Санкт-Петербургский государственный университет

ЯДРИХИНСКАЯ Юлия Сергеевна

Выпускная квалификационная работа

Разработка прототипа мобильного приложения по построению пешеходных маршрутов с пользовательскими параметрами

Уровень образования: магистратура

Направление 05.04.03 Картография и геоинформатика

Основная образовательная программа ВМ.5523 Геоинформационное картографирование

Научный руководитель:

доцент,

кафедра картографии и геоинформатики,

кандидат технических наук,

Паниди Евгений Александрович

Рецензент:

директор по образовательным программам,

ООО «НекстГИС»,

Казаков Эдуард Эдуардович

Санкт-Петербург

2021

Содержание

Введение

В настоящее время число людей, предпочитающих пешеходный или велосипедный виды передвижения, становится все больше. Многие испытают потребность в проведении большего времени на улице, вдалеке от шума автомобилей. Известен факт, что продолжительные прогулки улучшают не только физическое, но и эмоциональное здоровье (Butler, 2011). Для этого маршрут должен удовлетворять потребности человека. Например, для туриста или жителя города, мало знакомого с окрестностями, подходящей бы стала прогулка через достопримечательности города. Она смогла бы стать более комфортной в сочетании с велосипедным движением. Для других было бы привлекательным пройти через озелененные места города, услышать звук природы, побыть вдалеке от шума города.

Каждый человек имеет свои собственные предпочтения. Для их реализации нужна подробная карта, решающая задачи передвижения по городу с максимальным комфортом для каждого. Карта должна выполнять функцию по выбору вида маршрута, анализировать несколько построенных прогулок и выбирать лучшую из них. Она должна реализовывать функции информативного характера, а именно: предоставлять сведения об объектах вдоль маршрута, оценивать время и длину прогулки, отображать местоположение человека в любой момент времени. Необходимо сделать ее удобной в пользовании на улице с возможностью доступа к ней максимального количества людей. К сожалению, функционирующие сервисы не сочетают в себе все эти характеристики. Они имеют свои ограничения. Например, отсутствие возможности задать время для прогулки, построить несколько видов маршрутов в одном приложении или просмотреть встречающиеся по пути интересные объекты с описанием и иллюстрациями.

Целью данной работы являлось создание проекта мобильного приложения, предоставляющее функцию построения пешеходных прогулок с пользовательскими параметрами, такими как: выбор и анализ вида пешеходного маршрута, обзор кофеен, пунктов проката велосипедов и самокатов, достопримечательностей города по маршруту с подробной информацией о них.

Для достижения поставленной цели необходимо было решить следующие задачи:

– рассмотреть существующие исследования и сервисы, связанные с построением пешеходных прогулок;

– выбрать и проанализировать исходные данные проекта;

– исследовать пешеходные маршруты, построенные по разным источникам данных;

– выбрать алгоритмы для реализации функций в мобильном приложении;

– разработать функционирующее мобильное приложение, основанное на клиент-серверной архитектуре.

Взаимодействие пользователя с приложением осуществляется с помощью устройства с установленной на нем операционной системой Android. Исходными данными для построения пешеходных маршрутов послужили материалы выпускной квалификационной работы, которая выполнялась в 2019 году; данные об объектах, полученные с помощью запросов к социальной сети «ВКонтакте», поисково-картографической службе «Яндекс.Карты», веб-ресурс Wikipedia. Тестовая территория проекта находится в городе Санкт-Петербург и расположена следующим образом: южная граница проходит вдоль Невского проспекта, северо-западная и северо-восточная границы – набережные рек Нева и Фонтанка соответственно.

Глава 1. Анализ существующих исследований по построению пешеходных прогулок

1.1 Существующие алгоритмы и сервисы, реализующие функцию построения пешеходных маршрутов

Существует множество фактов (Blair, 2009), свидетельствующих о том, что повседневные пешие прогулки оказывают положительные влияния на здоровье человека (Ewing, et. al., 2003), транспортную обстановку в городе (Ribeiro, Hoffimann, 2018, Saelens, et. al., 2003), качество воздуха (Debyser, 2014) и даже на экономику (Litman, 2003). Если в городе развита соответствующая инфраструктура (Handy, 2002), то пешеходные прогулки можно чередовать с велосипедными. Вместе они рассматриваются как способ добраться до работы (Oja, Vuori, Paronen, 1998) или как ежедневное средство передвижения по городу (Sallis, Owen, Fisher, 2008).

Было проведено значительное количество исследований, чтобы определить причины, побуждающие людей совершать пешеходные прогулки (Pikora, et. al., 2002, 2003, Forsyth, 2015). Исследователи приводят следующие факторы, создающие благоприятную окружающую среду для прогулок: безопасность, эстетика тротуаров, плотность населения, близость мест назначения и улично-дорожная сеть (Ewing, Handy, 2009, Forsuth, 2015, Lo, 2009, Vale, Saraiva, Pereira, 2015). Приведенные факторы были применены в разных комбинациях для разработки индекса (Frank, et. al., 2010, Forsyth, et. al., 2006, Habibian, Hosseinzadeh, 2018, Hall, Ram, 2018, Krambeck, 2006), определяющий среду для прогулок в рассматриваемом городе (далее – индекс) (Giles-Corti, et. al., 2014). В исследованиях (Kelly, et. al., 2011, Maghelal, Capp, 2011) для вычисления индекса обращалось внимание на факторы, связанные с комфортом пешеходов: шум и тень. Напротив, в работах (Brownson, et. al., 2009, Dannenberg, Cramer, Gibson, 2004) эти факторы не учитывались, возможно, из-за недостаточности исходных данных и вычислительных средств.

Стоит заметить, что рассчитываемые индексы не всегда учитывают весовые значения факторов, влияющих на пешеходные прогулки. Например, в работе (Gullón, et al., 2017) все факторы имеют равный вес. В другом исследовании, проведенном (Frank, et al., 2010), уличная сеть имеет вес, в два раза превышающий остальные факторы. В работе (Tsiompras, Photis, 2017) веса были присвоены на основе онлайн-опроса.

В некоторых исследованиях в качестве исходных материалов используются полевые данные или результаты опросов. Такой подход требует огромных затрат времени и ресурсов для получения подробных результатов. В других работах, напротив,

авторы предлагают жертвовать уровнем детализации в пользу более доступного способа получения индекса на основе вторичных данных (Kikuchi, et. al., 2018, Mayne, et. al., 2019).

Разработанная система пешеходных прогулок, описанная Ясуфуми Такамака, имеет своей целью улучшение здоровья человека (Yasufumi, Sasaki et.al., 2017). Авторы статьи предлагают строить пешеходные маршруты, учитывая указанные пользователем калории, которые он хочет потратить по пути следования до конечной точки.

Была разработана система, учитывающая уличное освещение для построения пешеходных маршрутов (Miura, Takeshima, et al., 2011). Замечено, что в своих работах исследователи все больше обращают внимание на ночное освещение городских улиц при разработке алгоритмов для построения пешеходных прогулок, потому что, во-первых, улицы без освещения могут быть опасными; во-вторых, красивые ночные улицы городов всегда привлекали туристов (Guo, et al., 2011).

Еще одна теория построения пешеходных маршрутов была разработана китайскими исследователями (Zhong, Chen, et al., 2017). Их идея основана на том, что пользователь может совмещать передвижения и пешком, и с использованием велосипедного транспорта. Авторы предлагают включать в маршрут две точки расположения велопарковочных станций, с учетом которых маршрут разделяется на три части: пешеходная (от начальной точки до велопарковочной станции), велосипедная (между двумя станциями) и пешеходная (от второй велостанции до конечной точки маршрута). Помимо этого, пользователь имеет возможность выбирать три разновидности будущего маршрута, основанные на алгоритме Дейкстры: кратчайший, самый безопасный (алгоритм учитывает число фактов нарушения правопорядка между крайними точками пути) и оптимальный (с учетом двух предыдущих условий). Такой подход для движения по улицам города применим только для крупных населенных пунктов, где достаточно развита сеть велопарковочных станций, расположенных равномерно по всему городу.

Популярные сервисы, такие как «Instagram» и «ВКонтакте», предоставляют возможность анализировать огромный набор данных о фотографиях. Учитывая это, были обнаружены точки притяжения туристов в городе Санкт-Петербург на основе данных «Instagram», которые также могут использоваться для построения пешеходных маршрутов (Mukhina, Rakitin, et al., 2017). Существует проект сервиса по созданию самых красивых прогулок в городе Санкт-Петербург. Алгоритм будущего сервиса разработан с учетом данных о фотографиях, опубликованных в социальной сети ВКонтакте и имеющих географические координаты (Semenov, 2018). Другие

исследователи тоже использовали фотографии пользователей социальных сетей с известными координатами. Они разработали алгоритм для пешеходов, который позволяет создавать самый красивый, тихий и улучшающий эмоциональное состояние человека пешеходные маршруты (Quercia, Rossano, et al., 2014).

Ниже приведена резюмирующая таблица факторов, которые учитываются для построения пешеходных маршрутов в различных исследованиях.

Автор	Страна	Факторы								
		Плотность населения	Вид деятельности	Уличная сеть	Шум	Тень	Красота мест	Уличное освещение	Безопасность	Другие
Al Shammas, Escobar, 2019	Испания	да	да	да	да	да				
Habiban, Hosseinzadeh, 2018	Иран	да	да	да						близость пункта назначения
Semenov, 2018	Китай				да		да			архитектура зданий, озелененные территории, гидрография, промзоны
Zhong, Chen, et al., 2017	Китай		да							велосипедное передвижение
Mukhina, Rakitin, et al., 2017	Россия						да			
Yasufumi, Sasaki et.al., 2017	Китай									калории
Gullón, et al., 2017	Испания	да		да						Близость магазинов

Quercia, Rossano, et al., 2014	Великобритания				да		да			эмоциональное состояние человека
Giles-Corti, et al., 2014	Австралия	да	да	да						
Coffee, et al., 2013	Австралия		да	да						сеть торговых точек, плотность застройки
Miura, Takeshima, et al., 2011	Япония						да			
Guo, et al., 2011	Китай						да	да	да	
Freeman, et al., 2010	США	да	да	да						плотности станций метро и торговых площадей
Glazier, et al., 2012	Канада	да	да	да						плотность застройки и расположение торговых точек в 10-ти минутной доступности
Frank, et al., 2010	США	да	да	да						торговые площади

Zhu, Lee, 2008	США	да	да	да		от деревьев			да	потенциальные пешеходы, городская инфраструктура, красота мест
Leslie, et al., 2007	Австралия		да	да						сеть торговых точек, плотность застройки
Pikora, et al., 2006	Австралия								да	Инфраструктура города, красота мест, климатические условия
Dannenberg, Cramer, Gibson, 2004	США					да				дорожная обстановка, инфраструктура, красота мест

На сегодняшний день известно несколько действующих приложений для построения городских пешеходных маршрутов с учетом различных факторов. Алгоритм, учитывающий факторы окружающей среды, был разработан в компании «Urbica»[1]. Созданное приложение под названием «Walkstreets» (Рис.1) предлагает строить самые экологичные, тихие, проходящие через зоны с наилучшими показателями чистоты воздуха маршруты. Одним из важных отличий от других приложений является способность навигатора предлагать пешеходный маршрут до произвольной точки назначения. Это очень полезно, если пользователь просто хочет прогуляться по приятным местам и не заботиться о пункте назначения. Компания контролирует качество воздуха и городской шум, а также использует спутниковые снимки для обнаружения озелененных зон, что учитывается при построении маршрутов. Данное приложение работает только на территории города Москва. Отсутствует возможность просмотра информации об интересных объектах вдоль построенного маршрута.

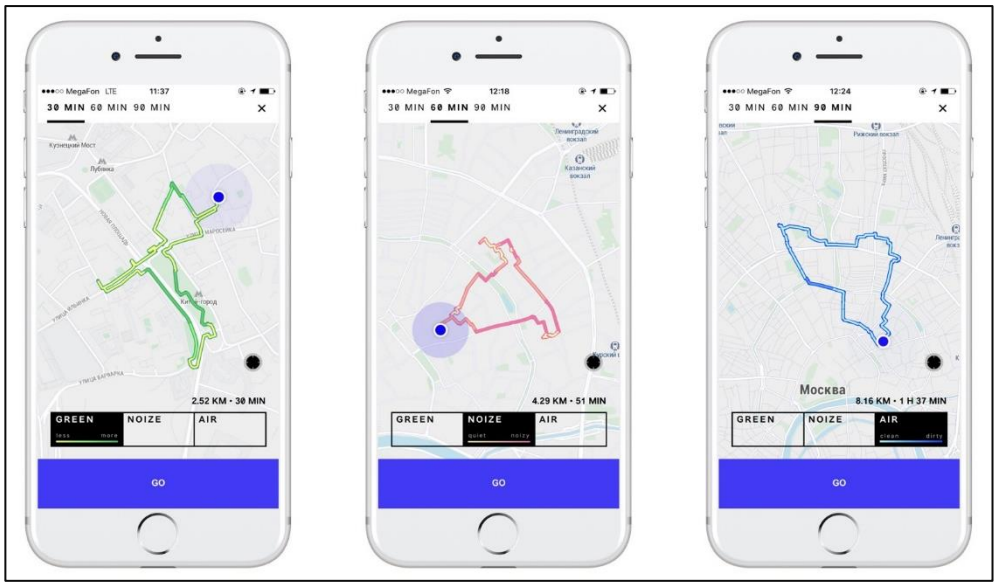


Рис.1. Мобильное приложение Walkstreets. Источник: https://medium.com/@Urbica.co/walkstreets-34e237237607

Сервисы, которые удовлетворяют требованию построения туристических пешеходных маршрутов, проходящих через достопримечательности Санкт-Петербурга, называются «Wander» (мобильное приложение) и «Travelpath» (версия, запускаемая из браузера)[2] (Рис.2). По мнению создателя проекта, в базах данных (далее – БД) обеих версий собраны наиболее популярные музеи, парки, памятники и церкви, через которые строятся пешеходные маршруты. В сервисах есть возможность выбрать опцию для прохождения по отдельности через популярные музеи, парки, памятники, церкви, или выбрать одновременно все пункты для их учета при построении маршрута. Пользователь

[1] https://medium.com/@urbica/walkstreets-5a41b22ae104
[2] http://travelpath.ru/

может посмотреть подробную информацию об этих объектах. В версии, запускаемой из браузера, возможно задание только крайних остановок планируемого маршрута. После его построения не отображается информация о длине и времени прохождения маршрута. Прогулки строятся по принципу кратчайшего следования.

Что касается мобильной версии «Wander» (Рис. 3), то она доступна только для пользователей платформы Android. В ней предоставлено больше опций, чем в версии, запускаемой из браузера. В добавлении к ним, есть информация о количестве точек (достопримечательностей), времени и длине маршрута.

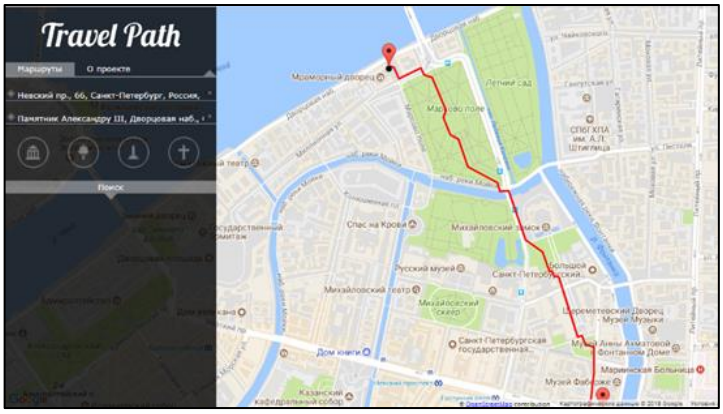


Рис.2. Сервис Travel Path.

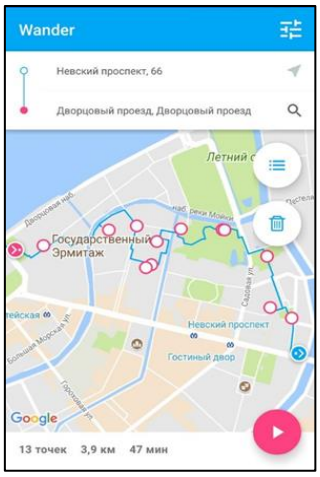


Рис.3. Приложение "Wander".

Реализованный алгоритм в приложениях «Wander» и «Travelpath» учитывает только достопримечательности, локализованные в точке. К недостаткам приложения можно отнести то, что при построении пешеходных маршрутов не учитываются расположения набережных города, которые также могут заинтересовать туристов. Имеются неудобства в использовании, такие как невозможность просмотреть длину и время на прохождение построенного маршрута (для браузерной версии). Отсутствует возможность выбрать дополнительные параметры маршрута, например, указание желаемого времени прохождения маршрута и просмотр местоположений кофеен по пути следования. Последняя опция была бы полезна для пользователей, планирующих продолжительные прогулки. В версии, запускаемой из браузера после построения не отображаются объекты по пути маршрута, указанные в меню программы (парки, музеи и так далее). Это реализовано в мобильной версии, но тип объекта сложно понять, не нажав на него – условные обозначения музеев, парков, церквей и памятников одинаковые.

