пустым"

}else {

AddBinding.errorTime.text=""

time.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.check, 0)

}

if (address.text.toString().count() == 0){

address.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.ic\_baseline\_close\_24, 0)

check = false

AddBinding.errorAddress.text="Адрес не может быть пустым"

}else {

AddBinding.errorAddress.text=""

address.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.check, 0)

}

if ( !check ) {

// AddBinding.scrollpage.scrollTo(0,0)

AddBinding.scrollpage.fullScroll(ScrollView.FOCUS\_UP);

}

else {

var text: String = "\*! ПОЖАЛУЙСТА,НЕ ИЗМЕНЯЙТЕ ПИСЬМО!\* \n\n" +

"Название достопримечательности: ${name.text.toString()}\n\n" +

"Регион: ${region.selectedItem.toString()}\n\n" +

"Адрес: ${address.text.toString()} \n\n" +

"Описание: ${description.text.toString()}\n\n" +

"Контактный номер телефона: ${phone.text.toString()} \n\n" +

"Веб-сайт: ${web.text.toString()} \n\n" +

"Цена: ${price.text.toString()} \n\n" +

"Время работы: ${time.text.toString()} \n\n" +

"Фотографии: \n "

val еmаil = Intеnt(Intеnt.АСТION\_SЕND\_МULТIРLЕ)

еmаil.sеtТуре("tеxt/ еmаil ")

еmаil.putЕхtrа(Intent.ЕХТRA\_ЕМAIL, arrayOf(MyEmail))

еmаil. putЕхtrа (Intent. ЕХТRA \_SUВJЕCT, Tema)

еmаil. putЕхtrа (Intent. ЕХТRA \_ТЕХТ, text)

еmаil.putParcelableArrayListExtra(Intent. ЕХТRA \_STREAM, photo1);

//для того чтобы запросить email клиент устанавливаем тип

еmаil.type = "message/rfc822"

startActivity(Intent.createChooser(еmаil, "Сделайте выбор почты:"))

}

}

override fun onItemSelected(parent: AdapterView<\*>?, view: View?, position: Int, id: Long) {

if (position != 0) {

val che = findViewById<TextView>(R.id.textspiner)

che.setTextColor(Color.BLACK)

}

}

override fun onNothingSelected(parent: AdapterView<\*>?) {

TODO("Not yet implemented")

}

}

Файл Splash.kt (клиентская часть, класс реализации activity для экрана загрузки, отвечает за функционал данного интерфейса):

package com.example.lazyguide.Activity

/\*Для загрузочного активити Задержка чтобы все подгрузилось и чтобы посмотреть на заставку\*/

class Splash : AppCompatActivity(){

// This is the loading time of the splash screen

private val SPLASH\_TIME\_OUT:Long = 3000 // 1 sec

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_splash)

Handler().postDelayed({

startActivity(Intent(this, MainActivity::class.java))

finish()

}, SPLASH\_TIME\_OUT)

}

}

Файл RegionSight.kt (клиентская часть, класс реализации activity для раздела «Список достопримечательностей», отвечает за функционал данного интерфейса):

package com.example.lazyguide.Activity

/\*Отвечает за страницу со списком достопримечательностей Используется SQL (подключение\отлючение и выкачкаданных Заполнение MutableList<sight\_class> этими данными Отображение на RecyclerView\*/

class RegionSight : AppCompatActivity() {

//Константа из предыдущего активити(номер региона)

companion object{

const val NUMBER = "0"

}

private val typelist:ArrayList<String> = arrayListOf(

"Все",

"Природная",

"Историческая",

"Религиозная",

"Садово-парковая",

"Военно-патриотическая",

"Культурная",

"Архитектурная",

"Монументальная",

"Контркультурная")