Maps.me – это свободно распространяемое мобильное приложение[3], которое доступно на платформах iOS, Android (Рис.4). Картографическая основа приложения –

[3] https://maps.me/

данные OpenStreetMap. Отличительной особенностью является возможность работы сервиса в режиме офлайн с предоставление всего числа функций. Перед началом использования приложения происходит загрузка карты местности в память устройства. Доступны функции ориентирования по GPS, голосовой навигации, построения маршрутов для автомобилей и велосипедистов, поиск по объектам, экспорта и импорта данных в формате KML, а также предоставления подробной информации об объектах на карте. Кроме прочих функций, приложение предоставляет услугу по построению пешеходных прогулок между двумя заданными точками. В январе 2019 года Maps.me объявил о запуске платных путеводителей, доступны несколько путеводителей без взимания платы для наиболее популярных городов во всем мире. Маршруты для приложения предоставляются партнёрами сервиса. В Maps.me существует функция для работы с метками. Их можно группировать, выбирать для каждой метки цвет булавки, делиться ими с друзьями, экспортировать в виде KMZ-файла. Отображение меток можно отключать – каждую индивидуально, по группам или все сразу, либо отображать все метки на карте. Это очень удобная функция для туристов, кто приезжают в город на непродолжительное время и планируют свое путешествие заранее. Также есть возможность сохранения трека путешествия. Отсутствует возможность выбора разных видов маршрутов, реализован только кратчайший без возможности построить прогулку через большее количество достопримечательностей, парков и так далее.

Таким образом, приложение является одни из сервисов для туристов, которые предпочитают исследование города самостоятельно вместо организованной экскурсии с гидом.

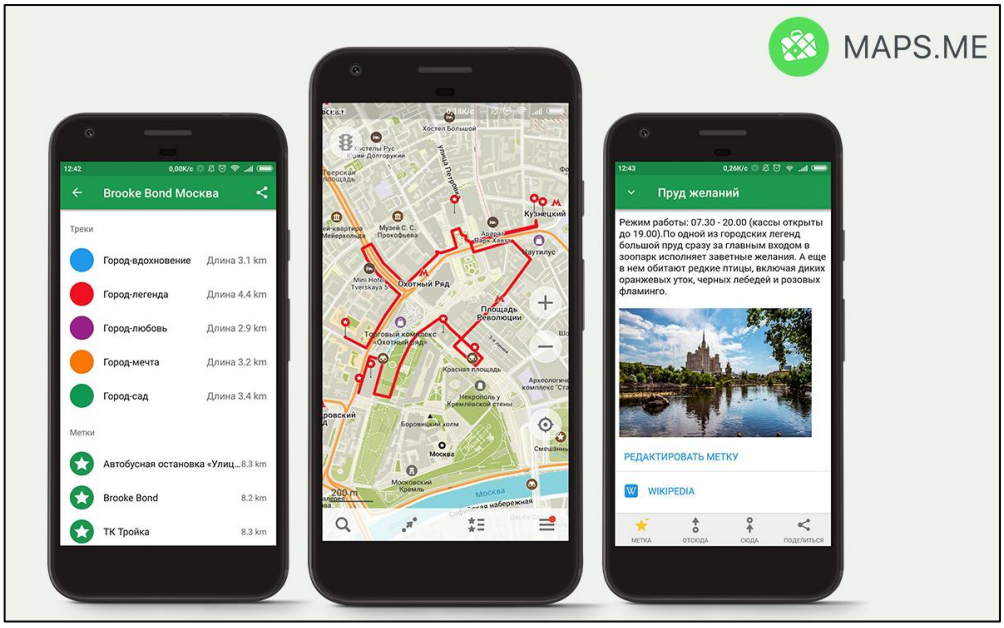


Рис.4. Сервис Maps.me. Источник: https://3dnews.ru/968635.

На один из районов города Санкт-Петербург разработана «Интерактивная карта Красногвардейского района»[4] в формате картографического веб-приложения. На сайте карты содержится раздел «Маршруты», в котором представлены 10 пешеходных и 2 велосипедных тематических прогулок, например, «Старая Ржевка», «От малиновки до Исаковки» и так далее. Любой маршрут состоит из остановок, которые отображаются на карте сервиса (Рис.5), доступны описания каждой остановки (Рис.6), информация времени прохождения и длине маршрута. Предоставлена опция просмотра панорам улиц. Есть функция «Сформировать тур», с помощью которой можно создать собственный маршрут, добавив в него предложенные сервисом остановки. Навигация по маршруту с помощью GPS отсутствует.

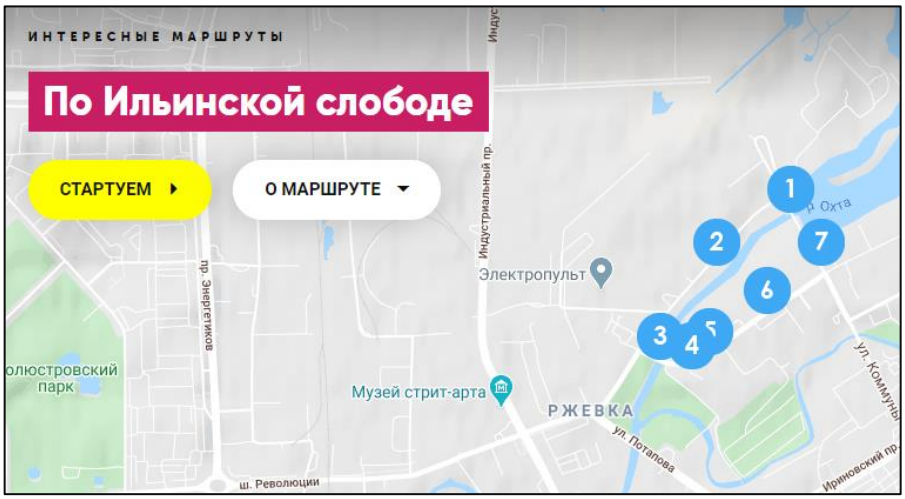


Рис.5. Отображение пешеходного маршрута в окне интерактивной карты.

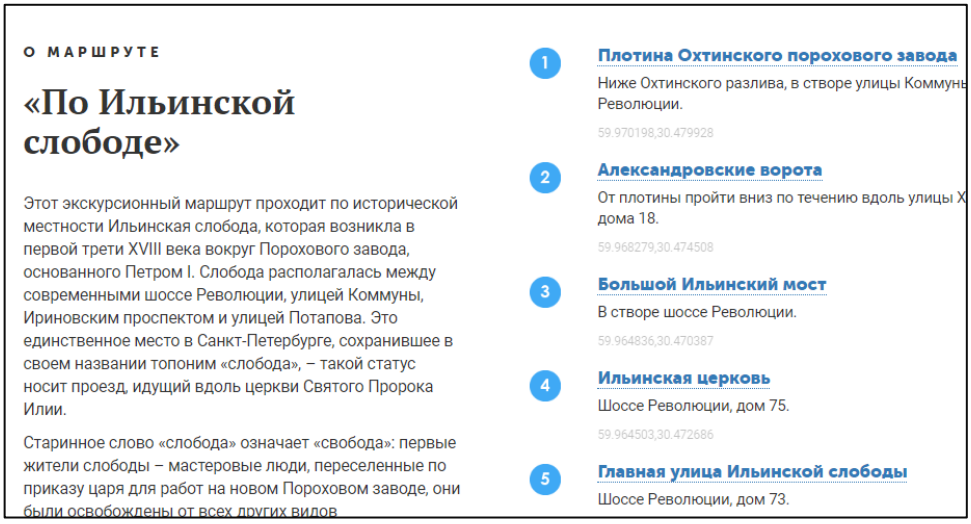


Рис.6. Описание пешеходного маршрута.

Таким образом, существующие приложения для построения пешеходных прогулок только по отдельности позволяют строить маршруты с учетом факторов окружающей среды и с прохождением через достопримечательности города.

[4] http://krasnakarta.ru/

Отсутствует единый сервис, позволяющий строить разные виды маршрутов на территории Санкт-Петербурга. Имеются ограничения в выборе дополнительных параметров при построении маршрутов. Возникают неудобства при пользовании интерфейсом приложений с целью просмотра объектов вдоль построенного маршрута.

1.2. Мобильное приложение как инструмент планирования пешеходных маршрутов с учетом достопримечательностей и факторов окружающей среды

Ранее в рамках бакалаврской выпускной работы был создан пилот-проект клиент-серверного веб-приложения с использованием свободно распространяемого программного обеспечения, которое предоставляет функцию построения пешеходных маршрутов для прогулок в городе Санкт-Петербург, а именно: «самый интересный» маршрут – проходящий через популярные достопримечательности; «самый тихий» – строится с учетом распространения шумового загрязнения; «самый экологичный» – через озелененные территории; «самый быстрый» – кратчайший маршрут.

«Самый интересный» маршрут – определяется с учетом популярности памятников, фонтанов, зданий с архитектурной и исторической ценностью, мостов, набережных города, площадей и озелененных территорий в общем рейтинге, составленном поисково-информационной картографической службой «Яндекс.Карты»[5]. Такой подход к оценке перечисленных объектов является объективным, поскольку определяется с учетом оценивания многочисленных пользователей «Яндекс.Карт». «Самый тихий маршрут» – создается на основе данных по «карте» шумового загрязнения на территорию проекта и проходил через наименее шумные участки местности (Mostafa Refat, 2014). «Самый экологичный маршрут» – при его создании учитываются расположения различных озелененных территорий (парков, садов, скверов и так далее) без градации их по популярности. Алгоритм при построении пешеходной прогулки учитывает близко расположенные озелененные территории между крайними точками маршрута. «Самый быстрый маршрут» – кратчайший маршрут между двумя заданными точками. В течение работы над проектом была создана БД, на основе которой была реализована функция построения пешеходных маршрутов на основе сетевой модели данных, отображения разных объектов вдоль них, оценки времени и длины прохождения построенного пути.

Работа над этим приложением дала необходимый практический опыт применения веб-технологий и функций БД для решения задач создания маршрутов.

[5] https://yandex.ru/maps

Были получены навыки программирования на языках JavaScript, HTML, Python. В тоже время ограничениями проекта являлись:

– реализация в формате веб-страницы, что ограничивает удобство пользования приложением;

– взаимодействие с веб-приложением возможна только на компьютере с установленной на нем БД, что приводит к недоступности приложения широкому кругу пользователей;

– отсутствие функции определения местоположения пользователя во время работы приложения;

– отсутствие подробной информации о достопримечательностях по маршруту, такой как текстовое описание, фотография, ссылка на веб-ресурс для получения большей информации о каждом объекте;

– невозможность поиска крайних точек маршрута по их адресу.

Отличие разрабатываемого прототипа приложения от предыдущей веб-версии и от созданных ранее заключается в том, что его реализация запланирована в формате мобильного приложения, работающего на базе операционной системы Android. Такой формат предоставит возможность удобного пользования приложением на улице во время прогулки. «Самый интересный» маршрут будет строится на основе обновленных данных из службы «Яндекс.Карты» и числа фотографий пользователей социальной сети «ВКонтакте». Задействование данных «ВКонтакте» позволит обновить оценки набережных по популярности, поскольку «Яндекс.Карты» такие данные больше не предоставляет. Предполагается сделать возможным просмотр текстового описания, фотографии и ссылки на веб-ресурс для каждого объекта вдоль построенного маршрута. Пользователь сможет отслеживать свое местоположение с помощью установленной на его устройстве функции GPS. Ему будет предоставлена возможность пользоваться тремя видами фоновыми карт, искать местоположения объектов на карте по введенному адресу, задействовать дополнительные элементы и режимы навигации по карте для лучшего ориентирования на местности при прохождении маршрута. Также будет доступна информация о пунктах проката велосипедов и самокатов в дополнении к информации о кофейнях.

1.3. Анализ и выбор программного обеспечения для разработки проекта

Для достижения поставленной цели необходимо было выбрать не только программное обеспечение для решения различных задач на протяжении всего процесса

реализации проекта, но и подходящие языки программирования, применяемые для создания алгоритмов при разработке мобильного приложения.

Весь процесс работы над проектом был разделен на два основных этапа: подготовка исходных данных и создание прототипа мобильного приложения. Это означает, что для каждого этапа необходимо выбрать программное обеспечение и другие инструменты обработки данных.

Существует множество геоинформационных программных продуктов, среди них наиболее популярными считаются: Quantum GIS (далее QGIS), ArcGIS Desktop (далее ArcGIS), GRASS GIS (далее GRASS GIS), MapInfo, «Панорама», «Нева». Последние три не подходили для решения задач проекта. Основные причины: ограниченный доступ для пользования этими программными продуктами, предоставление узкого спектра инструментов геообработки, отсутствие интеграции с созданной БД (ГИС «Нева») и возможности запуска геоалгоритмов, написанных на языке программирования. Несмотря на свободное распространение программного продукта GRASS GIS, его возможности также не были задействованы, поскольку существуют различные трудности при работе с векторными данными и имеются неудобства в пользовании, как у вышеописанных трех программ.

Для осуществления цепочки геообработки исходных данных и визуализации результатов анализа были задействованы две географические информационные системы (далее ГИС): QGIS и ArcMap (один из продуктов семейства ArcGIS).

QGIS является свободно распространяемой программой с открытым исходным кодом. Для решения задач проекта система была удобна при анализе, редактировании и визуализации как векторных, так и растровых данных. QGIS предоставляет возможность подключения к используемой в проекте БД (инструмент «Добавить слой PostGIS») и взаимодействия с ней: редактирование, просмотр и анализ данных, хранящихся в базе, импорт/экспорт таблиц (с использованием инструмента «DB Manager»). QGIS предоставил возможность визуализации построенных маршруты посредством запросов к БД, что являлось основной причиной выбора этой ГИС.

Инструменты не менее популярного семейства программных продуктов ArcGIS также были применены для решения определенных задач. В отличие от QGIS, ArcGIS не является программой с открытым исходным кодом, поэтому она не была выбрана в качестве основного геоинформационного программного обеспечения. Основная причина использования возможностей ArcGIS – удобство пользования некоторыми инструментами, а именно: пространственное соединение атрибутов векторных данных и вычисление евклидового расстояния на растре. Эти встроенные инструменты были

необходимы в процессе подготовки данных для построения всех видов маршрутов. Запуск созданного ранее инструмента, алгоритм которого был написан на языке программирования Python, также осуществлялся посредством возможностей ArcGIS. Этот инструмент требовался для решения задачи создания растровых данных на основе вычисления евклидового расстояния. Программный код запускался из-под ArcMap и результат по окончанию работы сразу можно было оценить в диалоговом окне программы.

Таким образом, обе ГИС дополняли друг друга как во время работы с исходными данными на первом этапе создания проекта. Более того, обе программы поддерживают общий формат векторных данных – *.shp*, что также было удобно при работе в обеих программах.

В процессе подготовки исходных данных создавались собственные алгоритмы для решения конкретных задач. Они были написан на языке Python. Применение возможностей языка Python требовалось для создания инструмента в ArcMap, где другой язык программирования не применим. Кроме того, возможности Python были задействованы при выгрузке данных с помощью «API поиска по организациям» компании «Яндекс» (см. раздел 1.4) и их последующем представлении в виде shape-файлов. Более того, Python применялся при выгрузке данных о фотографиях с помощью интерфейса прикладного программирования (далее – API) «ВКонтакте» и их последующем анализе.

Python – высокоуровневый интерпретируемый язык программирования с динамической типизацией (Лутц, 2011). Во время создания проекта были использованы следующие библиотеки языка Python:

– ogr – создание shape-файлов;

– fiona – удаление из shape-файла лишних выгруженных объектов;

– shapely – работа с shape-файлами.

Использованные модули языка:

– arcpy – запуск написанных алгоритмов в ArcMap;

– json – работа с одноименным форматом;

– os - работа с операционной системой;

– urllib – получение описания используемой проекции при создании shape-файла;

– re - работа с регулярными выражениями;

– shutil - обработка файлов;

– collections – работа со словарями.

В процессе работы над созданием прототипа мобильного приложения создавались его клиентская и серверная части (см. главу 3). Алгоритмы, написанные на языке Java, использовались для создания обеих частей. Java – объектно-ориентированный язык программирования, основанный на классах (Шилдт, 2015). Это один из самых популярных языков для создания клиент-серверных веб-приложений. Язык Python тоже используется для создания клиент-серверных приложений, но из-за неполноценного объектно-ориентированного подхода, который важен для разработки приложений, не пользуется популярностью среди разработчиков.

Основные преимущества Java перед другими языками программирования, определившие выбор именно этого языка:

– полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, в частности, на мобильном (Блинов, 2007);

– возможность создания приложения на устройстве с операционной системой Android;

– автоматическое управление памятью – язык Java автоматически распределяет память под объекты и освобождает ее, когда объекты больше не используются;

– реализация объектно-ориентированного подхода, что позволяет решать задачи по созданию гибких, масштабируемых приложений (Шилдт, 2015).

Во время серверной части были использованы следующие основные классы языка Java:

– Socket, ServerSocket – для соединения между клиентом (мобильным устройством) и сервером (серверным приложением);

– Connection – осуществляет соединение к БД;

– Statement – необходим для работы с запросами к БД;

– DriverManager – для установления соединения между серверным приложением и БД.

Серверная часть прототипа приложения состоит из серверного приложения и БД (PostgreSQL[6]). Разработка серверного приложения проводилась в программном обеспечении IntelliJ IDEA[7] (далее – ПО IntelliJ IDEA). Это свободно распространяемая интегрированная среда программирования для многих языков, в том числе и на Java. В числе отличительных особенностей, которые определили выбор этого ПО, выделяются:

[6] https://www.postgresql.org/about/
[7] https://www.jetbrains.com/ru-ru/idea/

автодополнение кода, набор инструментов для разработки Android-приложений, удобная навигация, отладчик кода.