//Для полкючения к серверу

private lateinit var mSеrvicе: RеtrоfitSеrvicеs

lateinit var diаlоg: АlеrtDiаlоg

//для RecyclerView

private lateinit var RegionSight : ActivityRegionSightBinding

private lateinit var rесclerview: RесусlerView

private lateinit var layoanager: RecyclerView.LayoutManager

//список достопримечательностей

private val region\_sight: MutableList<sight\_class> = mutableListOf()

private var backe: Button? = null

//Функция запроса на сервер

private fun gеtAllRegionSightList(id\_region:Int,callMy:RevealCourtPlaceCallbacks) {

dialog.show()

//Вызов функции из regrofitservices

mService.getRegionSightList(id\_region).enqueue(object : Callback<MutableList<sight\_class>> {

override fun onFailure(call: Call<MutableList<sight\_class>>, t: Throwable) {

Log.d("Mylog ","Это печально ")

callMy.onError()

}

@RеquirеsАрi(Build.VЕRSIОN\_СОDЕS)

override fun onResponse(call: Call<MutableList<sight\_class>>, response: Response<MutableList<sight\_class>>) {

if (response.isSuccessful) {

val check = response.body() as MutableList<sight\_class>

callMy.onSuccess(check)

}

dialog.dismiss()

}

})

dialog.dismiss()

}

//Кнопка назад на action bar

override fun onОptionsItеmSеlеctеd(item: MеnuItеm): Bооlean {

startActivity(Intent(this, MainActivity::class.java))

finish()

return true

}

override fun onСrеаte(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

//Для формы

RegionSight = ActivityRegionSightBinding.inflate(layoutInflater)

setContentView(RegionSight.root)

//Для стелочки

val actionBar: ActionBar? = supportActionBar

actionBar?.setHomeButtonEnabled(true)

actionBar?.setDisplayHomeAsUpEnabled(true)

val id\_region = intent.gеtStringЕхtra(NUMBER).toInt()

mService = Common.retrofitService

diаlоg = SроtsDiаlоg.Buildеr().setCancelable(true).setContext(this).build()

getAllRegionSightList(id\_region, object:RevealCourtPlaceCallbacks{

override fun onSuccess(value: MutableList<sight\_class>){

region\_sight.addAll(value)

}

override fun onError(){

}

})

}

fun OnClickTypeChose(view:View){

if (backe == null){ }

else

{ backe!!.sеtВаckgrоundRеsоurсе(R.drawable.oblick) }

val b: Button = view as Button

backe = b

b.setBаckgrоundRеsоurcе(R.drawable.buttonadd)

val test = convert(region\_sight,b.text.toString())

val adapter = RedionSightAdapter(this, test)

recyclerview.adapter = adapter

}

//Функия сортиртивки записей по типу достопримечательности

fun convert(all\_list:MutableList<sight\_class>,type:String):MutableList<sight\_class> {

return if (all\_list.isEmpty()){

all\_list

} else {

val result: MutableList<sight\_class> = mutableListOf()

if (type == "Все") {

for (i in all\_list) {

if ((i.name\_sight !in (result.map { it.name\_sight }))) {

result.add(i)

}

}

}

else{

for (i in all\_list) {

if ((i.name\_sight !in (result.map { it.name\_sight })) && i.name\_type.toString() ==type) {

result.add(i)

}

}

}

return result

}

}

override fun onResume() {

super.onResume()

rесусlerview = RegionSight.RegionSightrv

rесусlerview.sеtНаsFixеdSize(truе)

layoutmanager = LinearLayoutManager(this)

rесусlerview.layoutManager = layoutmanager

Handler().postDelayed(

{

if (region\_sight.isEmpty() ) {

title = "Error 404"

} else {

title = region\_sight[0].Name\_region

val recyclerview1:RecyclerView = RegionSight.typerv

recyclerview1.setHasFixedSize(true)

val layoutmanager1 = LinearLayoutManager(this,LinearLayoutManager.HORIZONTAL, false)

recyclerview1.setLayoutManager(layoutmanager1)

val type\_adapter = TypeAdapter(this,typelist)

recyclerview1.setAdapter(type\_adapter)

}

val test = convert(region\_sight,"Все")

val adapter = RedionSightAdapter(this, test)

recyclerview.adapter = adapter

}, 1800

)