PostgreSQL – система объектно-реляционной базы данных с открытым исходным кодом, которая использует и расширяет язык SQL в сочетании с множеством функций, обеспечивающих безопасное хранение и масштабирование данных. Основные причины использования этой СУБД являлись доступность в использовании, интеграция с QGIS и наличие возможности для работы с пространственными данными посредством PostGIS[8]. PostGIS – расширение системы объектно-реляционной БД PostgreSQL. PostGIS позволяет хранить пространственные объекты и поддерживает функции для их анализа и обработки. Например, СУБД PostGIS поддерживает расширение pgrouting[9], которое применялось для осуществления функций маршрутизации и другого сетевого анализа. Кроме дорожного графа для построения маршрутов, БД была необходима для хранения объектов, обозначенных вдоль любого построенного маршрута в мобильном приложении.

Для написания программного кода клиентской части приложения использовалась среда разработки Android Studio[10]. Эта одно из самых распространенных интегрированных сред разработки. Она предназначена для разработки приложений, функционирующих на всех типах устройств с операционной системой Android. В этой среде с использованием языка программирования Java была настроена связь между серверным приложением и мобильным устройством, разработан интерфейс приложения, его взаимодействие с пользователем.

Android Studio имеет характерные особенности, которые определили использование именно этого программного обеспечения при работе над созданием мобильного приложения:

– визуальное редактирование макетов окон приложения;

– возможность писать программный код и редактировать его;

– сборка приложения;

– генерация *.apk* файла – файл приложения для установки его на мобильное устройство;

– пользование встроенным анализатором кода, который позволяет находить ошибки в программном коде;

[8] https://postgis.net/
[9] https://pgrouting.org/
[10] https://developer.android.com/studio

– запуск эмулятора мобильного устройства, на котором тестировались функции приложения.

Проект приложения в Android Studio состоит из нескольких директорий, в которых расположено множество файлов для обеспечения работоспособности мобильного приложения (Рис.7).

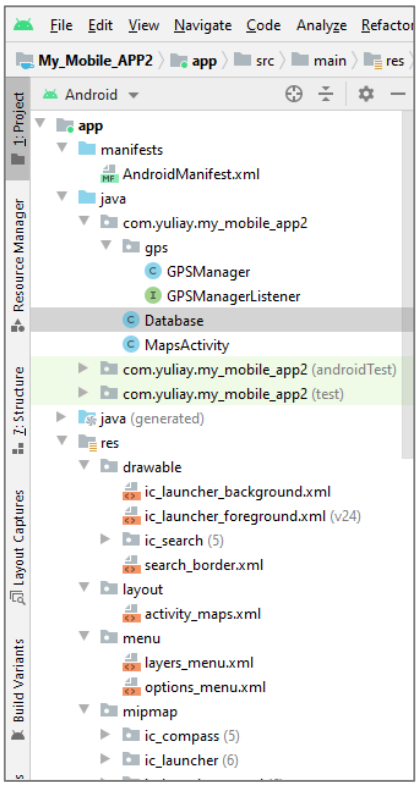


Рис.7. Дерево проекта в Android Studio.

При создании приложения к проекту были подключены зависимости (dependencies). Помимо стандартных зависимостей использовались такие как:

– com.google.android.gms:play-services-maps:17.0.0 – для работы с картами, которые используются в качестве фона в приложении (Google Map, Google Satellite, Google Hybrid);

– org.postgresql:postgresql:42.2.5.jre6 – драйвер JDBC для работы с БД.

Зависимости состоят из пакетов. Большое количество пакетов было импортировано в процессе разработки, основные из них:

– android – необходим для подключения геокодера и его работы при прямом и обратном геокодировании, настройки содержательной части окна приложения (кнопок, поисковых строк, всплывающего окна);

–com.google.android – осуществляет функции перемещения по карте, масштабирования, работы с маркерами на карте (точки местоположений пользователя);

– java.util – для работы со временем, словарями и массивами.

В результате анализа возможных инструментов для разработки проекта, были выбраны система управления БД PostGIS, установленная на сервере, ГИС, языки программирования Python и Java для подготовки исходных данных и создания мобильного приложения. Определена среда разработки приложения – Android Studio.

1.4. Источники исходных данных для построения пешеходных маршрутов

Основным источником исходных данных проекта является БД, созданная ранее при работе над версией приложения в формате веб. В ней содержатся граф дорог для построения прогулок и информация об объектах, отображаемых вдоль маршрута. В отношении достопримечательностей содержится название, координаты, рейтинговая оценка по популярности для каждого объекта. В БД есть информация для кофеен: их названия и координаты местоположения.

Для создания мобильной версии приложения БД была обновлена – добавлены достопримечательности, их описание, фотографии, ссылки на них в сети Интернет, обновлены их рейтинговые оценки. Добавлена информация о местоположении пунктов проката велосипедов и самокатов. Обновлена атрибутивная информация графа дорог – весовые значения для каждого элемента графа (ребра). Эти весовые значения применяются для построения четырех видов прогулок.

Обновленная информация о достопримечательностях (координаты, названий, рейтинговые оценки) и пунктах проката велосипедов и самокатов была получена из поисково-информационной картографической службы «Яндекс.Карты» при помощи API-поиска по организациям[11]. Поиск данных выполнялся по тексту поискового запроса – прямой поиск. Результаты запроса возвращались в формате *.json*.

Формат запроса. Начало запроса: *'https://searchmaps.yandex.ru//v1//*. Далее указывались параметры запроса (под номерами 1,2,4 являлись обязательными):

1. [apikey] – ключ доступа к сервису;
2. [text] – текст поискового запроса. Использовалось название трех типов объектов;
3. [type] – типы возвращаемых результатов. Использованное значение: biz — организации;
4. [lang] – язык ответа. Выбирался русский язык;

[11] https://tech.yandex.ru/maps/doc/geosearch/concepts/about-docpage/

5. [ll] – центр области поиска, задавался долготой и широтой в градусах в виде десятичной дроби, разделенных запятой;
6. [spn] – размер области поиска, указывался в виде долготы и широты в градусах в виде десятичной дроби, разделенных запятой;
7. [results] – количество возвращаемых объектов. Максимальное значение — 500;

Формат ответа. Сервер возвращал найденные объекты, количество которых в одном поисковом запросе не могло быть больше 500. Ответ был представлен в виде вложенных словарей с ключами. Названия некоторых из них:

- results – максимальное количество возвращаемых результатов;
- skip – количество пропускаемых результатов;
- found – количество найденных объектов;
- type – тип геометрии;
- coordinates – координаты объекта;
- boundedBy – границы области показа найденных объектов;
- geocoderMetaData – подробная информация о найденном объекте;
- text – полный информация об объекте.

Поскольку актуальные данные «Яндекс.Карт» больше не содержат рейтинговые оценки набережных, которые необходимы для построения «самых интересных» прогулок, был задействован новый источник данных – социальная сеть «ВКонтакте». С помощью API ВКонтакте[12] была получена информация о фотографиях, сделанных на территории проекта. API ВКонтакте – это интерфейс, который позволяет получать информацию из базы данных vk.com с помощью http-запросов к специальному серверу. Для выгрузки фотографий использовался метод *photos.search*[13]. Он возвращает объект, содержащий число результатов в поле *count* и массив объектов фотографий в поле *items*.

Формат запроса. Начало запроса:*'https://api.vk.com/method/photos.search?v=5.00*. Далее указывались параметры запроса:

1. api_key – ключ доступа к сервису;
2. lat, long – географические широта и долгота отметки, заданная в градусах;
3. count – количество возвращаемых фотографий;
4. radius – радиус поиска в метрах;

[12] https://vk.com/dev/first_guide
[13] https://vk.com/dev/photos.search

5. start_time – время в формате unixtime, не раньше которого должны были быть загружены найденные фотографии;
6. end_time – время в формате unixtime, не позже которого должны были быть загружены найденные фотографии;

Формат ответа. Сервер возвращал найденные объекты, количество которых в одном поисковом запросе не могло быть больше 1000. Ответ был представлен в виде вложенных словарей с ключами. Названия некоторых из них:

– count – количество фотографий;
– id – идентификатор фотографии;
– owner_id – идентификатор пользователя, опубликовавшего фотографию;
– photo_* – ссылка на фотографию в формате *.jpg*;
– text – комментарий к фотографии того, кто ее опубликовал;
– date – дата публикации;
– lat, long – местоположения фотографий (далее – геотеги).

В целях пользования фоновыми картами, расположенными на главной странице приложения, была реализована функция с использованием API Maps SDK для отображения трех карт: Google Maps, Google Satellite, Google Hybrid. Для этого был создан аккаунт на сайте Google, далее в программном коде клиентской части приложения был составлен запрос к онлайн-сервису с указанием ключа для доступа. Отображаемые карты являются результатом запроса, выполняемого каждый раз при загрузке мобильного приложения (Рис.8).

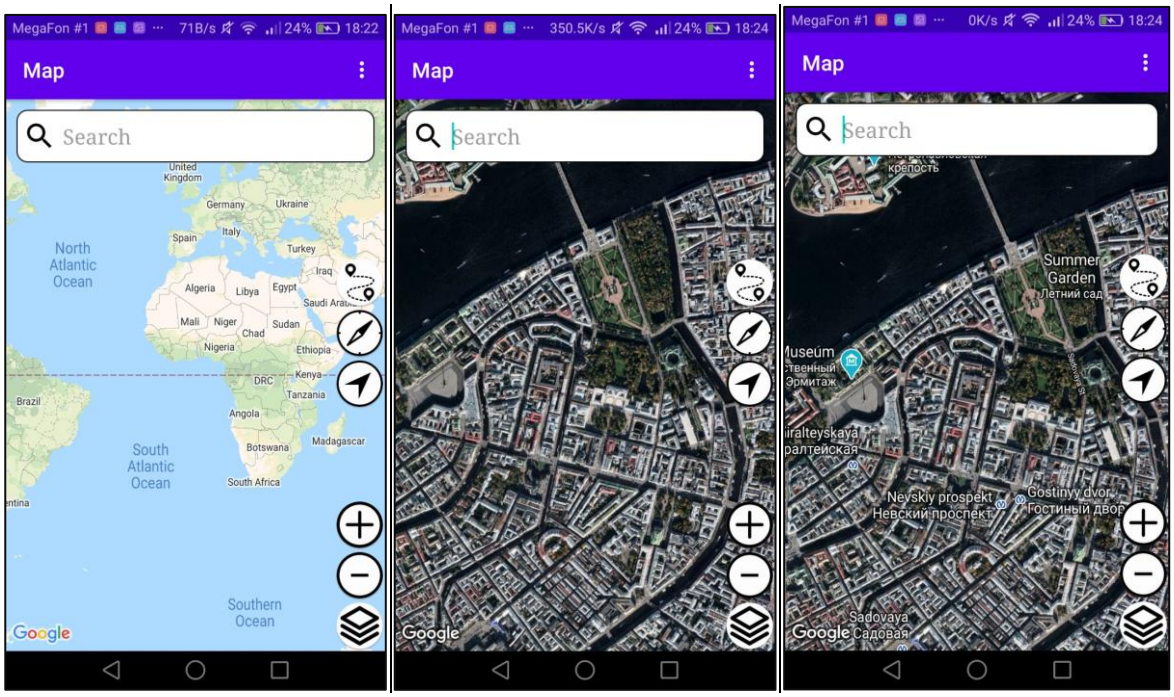


Рис. 8. Отображение карт Google Maps, Google Satellite, Google Hybrid в приложении.

Для получения описания, фотографии и ссылки на веб-ресурс[14] для каждой достопримечательности применялось API Wikipedia. Модуль Wikipedia языка программирования Python являлся инструментом для поиска нужной информации. Формат запроса состоял из двух частей:

1. *'http://ru.wikipedia.org/w/api.php?action=query&prop=pageimages&format=json&piprop=original&titles='*
2. Название страницы на веб-ресурсе. Другими словами, название достопримечательности.

Результат возвращался в формате объекта класса WikipediaPage, поля которого позволили получить описание (WikipediaPage.summary), веб-ссылку (WikipediaPage.url) и фотографию (WikipediaPage.title) каждой достопримечательности.

Таким образом, рассмотрено множество исследований и готовых сервисов по построению пешеходных маршрутов, начиная с 2004 по 2019 год. В ходе анализа существующих сервисов по построению пешеходных маршрутов были сформулированы функциональные особенности разрабатываемого прототипа приложения. Основные из них: мобильная версия приложения, добавление и обновление информации об объектах, инструменты навигации. Обозначены области применения функциональных возможностей геоинформационных программных продуктов, СУБД, языков программирования и среды разработки клиентской части приложения. Были определены источники исходных данных для разработки приложения на разных этапах.

[14] https://ru.wikipedia.org/wiki/Заглавная_страница

Глава 2. Исследование исходных данных проекта

2.1. Анализ данных из различных источников

Помимо изучения существующих алгоритмов и сервисов, было проведено исследование данных из трех источников: поисково-информационной картографической службы «Яндекс.Карты», веб-ресурса «Wikipedia» и социальной сети «ВКонтакте». Последний использовался как ресурс для получения фотографий. Данные о них рассматривались как источник для получения актуальных оценок по популярности для набережных, поскольку служба «Яндекс.Карты» такую информацию больше не предоставляет. Данные веб-ресурса «Wikipedia» применялись для получения информации о достопримечательностях (текстовое описание, фотографии, ссылка о них на веб-ресурс)

Данные «Яндекс.Карты» использовались для построения «самых интересных» прогулок. Было проведено сравнение двух наборов данных, полученных в апреле 2019 года и в апреле 2020 года из этого источника. В набор данных входило: местоположения центроидов, названия и рейтинговые оценки памятников, фонтанов, зданий с архитектурной ценностью, мостов, набережных города, площадей и озелененных территорий. Целью сравнения являлось исследование данных за апрель 2020 года на предмет добавления/обновления/изменения по сравнению с набором за апрель 2019 года. Для получения данных составлялся запрос к серверу «Яндекс.Карты». Затем ответ на запрос был преобразован в несколько точечных shape-файлов. Каждый файл содержал один из видов достопримечательностей, который обозначался в переменной «text» в программном коде (Рис.9).

```
def getWKT_PRJ(epsg_code):
    # access projection information
    wkt = urllib.request.urlopen("http://spatialreference.org/ref/epsg/
    # remove spaces between charachters
    remove_spaces = wkt.read().decode('ISO-8859-1').replace(" ", "")
    # place all the text on one line
    output = remove_spaces.replace("\n", "")
    return output

text='достопримечательности'
ll='30.32815,59.94071'
spn='0.035,0.017'
results='500'


border_terr='C:\\Documentiki\\My Diploma\\Rama\\rama.shp'
file_name='dostoprims'

# path to .json file
folder='C:\\Documentiki\\Research_April\\new_data_Yandex\\dostoprim'

site='https://search-maps.yandex.ru//v1//?text=%(text)s&ll=%(ll)s&spn=%
    %{'text':text,'ll':ll,'spn':spn,'results':results,'apikey':apikey}
r = requests.get(site)
r.encoding = 'ISO-8859-1'
data=js.loads(r.text)
```

Рис.9. Фрагмент программного кода для выгрузки местоположений зданий с архитектурной и исторической ценностью.

В процессе сравнения двух наборов данных было замечено:

1. увеличение количества объектов в каждой категории;
2. изменение названий объектов. Например, название объекта «Иверская часовня-ризница при соборе Воскресения Христова» заменено на «Ризница»;
3. изменение оценок по популярности. Например, объект «Певческий мост» имел оценку 0, сейчас его оценка равна 4,5;
4. изменение местоположений. Связано со сменой расположения объекта на веб-странице «Яндекс.Карты» (Рис.10б);

Были выявлены следующие особенности актуальных данных:

1. появились новые категории: «жанровая скульптура» и «мемориалы». В ответе на запрос «мемориалы» приходило примерно треть объектов категории «жанровая скульптура». Не все объекты из категории «мемориалы» входили в категорию «достопримечательности», даже если они имели оценку по популярности. Ответ на запрос «памятники, мемориалы» приходил другой результат, нежели на запрос «мемориалы» и «памятники» раздельно;
2. по запросу «мемориал» данных выгружалось больше, чем по запросу «мемориалы»;
3. ответом на запрос «достопримечательности» являлся идентичный ответ, как и на запрос «достопримечательность»;
4. при сравнении объектов, выгруженных с интервалом в одну неделю, выяснилось, что более ранний запрос для выгрузки «достопримечательностей» содержал на 4 объекта больше, чем запрос неделей позже. Это связано с удаление дублирующихся объектов (Собор Воскресения Христова Спас на Крови находился и в Доме служащих Русского музея (Рис.10а)), изменением категории объекта (например, неделей раньше музей «Фаберже» относился к категории «достопримечательность», а не к «музеи»);
5. ответы на запросы «памятник», «памятники», «памятники,мемориалы» и «памятник,мемориал» являлись идентичными.

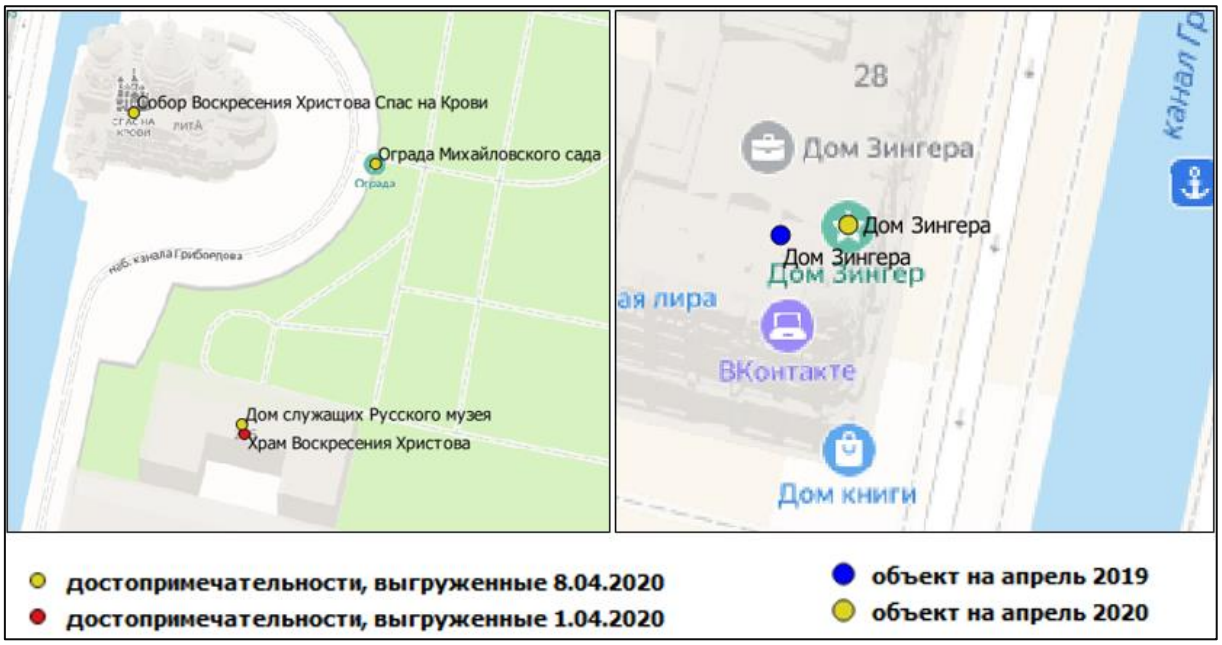


Рис.10а. Изменение названия объекта.

Рис.10б. Изменение местоположения объекта.

Таким образом, сервис «Яндекс.Карты» выполняет обновление своей БД как минимум с периодом в 1 неделю. Добавляются новые места, изменяются оценки по популярности, корректно переносятся местоположения объектов, происходит упрощение названий. Это означало, что данные можно использовать и далее для построения «самых интересных» прогулок при создании прототипа мобильного приложения.

Помимо анализа данных «Яндекс.Карт» был изучен набор данных из социальной сети «ВКонтакте». С помощью API этой социальной сети удалось получить и проанализировать геотеги на тестовую территорию. Для выгрузки данных были использованы возможности языка программирования Python. В результате работы программного кода (Рис.11а), был осуществлен запрос к серверу социальной сети и получен ответ, который сохранялся в таблицу БД. Формат ответа с сервера представлен на Рис.11б.

```
def get_photos(self, rid, geom, date_from, date_to):

    #print(type(geom))

    url = 'https://api.vk.com/method/photos.search?' \
          'v=5.00&access_token=%(api_key)s&' \
          'lat=%(lat)s&long=%(lon)s&' \
          'count=%(count)s&' \
          'radius=%(radius)s&' \
          'start_time=%(start_time)s&' \
          'end_time=%(end_time)s' % {
              'radius': self.radius,
              #'lat': geom.y,
              'lat': geom[0].y,
              #'lon': geom.x,
              'lon': geom[0].x,
              'api_key': self.api_key,
              'count': self.photos_max_count,
              'start_time': date_from,
              'end_time': date_to
          }
```

Рис.11а. Фрагмент кода для получения данных из «ВКонтакте».


```
{
  "response": {
    "count": 23,
    "items": [{
      "id": 363464848,
      "album_id": -6,
      "owner_id": 65056617,
      "photo_75": "https://pp.vk.me/...672/mR_9By_IT9o.jpg",
      "photo_130": "https://pp.vk.me/...673/b_M63SxBEVo.jpg",
      "photo_604": "https://pp.vk.me/...674/Lsqt2qs8SYI.jpg",
      "photo_807": "https://pp.vk.me/...675/umc1xwq9M-g.jpg",
      "photo_1280": "https://pp.vk.me/...676/ACo0Gf8sBC4.jpg",
      "width": 1280,
      "height": 960,
      "text": "",
      "date": 1429623443,
      "lat": 29.954948,
      "long": 29.838876,
      "post_id": 240
    }, {
      "id": 360453137,
      "album_id": -7,
```


Рис.11б. Формат ответа на запрос.

Выгрузка данных осуществлялась для каждого центроида, расположенного в экстенте территории на расстоянии 200 метров друг от друга (Рис.12). Координаты каждого центроида указывались в строке запроса (Рис.11а). Предполагалось, что вокруг каждого центроида будут получены геотеги фотографий с соответствующей атрибутивной информацией. Создание центроидов осуществлено с помощью инструмента «Регулярные точки» в программе QGIS. Перед началом работы над выгрузкой данных, центроиды были импортированы в отдельную таблицу БД. Были созданы необходимые поля в этой таблице (Рис.13): «id» – идентификатор центроида; «geom» – создано автоматически, поле описания координат объекта; «centroid» – координаты центроида в формате WKT, необходимые для работы программного кода. Значения в этом поле были сформированы с помощью «Калькулятора полей» в программе QGIS (Рис.14); «proc» – поле, идентифицирующее окончание выгрузки данных для объекта. Для каждого центроида этом поле могло содержаться одно из двух значений: «true» – выгрузка данных для центроида выполнена и «false» – выгрузка еще не произошла.

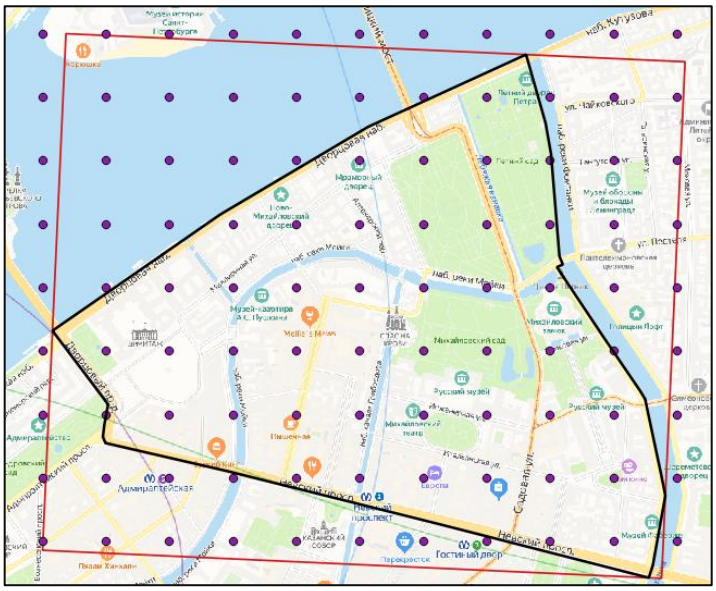


Рис.12. Центроиды для выгрузки данных.

id [PK] bigint	geom geometry	proc boolean	centroid text
1	0104000020E61...	false	MultiPoint ((30.31239988 59.94855582))
2	0104000020E61...	false	MultiPoint ((30.31597849 59.9485473))
3	0104000020E61...	false	MultiPoint ((30.31955709 59.94853869))
4	0104000020E61...	false	MultiPoint ((30.3231357 59.94852998))
5	0104000020E61...	false	MultiPoint ((30.3267143 59.94852117))
6	0104000020E61...	false	MultiPoint ((30.3302929 59.94851227))

Рис.13. Поля таблицы центроидов.

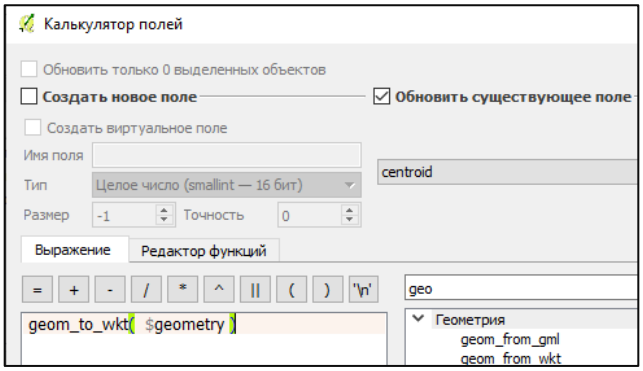


Рис.14. Вычисление значения поля.

Для сохранения ответа на запрос была сформирована таблица в БД с соответствующими полями (Рис.15): «id» – идентификатор фотографии; «owner_id» – идентификатор автора фотографии; «title» – комментарий к фотографии; «cell_id» – идентификатор центроида, для которого фото было получено; «geom» – геотег; «upl_time» – время загрузки фото в сеть; «photo_807» – ссылка на фото.

id [PK] bigint	geom geometry	owner_id bigint	title character varying (254)	cell_id bigint	upl_time date	photo_807 character varying (254)
457245473	0104000020E61...	371207722	Часы Потёмкина	34	2020-01-06	https://sun9-12.userapi.co...
457252070	0104000020E61...	83316358	[null]	34	2020-01-06	https://sun9-11.userapi.co...
457252067	0104000020E61...	83316358	[null]	34	2020-01-06	https://sun1-88.userapi.co...

Рис.15. Фрагмент таблицы с фотографиями.

Краткое описание работы программного кода:

1. выборка необработанных центроидов (где значение поля «proc» в таблице центроидов равно «false»);
2. составление запроса на сервер для каждого из них;
3. если число объектов (фотографий) за один запрос превосходит 1000 штук (ограничение со стороны сервера), то временной промежуток разбивается на две части и запрос составляется для каждого промежутка по отдельности;
4. полученный результат преобразуется для добавления в итоговую таблицу (Рис.15);
5. центроид отмечается как обработанный («proc» = true).

Был проведен анализ результатов для одного центроида (id=60) для проверки качества выполнения запросов (Табл.1). Запросы составлялись для дат с 1 по 7 января 2020 года (период публикации фотографий пользователями). В строке «Радиус поиска» в скобках указаны фактические расстояния от центроида с id=60 до самого удаленного геотега.

Таблица 1.

Данные для центроида с id=60.

Радиус поиска, м	10	40,20(300-400 м до границ)	100(300-400 м до границ)	200(2-3 км до границ)	500(2-3 км до границ)	1000 (13-30 км до границ)
Кол-во геотегов, штук	0	1108	1108	7214	7214	15641

В результате анализа обнаружено:

1. радиус поиска весьма условный. Существует градация границ выгрузки данных для каждого радиуса поиска. Например, при радиусе поиска от 20 до 100 метров геотеги выгружаются на одну и ту же площадь, большую заданной. Тоже самое происходит для радиуса поиска 200-500 метров и от 1000 метров (Рис.16);
2. границы выгрузки данных имеют форму, подобную квадратной;

3. центроид не располагается в центре области выгрузки данных, расстояние между ним и границей территории варьируется (см. Табл.1);

4. замечена особенность, что для радиуса поиска 50 метров геотегов. выгружается меньше, чем для 20-ти или 40-ка метровом радиусе (область выгрузки данных одинаковая).

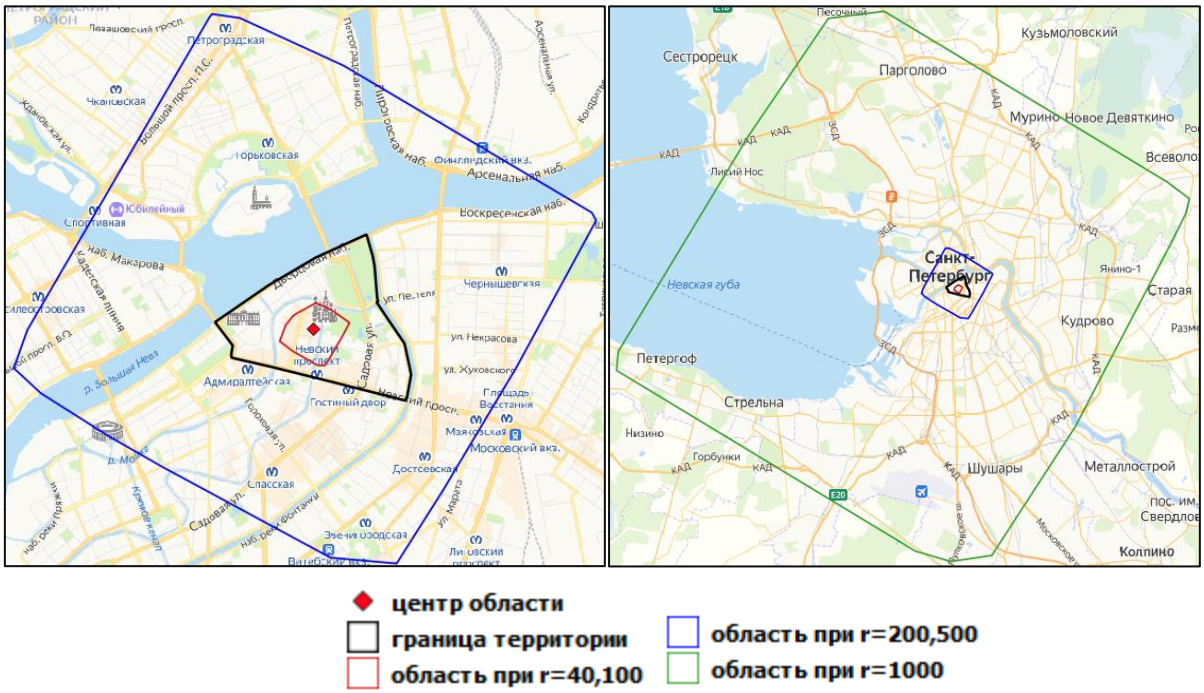


Рис.16. Границы областей выгрузки геотегов, в метрах.

Был проведен анализ результатов при условии получения геотегов для двух центроидов за один запуск программного кода. Результаты представлены на Рис.17, где отображены два центроида, полученные геотеги и области выгрузки данных. На Рис.18 обозначены области выгрузки геотегов для каждого центроида по отдельности и при обработке обоих центроидов за один запуск программного кода. При сравнении двух рисунков обозначены следующие выводы:

1. в области перекрытия двух экстентов выгрузки геотегов их количество и местоположение не совпадают;

2. получено 1390 геотегов в результате выгрузки одновременно для двух центроидов; 1419 – при выгрузке по отдельности (дублирующиеся геотеги удалены);

3. граница при одновременной выгрузке для двух центроидов (синий котнтур на Рис.18) не совпадает с границами при выгрузке для каждого центроида по отдельности (красный, зеленый контуры на Рис.17);

4. центры областей выгрузки геотегов не совпадают с местоположениями центроидов.

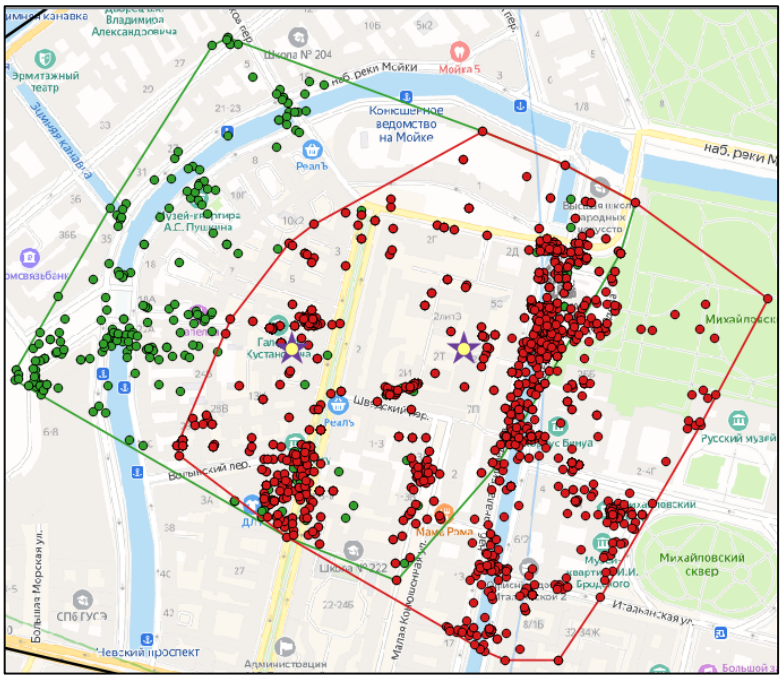


Рис.17. Геотеги для двух центроидов. Красным цветом обозначены границы и геотеги для одного центроида; зеленым – для второго. Каждый центроид обозначен звездой.

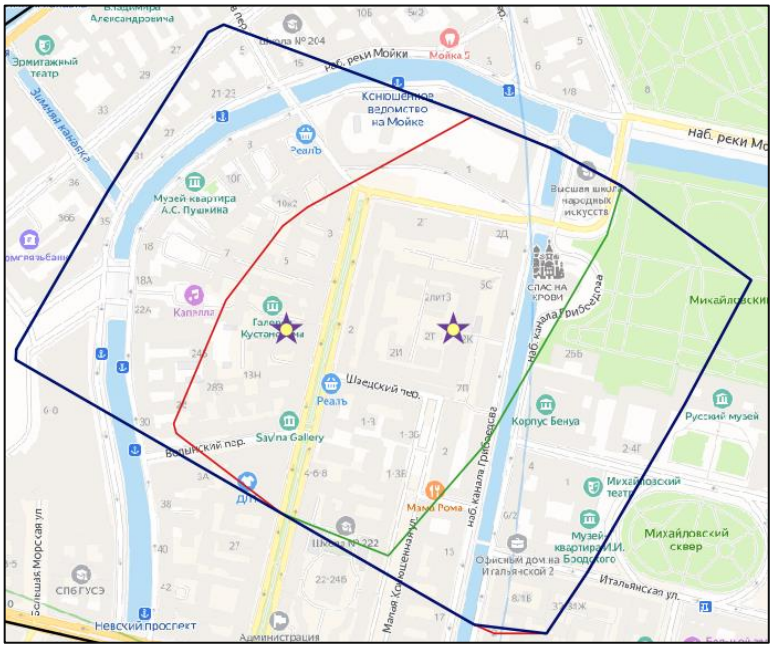


Рис.18. Области выгрузки геотегов. Синим контуром обозначена граница геотегов при выполнении программного кода сразу для двух центроидов.