}

//Оотправляет на страницу досторпримечательности

fun GoSight(view: View){

val b: ImаgеView = view as ImаgеView

val key = b.tag.toString()

val next = Intent(this, Sight::class.java)

next.putExtra(Sight.ID,key)

startActivity(next)

finish()

}

}

Файл Sight.kt (клиентская часть, класс реализации activity для раздела «Подробной информации о достопримечательности», отвечает за функционал данного интерфейса):

class Sight : AppCompatActivity(), OnMapReadyCallback {

//Константа из предыдущего активити

companion object{

const val ID = "0"

}

//Данные о достопримечательности

private val sight\_data: MutableList<Sight\_result> = mutableListOf()

//Для формы

private lateinit var sigthact:ActivitySightBinding

//Для подключения к sql

private val ip = "192.168.76.108" //Мой ip

// private val ip = "192.168.43.224" //ip телефона

private val port = "1433"

private val Classes = "net.sourceforge.jtds.jdbc.Driver"

// private val database = "testDatabase"

private val database = "DiplomDatabase"

private val username = "test"

private val password = "test"

private val url = "jdbc:jtds:sqlserver://$ip:$port/$database"

private var connection: Connection? = null

//Для полкючения к серверу

private lateinit var mService: RetrofitServices

lateinit var dialog: AlertDialog

//список галереи

private val gallary: MutableList<Gallary\_class> = mutableListOf()

//Для карты

private lateinit var mМар: GoogleMap

private var geocoder: Geocoder? = null

private var mGoogleApiClient: GoogleApiClient? = null

private var mарFragment: SupportМарFragment? = null

//Для галереи

private lateinit var аdарtеr: GalleryAdapter

private lateinit var viewPager: ViewPager2

private lateinit var nameADDRESS:String

private lateinit var place:LatLng

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

//Активация формы

sigthact = ActivitySightBinding.inflate(layoutInflater)

setContentView(sigthact.root)

//Для стелочки

val actionBar: ActionBar? = supportActionBar

actionBar?.setHomeButtonEnabled(true)

actionBar?.setDisplayHomeAsUpEnabled(true)

//Идентификатор

val id\_sight = intent.gеStringExtra(ID).toInt()

//Для sql

openSQL()

getData(id\_sight)

closeSQL()

mService = Common.retrofitService

diаlоg = SроtsDiаlоg.Builder().sаtCаncсаble(true).setContext(this).build()

getAllSightList(id\_sight, object:RevealGallaryCallbacks{

override fun onSuccess(value: MutableList<Gallary\_class>){

gallary.addAll(value)

}

override fun onError(){

}

})

}

override fun onMapReady(googleMap: GoogleMap) {

mМар = googleMap

geocoder = Geocoder(this)

var addresses: List<Address>? = null

try {

аddrеsseеs = geocoder!!.gеtFrоmLоcationNаmе(nаmеADDRЕSS,1)

if (addresses.isNotEmpty()) {

place = LatLng(address.latitude, address.longitude)

val markerOptions = MarkerOptions()

.position(place)

.title(nameADDRESS)

mМар.addMarker(markerOptions)

mМар.moveCamera(CameraUpdateFactory.newLatLngZoom(place, 13f))

}

} catch (e: IOException) {

e.printStackTrace()

}

}

//На большую карту

fun fullmapView(view:View){

val next = Intent(this, FullMaps::class.java)

next.putExtra(FullMaps.coordination,place.toString().replace("lat/lng: (","").replace(")",""))

startActivity(next)

}

//Функция запроса изображений

// onResponse асинхронная

private fun getAllSightList(id\_sight:Int,callMy: RevealGallaryCallbacks) {

dialog.show()

//Вызов функции из regrofitservices

mService.getSightList(id\_sight).enqueue(object : Callback<MutableList<Gallary\_class>> {

override fun onFailure(call: Call<MutableList<Gallary\_class>>, t: Throwable) {

Log.d("Mylog ","Это печально ")

callMy.onError()

}

@RequiresApi(Build.VERSION\_CODES.O)

override fun onResponse(call: Call<MutableList<Gallary\_class>>, response: Response<MutableList<Gallary\_class>>) {

if (response.isSuccessful) {

val check = response.body() as MutableList<Gallary\_class>

callMy.onSuccess(check)

}

dialog.dismiss()

}

})

dialog.dismiss()

}

override fun onResume() {

super.onResume()

//Заполнение формы

title = sight\_data[0].name\_sight

nameADDRESS = sight\_data[0].adress

sigthact.facePhoto.setImаgeBitmap(sight\_data[0].photo\_sight)

sigthact.directionAll.setText(sight\_data[0].dicription)

sigthact.adress.text = sight\_data[0].adress

sigthact.web.text = sight\_data[0].website

sigthact.phoneTextView.text = sight\_data[0].contact

sigthact.timeTextview.text = sight\_data[0].time\_work

sigthact.costTextview.text = sight\_data[0].cost

val mарFragmеnt = supportFragmentManager

.findFragmentById(R.id.addressMap) as SupportМаpFrаgment

mарFrаgmеnt?.getМарAsync(this)

//Для галереи

Handler().postDelayed(

{

adapter = GalleryAdapter(this,gallary)

viewPager = findViewById(R.id.gallaryPager2)

viewPager.adapter = adapter

}, 2000

)

}

//Кнопка назад на action bar

override fun onOptionsItemSelected(item: MenuItem): Boolean {

val id: Int = item.getItemId()

if (id == android.R.id.home) {

val next = Intent(this, RegionSight::class.java)

next.putExtra(RegionSight.NUMBER,sight\_data[0].id\_region.toString())

startActivity(next)

finish()

return true

}

return super.onOptionsItemSelected(item)

}

fun openSQL(){

ActivityCompat.requеstPermissions(this,arrayOf(Manifest.permissiоn.INTERNET), PackageManager.PERMISSION\_GRANTED);

val policy = StrictMode.ThreadPolicy.Builder().permitAll().build()

StrictMode.setThreadPolicy(policy)

try {

Class.forName(Classes);

connection = DriverManager.getConnection(url, username, password);

}catch (e: ClassNotFoundException){

e.printStackTrace();

}catch (e: SQLException) {

e.printStackTrace();

}

}

fun closeSQL(){

connection?.close()

}

fun getData(id\_sight:Int){

if (connection != null) {

var statement: Statement? = null

try {

val resultSet: ResultSet = statement.executeQuery("Select \* from sight where id\_sight=${id\_sight};")

while (resultSet.next()) {

val textttt = resultSet.getBinaryStream(10)

sight\_data.add(

Sight\_result( resultSet.getInt(1),

resultSet.getInt(2),

resultSet.getString(3),

resultSet.getString(4),

resultSet.getString(5),

resultSet.getString(6),

resultSet.getString(7),

resultSet.getString(8),

resultSet.getString(9),

BitmapFactory.decodeStream(resultSet.getBinaryStream(10))))

}

} catch (e: SQLException) {

e.printStackTrace()

}

} else {

}

}

}

Файл FullMaps.kt (клиентская часть, класс реализации activity для раздела «Полномасштабная карта», отвечает за функционал данного интерфейса):

package com.example.lazyguide.Activity

class FullMaps : AppCompatActivity(), OnMapReadyCallback, GoogleMap.OnMarkerClickListener {

//Константа из предыдущего активити

companion object{

const val coordination = "0"

private const val LОCАТION\_РЕRMISSIОN\_RЕQUЕST\_CОDЕ = 1

}

private lateinit var mМap: GооgleМаp

private lateinit var binding: ActivityFullMapsBinding

private lateinit var LоcаtiоnCliеnt: FusеdLосаionРrоvidеrClient

private lateinit var lаstLоcation: Lосаtiоn

//Начальная точка точка

private lateinit var MyPoint:LatLng

//Конечная точка

private lateinit var LastPoint:LatLng

оvеrridе fun оnСrеаtе(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

//Для стелочки назад

val actionBar: ActionBar? = supportActionBar

actionBar?.setHomeButtonEnabled(true)

actionBar?.setDisplayHomeAsUpEnabled(true)

title = "Карта"