В дополнении к вышеописанным исследованиям, были получены данные для всей территории проекта, где для каждого центроида (Рис.12) были получены результаты. Радиусы поиска задавались разные. Анализируя данные из Табл. 2, можно заметить, что чем больше радиус поиска, тем меньше геотегов приходило в качестве ответа. Вероятно, это было связано с ограничениями на результат запроса на сервере.

Таблица 2.

Результаты выгрузки геотегов при различных радиусах поиска.

Радиус поиска, м	20, 40	100	200
Количество геотегов внутри экстента, штук	6136	5345	3753

Выполнен анализ результатов в области при радиусе поиска 40 метров (красный контур) для одного центроида (рис.19). Сравнивались геотеги, выгруженные на всю территорию сразу и только для одного центроида. Были сделаны заключения, что:

1. получено 617 геотегов при выгрузке данных сразу на всю территорию; 1108 – при выгрузке только для одного центроида;
2. не все геотеги в количестве 617 штук содержатся во множестве 1108 штук.

Отсюда следует вывод, что геотеги выгружаются хаотично, причина этого явления неизвестна.

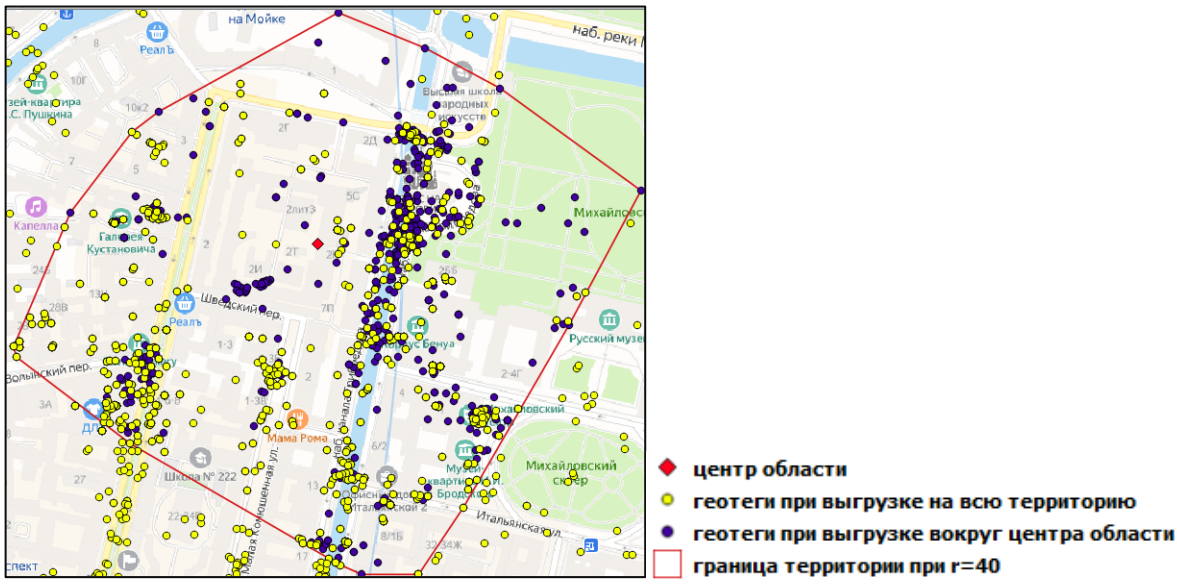


Рис.19. Результат выгрузки геотегов при разных условиях.

Также было замечено, что если составлять запрос с разделением временного промежутка пополам, то количество геотегов в сумме на равно количеству при неразделенном временном промежутке. При получении данных через сайт[15], выяснилось, что при изменении параметра «count» (количество возвращаемых фотографий), меняется общее число фотографий (оно не должно зависеть от «count»).

В ходе проведенного анализа данных социальной сети «ВКонтакте», были сформулировано несколько общих выводов:

1. чем меньше область поиска, тем больше данных;

[15] https://vk.com/dev/photos.search

2. результаты запросов на сайте отличаются от результатов, получаемых с помощью программного кода;

3. количество геотегов, получаемое для цельного промежутка времени, не равно сумме результатов выгрузки по интервалам;

4. записи в ответе на запрос повторяются.

Предположительно, причины таких особенностей связаны либо с ограничениями, либо некорректной работой со стороны сервера.

В целях отображения информации о достопримечательностях (текстовое описание, фотографии, ссылки на веб-ресурс) в приложении, был написан программный код на языке Python. С использованием методов, описанных в разделе 1.3, была получена необходимая информация в виде текстового файла. Затем методами геопространственного анализа результаты работы программного кода были представлены в БД, а именно, в таблице хранения достопримечательностей (Рис.20).

	id bigint	geom geome...	name character varying (254)	field_3 character varying (254)	photo character varying (254)
129	128	01040000...	Дом И.Г. Неймана - К.Б. ...	Не́вский проспе́кт — главная улица ...	C:\Documentiki\Final_Research\for_WIKIPEDIA\result\images\127.jpg
130	129	01040000...	Казармы саперного бат...	[null]	C:\Documentiki\Final_Research\for_WIKIPEDIA\result\images\128.jpg
131	130	01040000...	Дом П.В. Неклюдова	Лите́йный проспект — одна из важн...	C:\Documentiki\Final_Research\for_WIKIPEDIA\result\images\129.jpg
132	131	01040000...	Особняк И.А. Мусиных-...	Лите́йный проспект — одна из важн...	C:\Documentiki\Final_Research\for_WIKIPEDIA\result\images\130.jpg
133	132	01040000...	Дворец И.И. Шувалова	Дворец Шувалова в Санкт-Петербур...	C:\Documentiki\Final_Research\for_WIKIPEDIA\result\images\131.jpg
134	133	01040000...	Особняк семьи графа В...	Особняк Л. В. Кочубе́я (Ю. С. Нечае́в...	C:\Documentiki\Final_Research\for_WIKIPEDIA\result\images\132.jpg
135	134	01040000...	Доходный дом Немецко...	Не́вский проспе́кт — главная улица ...	C:\Documentiki\Final_Research\for_WIKIPEDIA\result\images\133.jpg

Рис.20. Результат работы программного кода по поиску информации о достопримечательностях.

Программный код осуществлял поиск веб-страницы по названию каждой достопримечательности. Затем он заимствовал первый абзац текстового описания с добавлением его в текстовый файл и загружал фотографию в специальный каталог на компьютере. В процессе анализа результатов выяснилось, что только у 282 из 627 объектов удалось найти текстовое описание, некоторые из них ошибочные (Рис.20); только 283 из 627 фотографий было загружено. Недостаточное количество найденной информации связано с отсутствием страницы на объект у веб-ресурса по его названию, хранящееся в БД. Вариант решения проблемы – поиск веб-страницы не по названию, а по координатам достопримечательности.

В результате, было выполнено обновление исходных данных из источника «Яндекс.Карты» и проанализированы нововведения актуальных данных. Получены и исследованы данные из нового источника – социальной сети «ВКонтакте» и «Wikipedia» с целью их дальнейшего применения в работе. Сделаны выводы об особенностях предоставления данных «ВКонтакте», найден оптимальный радиус поиска для

получения наибольшего числа фотографий. Выбраны исходные данные для решения различных задач: при построении «самых интересных» прогулок применялись данные «Яндекс.Карт», полеченные в апреле 2020 года, и данные «Вконтакте» для оценки по популярности набережных города. Для получения подробной информации о достопримечательностях использован веб-ресурс «Wikipedia».

2.2. Подготовка исходных данных для исследования «самых интересных» прогулок

Как уже упоминалось, для того чтобы построить «самый интересный» маршрут, необходимо наличие весового значения каждого ребра дорожного графа. В данном исследовании сравнивались маршруты, построенные по трем возможным категориям весов. Они были получены из трех наборов данных (каждому набору соответствовала своя категория весового значения).

Для исследования «самых интересных» прогулок использовались данные, содержавшие:

– достопримечательности сервиса «Яндекс.Карты», полученные в апреле 2019 года;

– достопримечательности сервиса «Яндекс.Карты», полученные в апреле 2020 года;

– фотографии социальной сети «ВКонтакте», полученные в апреле 2020 года. В рассматриваемом исследовании данные «ВКонтакте» использовались для присвоения всем ребрам графа дорог весовых значений, а не только набережным. Фотографии были опубликованы пользователями в июле 2019 года. Такой временной промежуток был выбран исходя из туристической привлекательности города Санкт-Петербург в это время и отсутствием на тот момент карантинных ограничений.

Для того чтобы вовлечь данные «ВКонтакте» в построение прогулок, нужно было рассмотреть два варианта их использования (другими словами, на основе чего будет происходить присвоение новых весовых значений ребрам дорожного графа):

– посчитать количество фотографий, приходящееся на каждый квадрат (элемент разбиения территории). В этом случае результатом соответствующих операций будет «карта» концентрации туристов. Можно предположить, что она прямо пропорциональна популярности той или иной области. Значит, на основе «карты» можно строить «самые интересные» прогулки. Этот подход имеет свои недостатки. Во-первых, весовое значение каждого элемента дорожного графа

зависит от того, в каком квадрате он содержится. Размер квадрата может быть любой, и он влияет на результирующее весовое значение. Во-вторых, все ребра графа будут иметь одинаковые веса в пределах одного квадрата. В итоге, этот метод не был реализован на практике;

– посчитать, сколько фотографий той или иной достопримечательности было сделано. Для этого необходимо присвоить новый атрибут достопримечательностям. Последние представлены в виде точек (например, фонтаны), линий (набережные) и полигонов (площади). Предполагается, что на одной фотографии изображена только одна достопримечательность (связь один к одному) и только одна фотография может быть сделана пользователем на одном месте. Таким образом, чем больше фотографий было сделано, тем популярнее достопримечательность. Это значит, что такой подход тоже мог бы использоваться для построения интересных прогулок.

Для реализации второго подхода было написано два программных кода. Первый решал задачу поиска уникальных фотографий (Рис.21). Краткое описание кода:

– импорт модулей;

– определение входного и выходного файлов, чтение входного файла;

– добавление в выходной файл данных об уникальных фотографиях из входного файла (идентификаторы фотографии и владельца, дата публикации, ссылка на фотографию). Уникальность определялась координатами создания фотографии и идентификатором владельца.

```
photos = []
bag = set()

for feature_in in layer_in:
    geom = feature_in.GetGeometryRef()
    owner_id = feature_in.GetField(1)

    item = (geom.GetX(), geom.GetY(), owner_id)
    if item not in bag:
        bag.add(item)
        feature_out = ogr.Feature(layer_out.GetLayerDefn())
        feature_out.SetFrom(feature_in)
        layer_out.CreateFeature(feature_out)
        feature_out.Destroy()
        feature_out = None

    feature_in.Destroy()
    feature_in = None

copy(shp_name_in.split(".")[0] + ".prj", os.path.join(root, base + ".prj"))

ogr_data_in.Destroy()
ogr_data_in = None
ogr_data_out.Destroy()
ogr_data_out = None
```

Рис.21. Фрагмент программного кода для поиска уникальных фотографий.

В результате, количество фотографий уменьшилось с 13452 до 9986 штук. Были удалены фотографии, сделанные за пределами области исследования (Рис.22). Удаленные фотографии не запечатлели достопримечательности в области территории

исследования, а значит, могли бы внести искажения при построении «самых интересных» маршрутов.

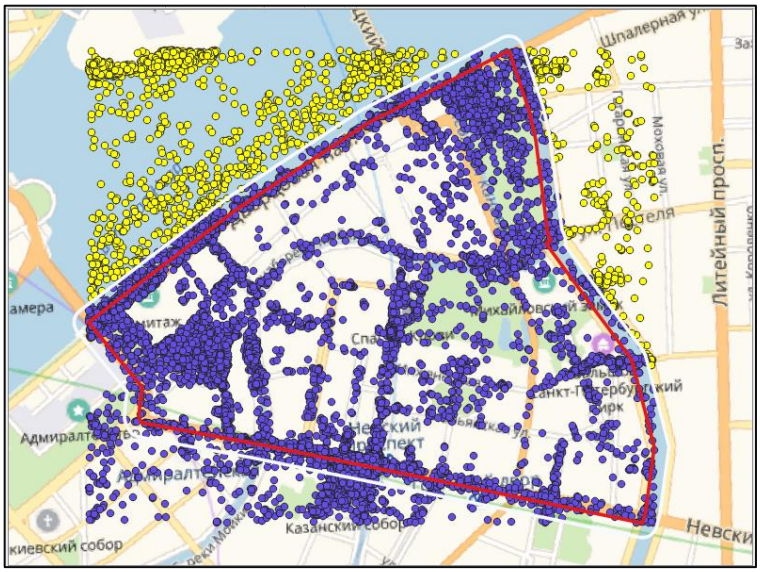


Рис. 22. Удаление геотегов, находящихся за реками Нева и Фонтанка. Желтым цветом отмечены объекты для удаления.

Второй программный код решал задачу присвоения достопримечательностям (точечным, линейным и полигональным) нового атрибута – количество фотографий каждой достопримечательности (Рис.23а). Краткое описание кода:

– импорт необходимых модулей;

– определение входных и выходных файлов;

– для каждой фотографии – подсчет расстояний до всех объектов в трех файлах достопримечательностей;

– поиск кратчайшего расстояния – присвоение фотографии ближайшей достопримечательности (увеличение общего количества фотографий на +1). Далее эта фотография в обработке не участвует.

В результате, в атрибутивных таблицах трех видов достопримечательностей были сформированы поля с названием «Count» – общее количество фотографий для каждого объекта (Рис.23б).

```
ogr_data_out_poly = drv.CreateDataSource(shp_name_out_poly)
layer_out_poly = ogr_data_out_poly.CreateLayer("polygon", No

for i in range(feat_def_poly.GetFieldCount()):
    field_def = feat_def_poly.GetFieldDefn(i)
    layer_out_poly.CreateField(field_def)

new_field = ogr.FieldDefn("Count", ogr.OFTInteger)
layer_out_poly.CreateField(new_field)


#open lines
ogr_data_in_line = ogr.Open(shp_name_in_line)
layer_in_line = ogr_data_in_line.GetLayer(0)
feat_def_line = layer_in_line.GetLayerDefn()
```

Рис.23а. Фрагмент кода для присвоения достопримечательностям числа фотографий.

Name	Value	Count
Фонтан-каскад ...	3.8000000000	50
Фонтан	0.5000000000	5
Фонтан	0.5000000000	0
Фонтан	0.5000000000	38
Фонтан	0.5000000000	2
Фонтан	0.5000000000	2
Нереида	0.5000000000	1
Коронный фонтан	0.5000000000	7

Рис.23б. Результат выполнения кода.

В процессе анализа результатов выяснилось, что полигональные достопримечательности «забирают все фотографии» в себя. Например, все фотографии, сделанные на Дворцовой площади, будут присвоены только самой площади. Для Александрийской колонны значение фотографий равно 0, потому что расстояния от нее до любой фотографии в пределах площади больше, чем до самой площади (которое равно 0). Для решения этой задачи в программе QGIS были построены буферные зоны 15 метров вокруг точечных достопримечательностей. Для Александрийской колонны буфер составлял 40 метров (Рис.24). Далее инструментом «Разность» в QGIS буферы были вырезаны из полигональных достопримечательностей (Рис.25). Результат работы этого инструмента использовался далее для исследований.

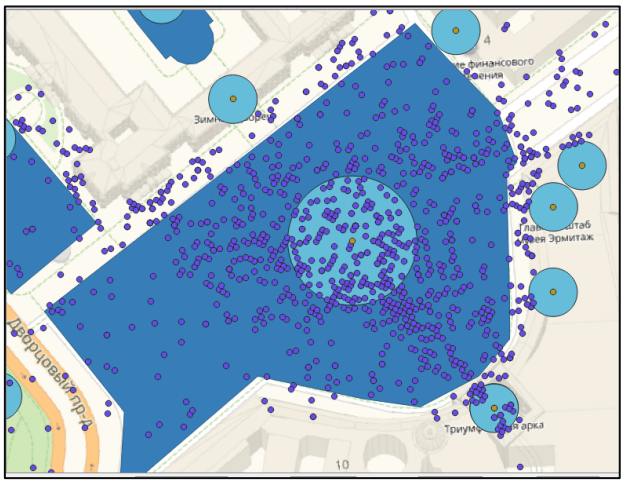


Рис.24. Буферные зоны вокруг достопримечательностей в виде точек.

Рис.24. Буферные зоны вокруг достопримечательностей в виде точек.

По этой же причине вокруг линейных объектов (набережных) был сделан буфер 5 метров

(Рис.26), найдена геометрическая разность с достопримечательностями в виде полигонов (Рис.27).