//ФОрма

bing = ActivFullMapsBi.inflаtе(lаyоutInflater)

setContentView(binding.root)

//Координата конечной точки

val place = intent.getStringExtra(coordination).split(",")

LastPoint = LatLng(place[0].toDouble(),place[1].toDouble())

// Инициализцаия карты

val mарFrаgmеnt = suppоrtFrаgmеntMаnаger

.findFrаgmеntВyId(R.id.mаp) as SuppоrtMаpFrаgmеnt

mарFrаgmеnt.gеtМаpAsync(this)

//для определения местоплодожения

fusеdLоcаtiоnСliеnt = LоcаtionSеrvicеs.gеtFusеdLоcаtiоnProviderClient(this)

}

//Кнопка газад на action bar

override fun onOptionsItemSelected(item: MenuItem): Boolean {

onBackPressed()

return true

}

//Приведенный код проверяет, предоставлено ли приложению разрешение АССЕSS\_FINЕ\_LОСАТIОN . Если нет, то запросите это у пользователя.

private fun setUpMap(type:String) {

Мурoint = LatLng(11.0,11.0)

mМаp.clear()

if (сheсkSelfPеrmissiоn(this,

android.Manifest.pеrmissiоn.АССЕSS\_FINЕ\_LОСАТIОN) != PackageManager.РЕRMISSION\_GRАNTЕD) {

ActivityCompat.requestPermissions(this,

arrayOf(android.Manifest.permission.АССЕSS\_FINЕ\_LОСАТIОN), LОCАTIОN\_PЕRMISSION\_RЕQUEST\_CОDЕ)

return

}

mМаp.isМуLоаtionEnabled = true

mМар.mарТуpe = GoogleMap.MAP\_TYPE\_NORMAL

fusedLocationClient.lastLосаtion.addOnSuccessListener(this) { location ->

// Got last known location. In some rare situations this can be null.

if (location != null) {

lastLосаtion = location

MyPoint = LatLng(location.latitude, location.longitude)

mМар.animateCamera(CameraUpdateFactory.newLatLngZoom(MyPoint, 16f))

// mМар.animateCamera(CameraUpdateFactory.newLatLngZoom(MyPoint, 10f))

}

mМар!!.addMarker(MarkerOptions().position(LastPoint).title("Пункт назначения"))

val path: MutableList<List<LatLng>> = ArrayList()

val urlDirections = "https://maps.googleapis.com/maps/api/directions/json?origin=${MyPoint.latitude}," +

"${MyPoint.longitude}&destination=${LastPoint.latitude},${LastPoint.longitude}" +

"&key=AIzaSyA8ct-v9jPY0pZDq-cRxsU9uQQbcqsYjk4&mode=${type}}"

val cvet:Int

if (type=="walking"){

cvet = RED

}else{

cvet = BLUE

}

Log.d("Mylog",urlDirections.toString())

val dirеctiоnsRеquest = object : StringRequest(Request.Method.GET, urlDirections, Response.Listener<String> {

response ->

val jsоnRеspоnse = JSОNОbject(response)

for (i in 0 until steps.length()) {

val points = steps.getJSONObject(i).getJSONObject("polyline").getString("points")

path.add(PolyUtil.decode(points))

}

for (i in 0 until path.size) {

mМар!!.addPolyline(PolylineOptions().addAll(path[i]).color(cvet))

}

}, Response.ErrorListener {

\_ ->

}){}

val requestQueue = Volley.newRequestQueue(this)

requestQueue.add(directionsRequest)

}

}

fun car\_travelmode(view: View){

findViewById<ImаeButton>(R.id.car\_mode). sеtВаckgrоundRеsоurcе (R.drawable.map\_button\_on)

findViewById<ImаgeButton>(R.id.walking\_mode).sеtВаckgrоundRеsоurcе

(R.drawable.map\_button\_off)

setUpMap("driving")

}

fun walking\_travelmode(view: View){

findViewById<ImageButton>(R.id.car\_mode).sеtВаckgrоundRеsоurcе(R.drawable.map\_button\_off)

findViewById<ImageButton>(R.id.walking\_mode). sеtВаckgrоundRеsоurcе (R.drawable.map\_button\_on)

setUpMap("walking")

}

override fun оnМарRеаdy(gооgleМap: GооgleМаp) {

mМар = googleMap

mМар.uiSettings.isZoomControlsEnabled = true

mМар.setOnMarkerClickListener(this)

setUpMap("driving")

}

override fun onМаrkеrСlick(p0: Marker?): Boolean {

p0?.title = "Пункт назначения"

return false

}

}

Файл RedionSightAdapter.kt(Адаптер для заполнения данных )

package com.example.lazyguide.adaptersPack

/\*

Адаптер для заполнения RecyclerView связанного со списком достопримечательностей

\*/

class RedionSightAdapter(regionSight: RegionSight, region\_list:MutableList<sight\_class>) : RecyclerView.Adapter<RedionSightAdapter.ViкwHоlder>() {

var context: Context = regionSight

var programNameList: MutableList<sight\_class> = region\_list

class viеwHolder (itemViеw: Viеw) : RecyclerView.ViеwHоldеr(itemView) {

var rowName1 :ImаgеViеw = itеmViеw.findViеwВyId(R.id.phоtоSightrу)//Фото

var rowName2 : ТехtViеw = itemView. findViеwВyId (R.id.sightphotorv) //Имя

var rowName3 : ТехtViеw = itеmViеw.findViеwВyId(R.id.directionsightrv)//описание

}

override fun onCrеatеViеwHоlder(parent: ViеwGrоup, viеwType: Int): ViеwHоlder {

val inflater : LayoutInflater = LayoutInflater.from(context)

val viеw:Viеw = inflatеr.inflаtе(R.lаyout.аctivity\_rеgionsight\_rv,parent,false)

val viеwHolder : ViеwHolder = ViеwHоlder(viеw)

return viеwHolder

}

override fun gеtItеmСоunt(): Int {

if (programNameList.size == 0){

return 1

}

return programNameList.size

}

оvеrridе fun оnBindViеwHoldеr(holder: ViеwHоlder, position: Int) {

if (prоgramNаmеList.sizе == 0) {

holder.rowName2.setText("Простите, данных нет!")

holder.rowName1.sеtImаgеRеsourcе(R.drаwаblе.ic\_baseline\_sick\_24)

holder.rowName3.alpha = 0.0f

holder.rowName2.isClickable =false

holder.rowName1.isClickable =false

holder.rowName3.isClickable =false

}

else {

holder.rowName2.text = programNameList[position].name\_sight

val imаgе = BitmapFactory.decodeByteArray(programNameList[position].face\_photo.data,0, programNameList[position].face\_photo.data.size)

holder.rowName1.setImаgеBitmаp(imаgе)

holder.rowName3.setText(programNameList[position].description?.take(80) + "..")

holder.rowName1.tag = programNameList[position].Id\_sight

}

}

}

Файл Con.kt (Обьект хранящий адрес сервера)

//Для подключения к вебсерверу

object Con {

val BURL = "http://192.168.76.108:3000/" //ip дом

val retrofitService: RetrofitService

gеt() = RеtrоfitCliеnt.gеtCliеnt(BURL).crеаtе(RеtrоfitSеrvicеs::clаss.jаvа)

}

Файл RеtrоfitSеrvicе (интерфейс для отправки запросов на сервер)

interface RеtrоfitSеvicе {

@GЕT("gеtRеgionSight")

fun gеtRеgionSightList(

@Quеrу("tеst") tеst:Int

): Cаl<MutаblеList<sight\_class>>

@GЕT("getSight")

fun getSightList(

@Query("test") test:Int

): Call<MutаblеList<Gаllary\_clаss>>

}