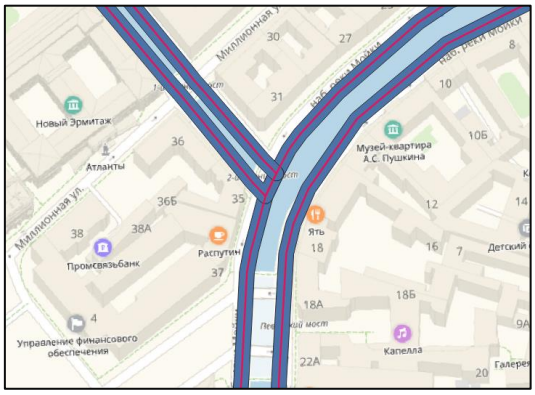


Рис.26. Буферные зоны вокруг набережных.

Рис.27. Результат работы инструмента.

После выполнения геометрической разности, программный код с использованием геометрически обновленных полигональных объектов был снова запущен. У объектов с количеством фотографий, равные 0, значение было установлено 0.5 с целью дальнейшего включения таких достопримечательностей в обработку. В результате, каждой достопримечательности, представленной одним из трех геометрическим примитивом, было корректно присвоено число фотографий. Максимальное количество фотографий для одной достопримечательности – 1300.

Данные об оценках достопримечательностей сервиса «Яндекс.Карты» варьируются в пределах от 0,5 до 5. Количество фотографий из социальной сети «ВКонтакте», присвоенных каждой достопримечательности на предыдущем этапе, варьировалось от 0,5 до 1300. Для дальнейшего сравнения результатов применения двух источников данных было необходимо нормализовать шкалу числа фотографий от 0,5 до 5. Для этого использовалась формула линейного преобразования. С помощью инструмента «Калькулятор полей» было создано новое поле и вычислены нормализованные значения для фотографий по формуле (Рис.28). Нормализированные значения – новый тип оценки достопримечательностей по фотографиям.

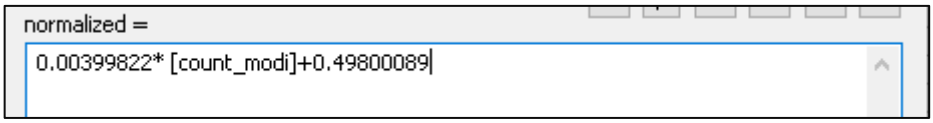


Рис.28. Нормализация данных.

Таким образом, были получены два вида оценок достопримечательностей по популярности: по данным «Яндекс.Карты» и «ВКонтакте». Значения двух видов оценок варьировались от 0,5 до 5. Для того чтобы оценить влияние оценок на построение «самых интересных» прогулок, были предложены параметры «притяжения» объектов в рамках

бакалаврской выпускной квалификационной работы (Табл.3). Оценка по популярности отдельного объекта говорила об его значимости среди других объектов. Следовательно, «притяжение» объекта прямо пропорционально его оценке. Чем выше была оценка, тем больше должен был отклоняться маршрут от кратчайшего с целью построения «самого интересного» маршрута через «притягательный» объект. Были выбраны параметры «Оценка объекта – Максимальное расстояние в метрах», где первый – оценка, взятая из источника «ВКонтакте» («Яндекс.Карты»); второй – максимальное расстояние, при котором его объект будет вносить свой вклад в результирующий вес дороги.

Таблица 3.

Параметры «притяжения» объектов.

Оценка объекта из источника	Максимальное расстояние в метрах
5	400
4	330
3	260
2	190
1	120
Менее 1	70

Результирующие веса дорогам для «самого интересного» маршрута зависели от трех видов объектов (достопримечательностей, набережных и озелененных территорий). Необходимо было создать общий массив данных, где учитывались бы влияния всех объектов. Для решения этой задачи применялся расчет по евклидовому расстоянию[16]. В рамках выпускной работы был написан программный код, позволяющий строить растровое изображение (далее – растр), где в качестве входных данных использовались точечные (достопримечательности), линейные (набережные с заданными оценками) и полигональные (озелененные территории, площади) объекты. Написанный алгоритм был внедрен в программный продукт ArcMap. Алгоритм работы кода был следующий:

– импорт модуля «arcpy» для работы алгоритма из-под ArcMap;

– задание параметров (исходный shape-файл с линейными, точечными или полигональными объектами, название поля со значениями оценок в этом файле, размер ячейки выходного растра, путь для сохранения выходного растра). Эти же параметры указывались в диалоговом окне нового инструмента в программе ArcMap (Рис.29). Размер ячейки растра был выбран 5 метров для повышения точности результатов;

– создание растра со значениями в каждой ячейки равными нулю;

[16] http://desktop.arcgis.com/ru/arcmap/latest/tools/spatial-analyst-toolbox/euclidean-distance.htm

– на каждой итерации проводилось создание растра для каждого по отдельности объекта входного shape-файла; значение в каждой ячейке растра высчитывалось по формуле, выведенной на основе данных из Табл.3;

В конце первой итерации цикла создавался новый растр, который получался в результате сложения значений соответствующих ячеек «нулевого» и созданного для отдельного объекта растров. На следующих итерациях вновь создавался растр для следующего объекта из исходного shape-файла и складывался с растром, полученным в предыдущей итерации (Рис.30).

– создание результирующего растра, значение в каждой ячейки которого было суммарное влияние всех объектов (достопримечательностей).

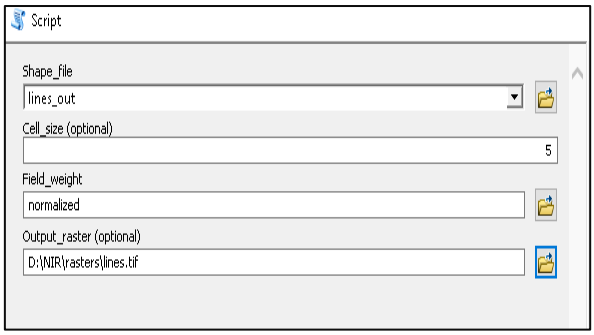


Рис. 29. Окно инструмента в программе ArcMap.

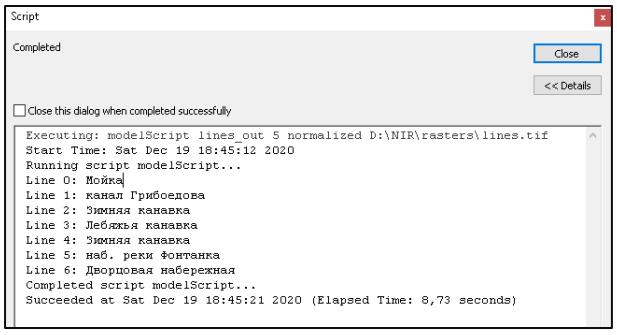


Рис.30. Процесс обработки входного линейного shape-файла.

Результаты работы программного кода, вычисляющий значения в ячейках растров данным социальной сети «ВКонтакте» представлены на Рис.31-33.

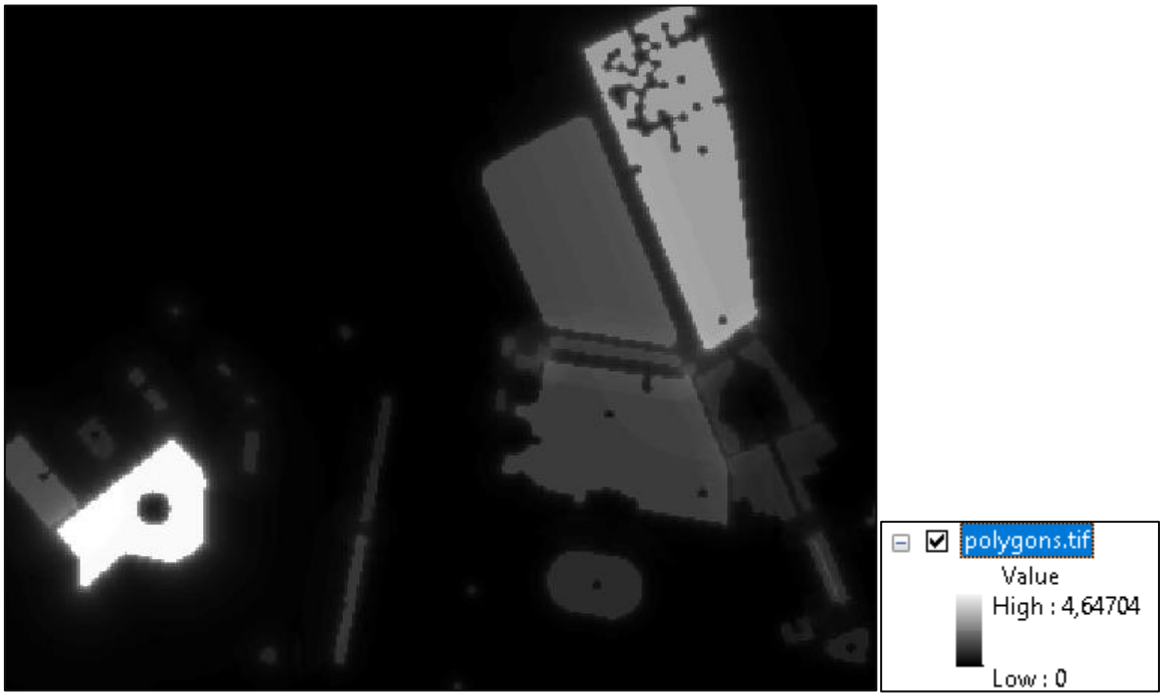


Рис.31. Результат выполнения кода для полигональных достопримечательностей (озелененные территории+ площади).

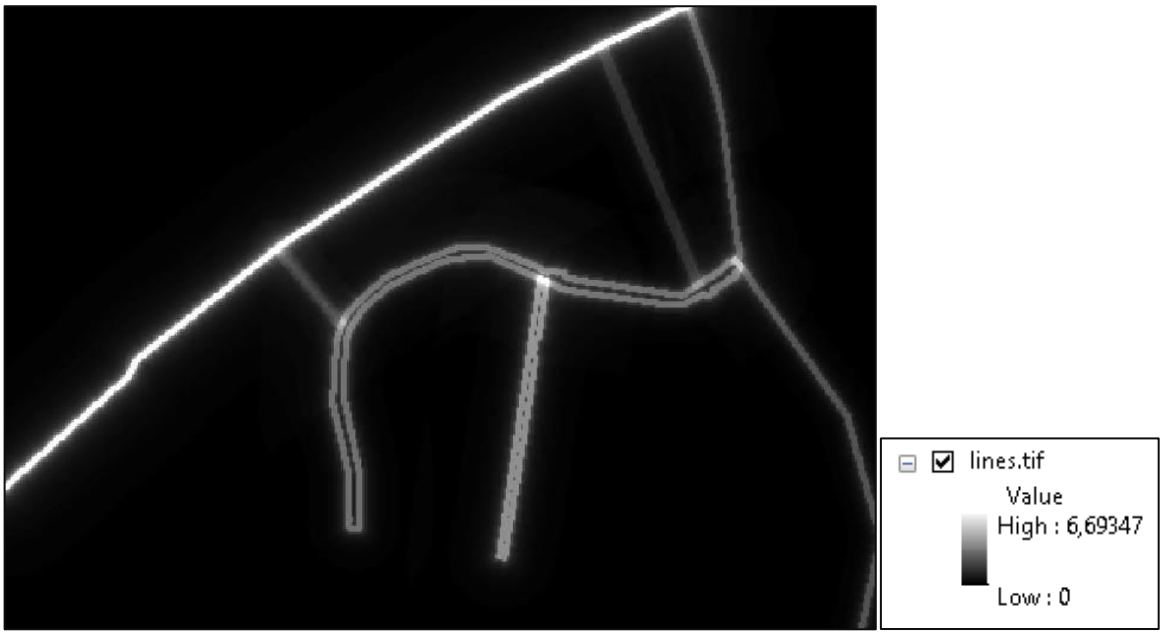


Рис.32. Результат выполнения кода для линейных достопримечательностей (набережные).

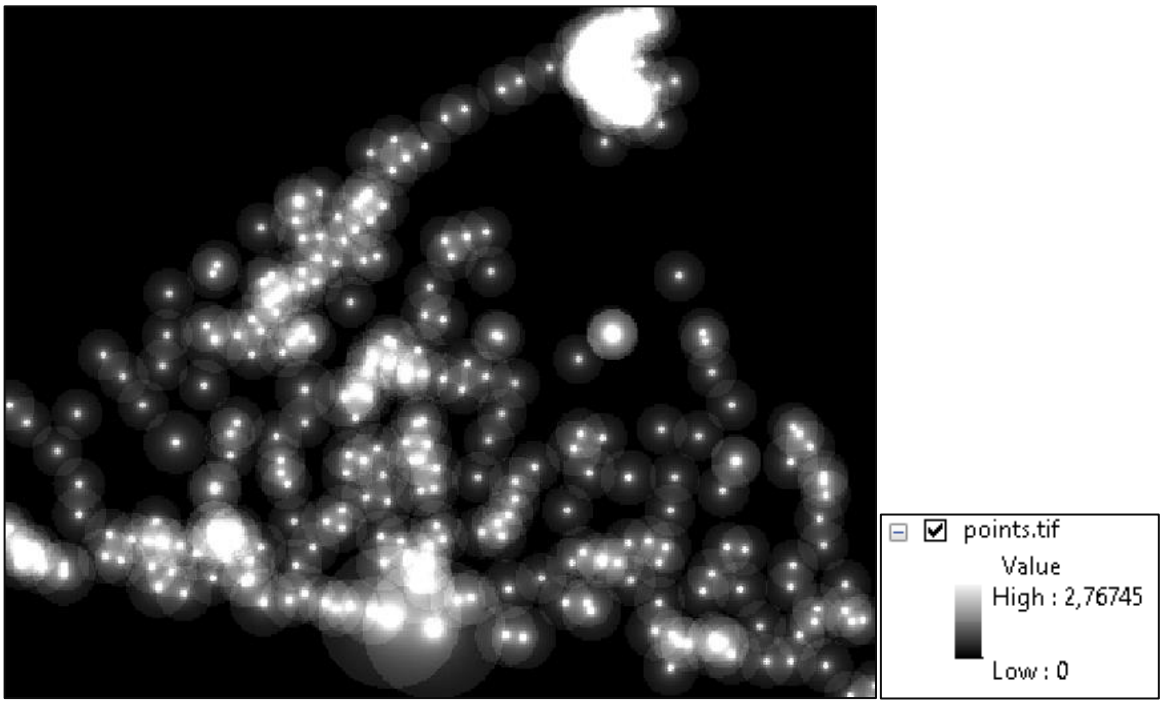


Рис.33. Результат выполнения кода для точечных достопримечательностей (фонтаны, памятники и так далее).

После вычисления трех растров, инструментом «Калькулятор растров» был получен общий «растр весов» – суммарная оценка на основе фотографий внутри области исследований (Рис.34).

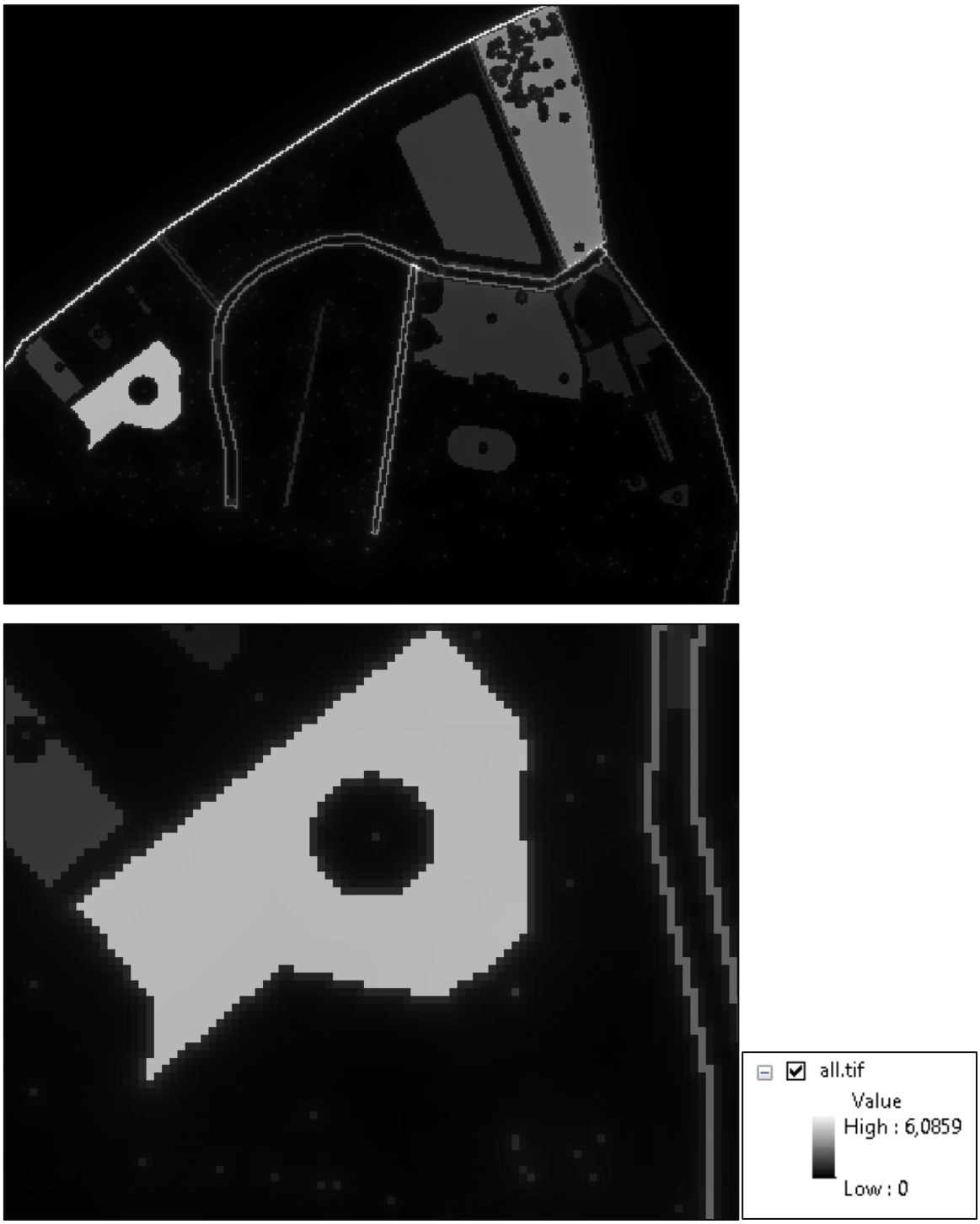


Рис.34. Растр весов на основе данных о фотогрфиях.

Как уже упоминалось ранее, данные о достопримечательностях сервиса «Яндекс.Карты» были представлены в виде нескольких точечных shape-файлов (памятники, фонтаны, мемориалы, центроиды парков, площади). Для дальнейшего анализа все точечные файлы были соединены в один, удалены дубликаты. Полигональным достопримечательностям (озелененные территории + площади) была присвоена оценка, взятая из точеченых файлов. Оценки для набережных сервис больше

не предоставляет, поэтому в рамках этого исследования (не для построения маршрутов в мобильном приложении) данные о них были взяты из бакалаврской выпускной работы.

С использованием вышеописанного метода, как и для данных из «ВКонтакте», был получен растр весов для данных «Яндекс.Карты» (Рис.35).

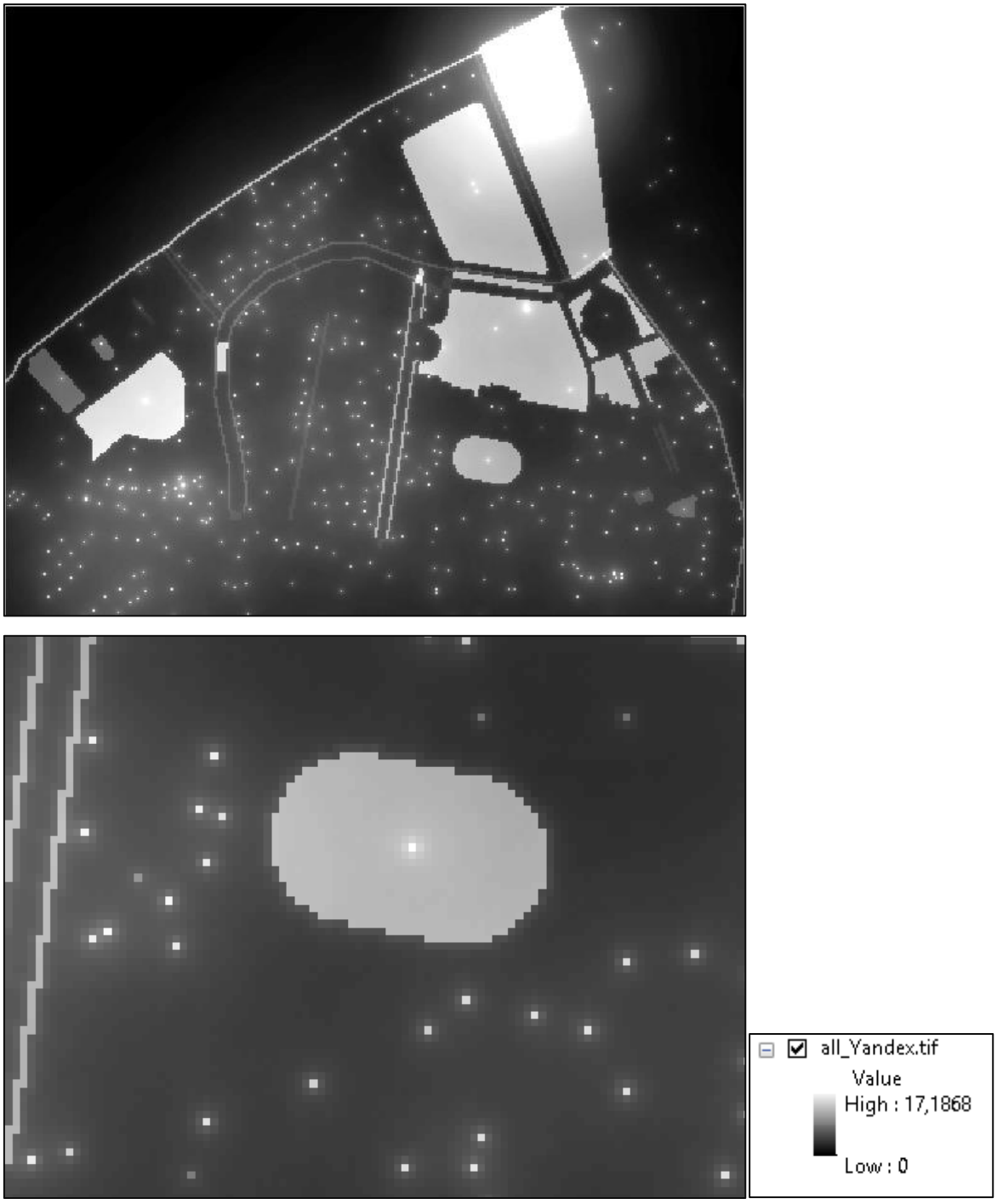


Рис.35. Растр весов для данных «Яндекс.Карты».

Поскольку пешеходные дороги были представлены в виде shape-файла, необходимо было оба растра весов перевести в векторный формат для дальнейшего присвоения пешеходным дорогам двух весовых значений, полученных из обеих наборов данных. Для начала значения в ячейках были перведены в целочисленный формат,

поскольку инструмент векторизации растра не работает с вещественными значениям в ячейках. Затем в программном продукте ArcGIS инструментом «Raster to polygon» была произведена автоматическая векторизация двух растров весов двух наборов данных. В результате, были получены два полигональных shape-файла, где каждый объект – полигон со значением, взятого из ячейки растра весов.

Следует напомнить, что все пешеходные дороги были разделены на отрезки, максимальная длина каждого составляла 20 метров. Для присвоения значений векторизованного растра всем частям пешеходных дорог был использован инструмент «Spatial Join» в программном продукте ArcMap (Рис.36). Если дорога пересекала несколько полигонов векторизованного растра, то ей присваивалось среднее значение пересекающих ее полигонов. После окончания работы инструмента в программе QGIS при помощи «калькулятора полей» весовые значения были возведены в степень -1. Эта операция требовалась для работы выбранного алгоритма при построении пешеходных маршрутов (см. раздел 3.3).

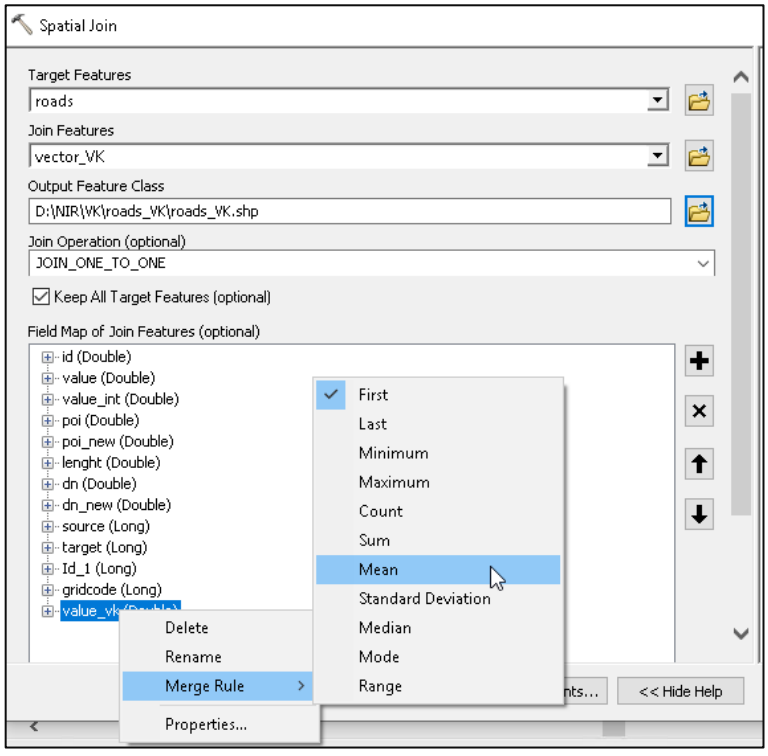


Рис.36. Инструмент «Spatial Join».

Таким образом, в атрибутивную таблицу shape-файла дорожной сети было добавлено два поля с оценками, полученных из двух векторизованных растров. Shape-файл дорожной сети (граф дорог) был импортирован в БД. В БД уже находилась геометрически идентичный граф дорог. Было необходимо присвоить значения из двух полей с оценками первоначальному графу, поскольку именно он используется при построении маршрутов в приложении. Для этой задачи были созданы два поля для двух

видов оценок в исходной таблице графа дорог (Рис.37). Затем значения были занесены в эту таблицу при условии, что поля «id» в целевой таблице и таблице с новыми оценками идентичны (Рис.38).

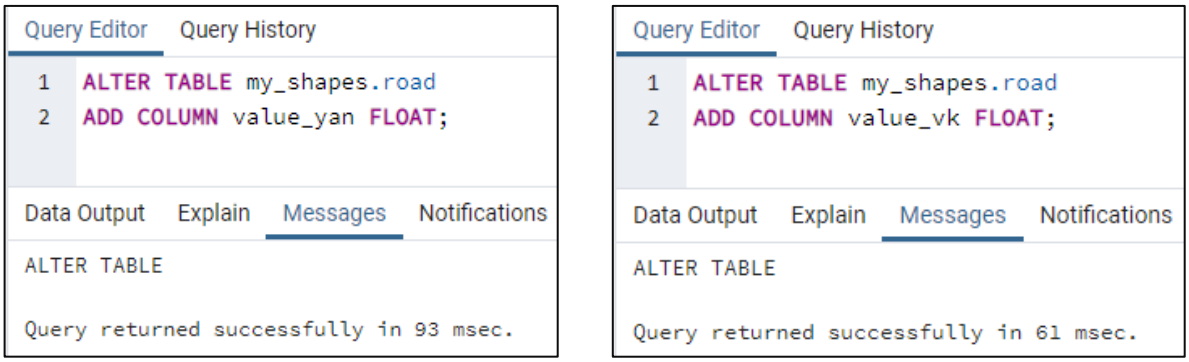


Рис. 37. Создание полей для таблицы дорожной сети.

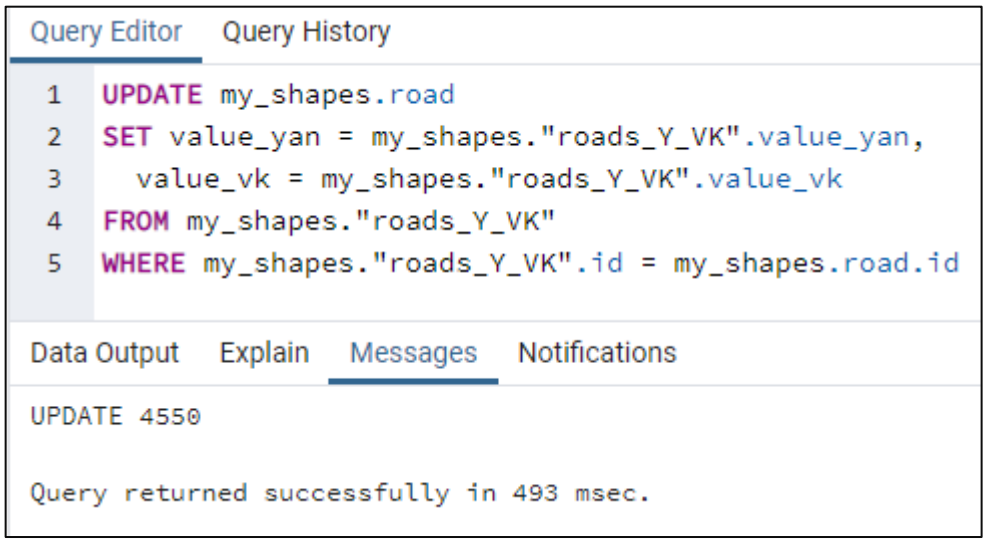


Рис. 38. Заполнение полей для таблицы дорожной сети.

Таким образом, были подготовлены данные для исследования «самых интересных» прогулок, взятые из двух источников («Яндекс.Карты» и «ВКонтакте»). Рассмотрены два подхода к применению данных о фотографиях и реализован один из них. Было присвоено количество фотографий для каждой достопримечательности. Построены два растра весов сумарного влияния достопримечательностей для каждого ребра графа дорог. Были созданы программные коды для проведения геопространственного анализа исходных данных. Осуществлены запросы к таблица БД для присвоения новых весовых значений дорожному графу.

2.3. Сравнительный анализ дорожной сети по полученным результатам

После внесения двух типов весовых значений в таблицу дорожной сети, были визуально проанализированы три вида значений по каждому из источников данных. Для этого в программе QGIS было создано подключение к БД и добавлен таблица (слой)

дорожного графа на карту программы. Граф дорог отображался с использованием цветовой шкалы. Анализируя данные «Яндекс.Карты», полученные в разные промежутки времени, можно сделать выводы (Рис.39, 40):

– максимальное суммарное влияние всех достопримечательностей (далее – оценка) возросла с 11 до 17;

– оценки в определенных областях уменьшились (обасти, которые входили в диапазон от 3 до 5, на 2020 год находятся в промежутке от 1 до 3);

– появились значения оценки, меньших 1, которые не отображены на Рис.40;

– есть области, у которых оценка увеличилась (Михайловский сквер, Летний сад (благодаря добавлению новых данных сервисом), Марсово поле);

Анализируя совместно данные «Яндекс.Карты» и «ВКонтакте» за 2020 год, замечено, что (Рис.40, 41 и 42):

– дапазон значений существенно отличается;

– оценки одних и тех же объектов разные (например, Михайловский сквер, Михайловский сад, Летний сад, набережная реки Мойка, Марсово поле);

– оценка для Дворцовой площади и некоторых частей Дворцовой набережной совпадают;

– количество фотографий не везде коррелирует с популярностью мета по данным «Яндекс.Карты» (например, Марсово поле);

Следует учесть тот факт, что данные «ВКонтакте» присваивались достопримечательностям, и если достопримечательность по какой-либо причине отсутсвовала в данных, то число фотографий разделялось между всеми ближайшими. Это могло повлиять на общий результат. Несмотря на большое количество фотографий в Летнем саду, они были разделены между огромным количеством памятников в нем. Как результат, общая привлекательность Летнего сада оказалась невысокой.

Рис.39. Значения весов суммарного влияния всех достопримечательностей для каждого участка пешеходной дороги по данным «Яндекс.Карты» за 2019 г., оценка.

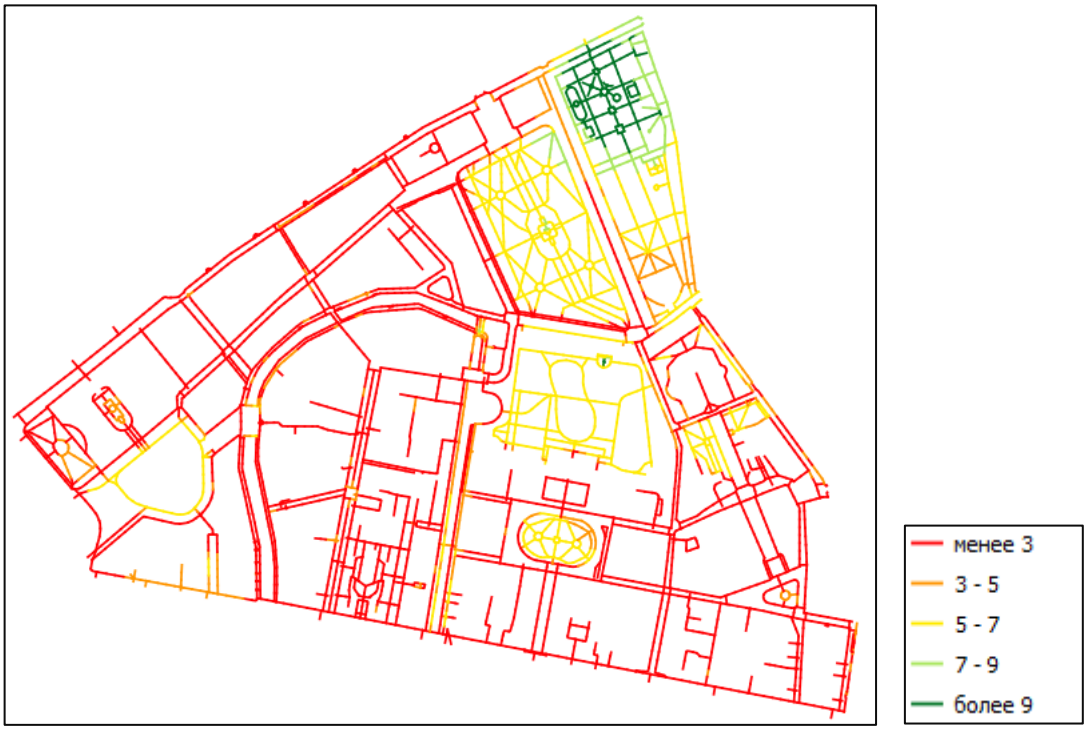


Рис.40. Значения весов суммарного влияния всех достопримечательностей для каждого участка пешеходной дороги по данным «Яндекс.Карты» за 2020 г., оценка.

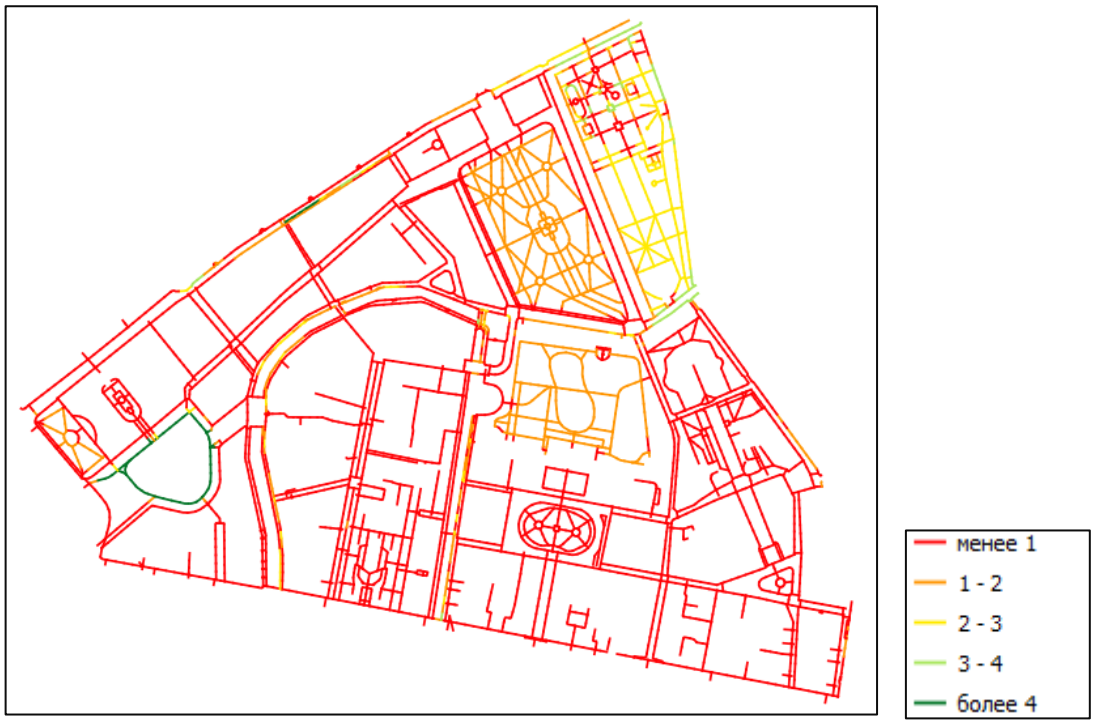


Рис.41. Значения весов суммарного влияния всех достопримечательностей для каждого участка пешеходной дороги по данным фотографий «ВКонтакте», опубликованных в июле 2019 г., оценка.

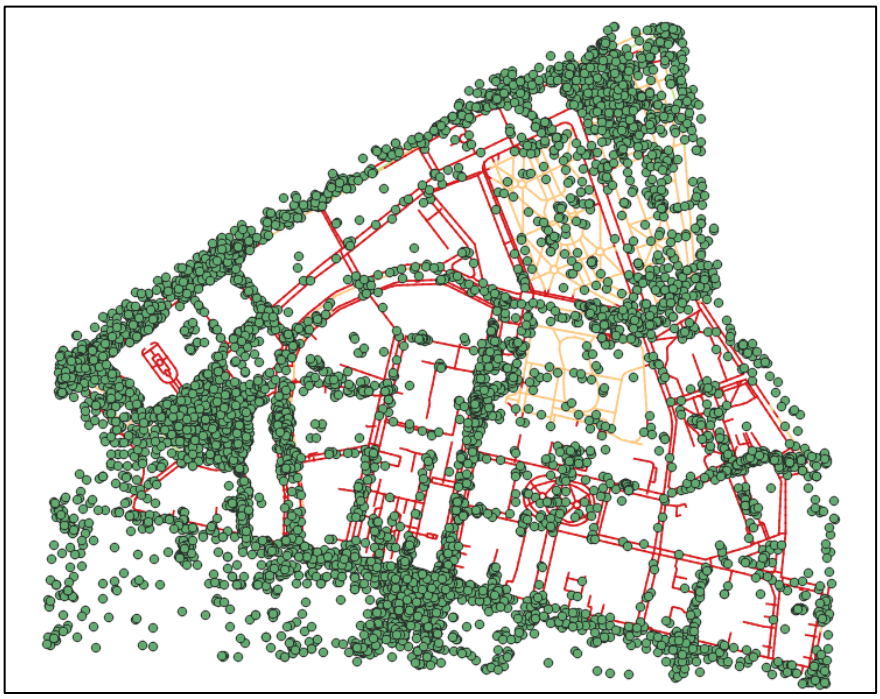


Рис.42. Количество фотографий и значения оценок по данным «ВКонтакте» за июль 2019 года.

2.4. Построение «самых интересных» прогулок на основе различных источников данных

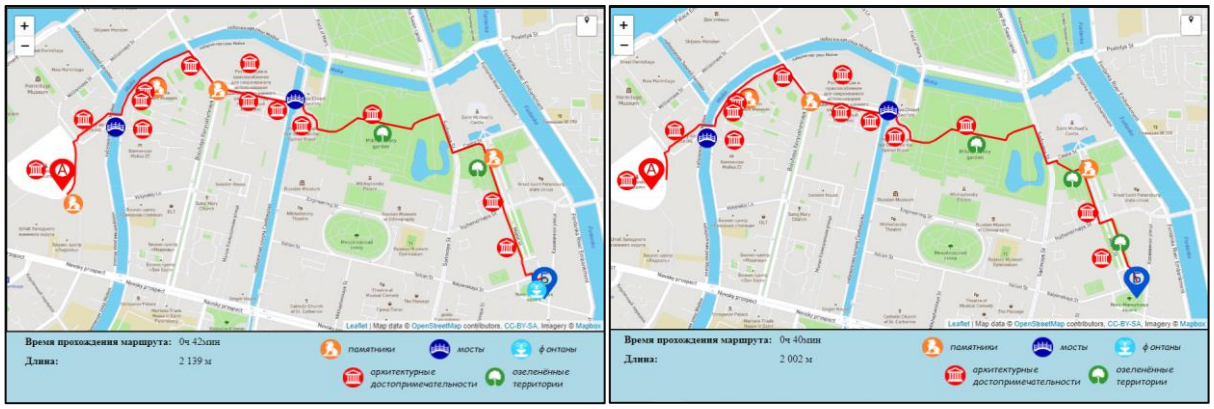
Для того чтобы построить «самые интересные» маршруты, было задействовано веб-приложение, созданное в рамках выпускной квалификационной работы. Были построены несколько вариантов прогулок, построенные по алгоритму Дейкстры (см. раздел 3.3). В качестве весовых значений в таблице дорожного графа были взяты поля с оценками из источника:

– «Яндекс.Карты», полученные на апрель 2019 года;

– «Яндекс.Карты», полученные на апрель 2020 года;

– «ВКотакте», в которой фотографии были опубликованы в июле 2019 года.

В результате, на Рис.43а-в представлены три варианта «самых интересных» прогулок с одинаковыми крайними точками, но отличающиеся по источнику данных для оценок.

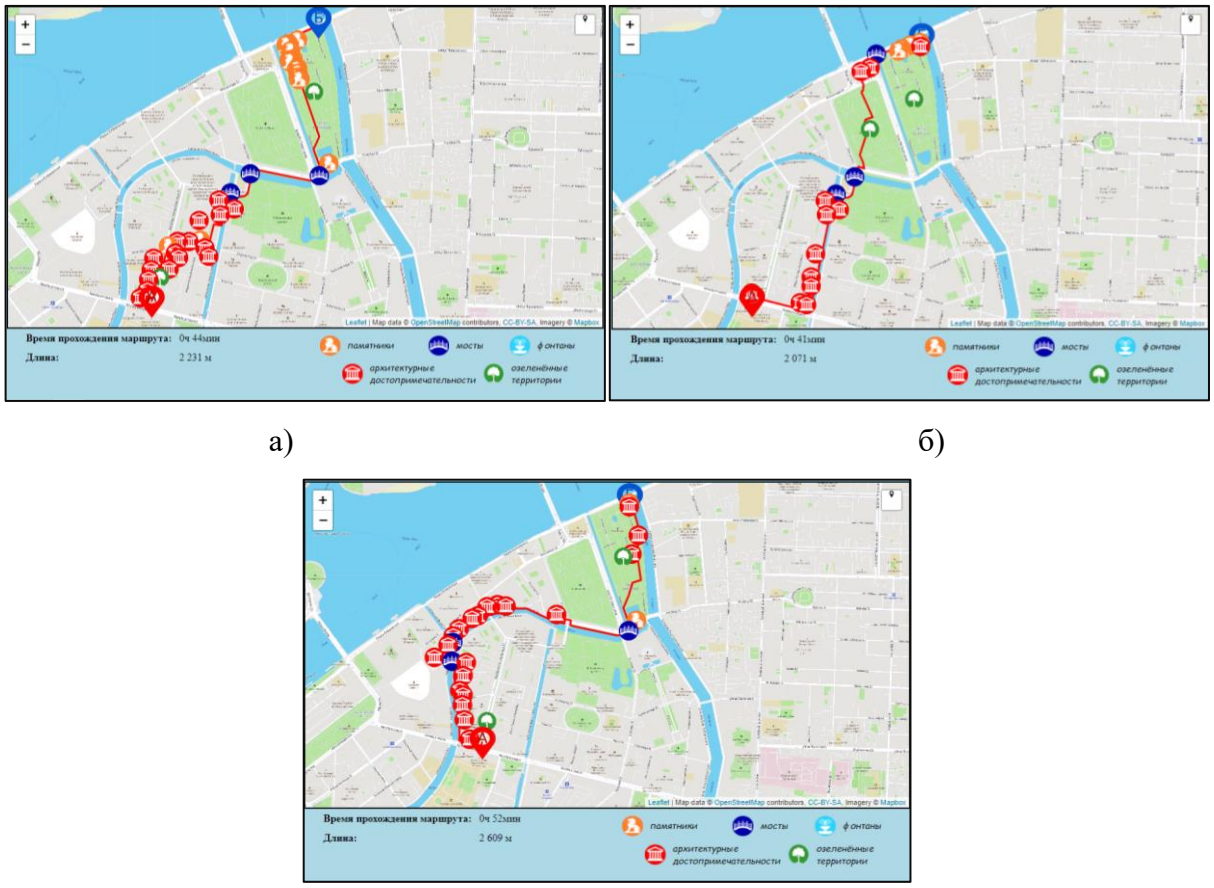


а) б)

в)

Рис.43. «Самый интересный» маршрут. а) – по данным «Яндекс.Карты» на апрель 2019 года; б) – по данным «Яндекс.Карты» на апрель 2020 года; в) – по данным «ВКонтакте» на июль 2019 года.

На Рис.44а-в представлены еще один набор «самых интересных» прогулок, построенных между одинаковыми крайними точками, но отличающиеся по входным данным.



а) б)

в)

Рис.44. «Самый интересный» маршрут №2. а) – по данным «Яндекс.Карты» на апрель 2019 года; б) – по данным «Яндекс.Карты» на апрель 2020 года; в) – по данным «ВКонтакте» на июль 2019 года.

Проанализировав пешеходные маршруты, построенные по двум парам крайних точек, можно сделать следующие заключения:

– маршруты по двум наборам данным «Яндекс.Карты» не всегда отличаются (Рис.43а,б). Маршруты по большей части идентичные, есть незначительные отличия.;

– маршруты по данным «ВКонтакте» строятся как можно ближе или по набережным (это следует из концентрации фотографий, сделанных на набережных (см. Рис.42)). Они заметно отличаются от маршрутов по данным «Яндекс.Карты»;

Из этого следует вывод, что в условиях отстствия актуальных данных об оценках набережных по популярности у сервиса «Яндекс.Карты» данные о фотографиях

«ВКонтакте» могли бы быть использованы. Высокая концентрация геотегов вдоль набережных мола бы привнести корректные результаты в построение «самых интересных» прогулок. Прогулки, построенные по двум наборам данных «Яндекс.Карты» не везде отличаются, это зависит от изменения оценки по популярности достопримечательности и добавление новых. В итоге, данные «ВКонтакте» использовались для оценок по популярности линейных достопримечательностй в виде набережных, «Яндекс.Карты» – для оценок точечных и площадных достопримечательностей.

Таким образом, были получены наборы данных из двух источников и проанализированы их особенности при разных условиях загрузки. Выполнена цепочка геообработки даннх, состоящая из присвоения числа фотографий каждой достопримечательности, создания растров весов и присвоения двух видов оценок ребрам графа дорог. По результатам обработки были построены и проанализированы «самые интересные» маршруты и сделан вывод об использовании разных источников для разных геометрических примитивов достопримечательностей.

Глава 3. Разработка мобильного приложения по построению пешеходных маршрутов

3.1. Построение клиент-серверной архитектуры приложения

Созданное приложение основано на клиент-серверной архитектуре (Рис.45). В роли сервера выступает удаленный компьютер, а в роли клиента – мобильное приложение. Клиент отправляет запрос на сервер с указанием IP-адреса и порта компьютера. Серверное приложение получает этот запрос, в котором может содержаться, например, координаты крайних точек, идентификаторы выбранного пользователем маршрута или достопримечательности для получения подробной информации. После того, как серверное приложение представило полученный запрос в соответствующем виде, он отправляется для выполнения в БД. Связь между БД и серверным приложением осуществляется с использованием технологии JDBC.

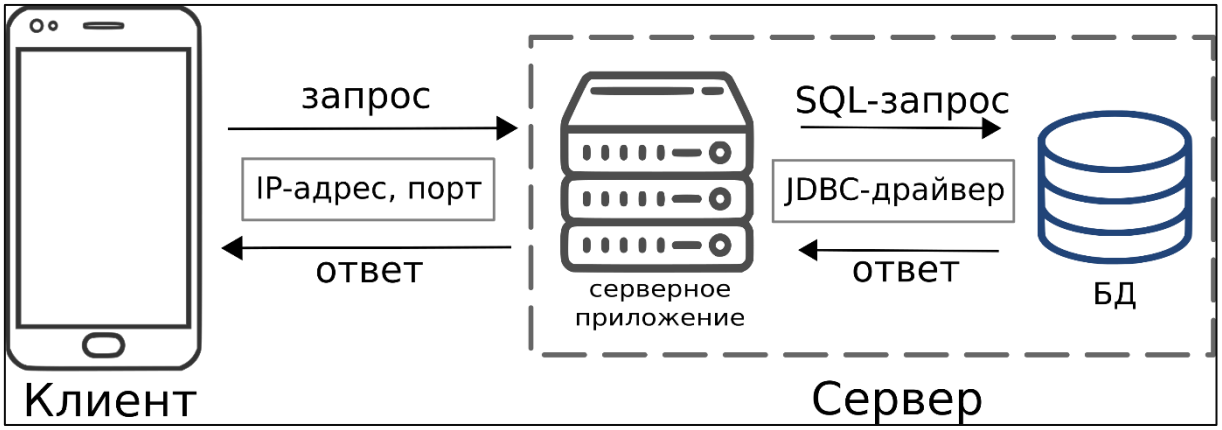


Рис.45. Схема клиент-серверной архитектуры приложения.

JDBC (от англ. Java Database Connectivity, пер. подключение к БД Java) – это API для языка программирования Java, который определяет, как клиент может получить доступ к БД[17]. JDBC предоставляет методы для запроса и обновления данных в БД и ориентирован на реляционные БД. Используя интерфейс JDBC API, был настроен доступ к используемой БД.

Чтобы использовать JDBC API с конкретной СУБД, необходим драйвер, который будет посредником между серверным приложением и БД. Драйвер PostgreSQL JDBC (сокращенно PgJDBC) позволил серверному приложению, написанному на языке Java, подключаться к БД (Рис.46). PgJDBC – драйвер с открытым исходным кодом, который взаимодействует по собственному сетевому протоколу PostgreSQL – jdbc:postgresql.

[17] https://www.educba.com/jdbc-architecture/

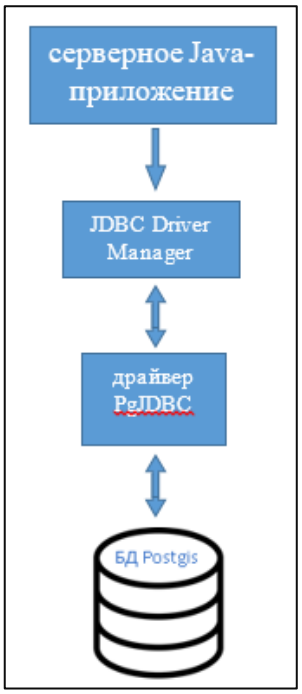


Рис.46. Взаимодействие приложения и БД.

Сервер отправляет запрос на подключение к БД с помощью специального метода (*getConnection*) и драйвера PgJDBC, предоставляемого посредством JDBC. В этом методе в качестве аргументов передаются: IP-адрес компьютера, порт, по которому прослушивается БД, ее название, имя пользователя и пароль. Так как серверное приложение и БД находятся на одном компьютере, то IP-адрес = «127.0.0.1»

Доказательством удачного подключения к БД являются создание поля *Connection* (связи между приложением и БД) и вывод соответствующего сообщения. Ответами от БД могут быть координаты вершин маршрута, его ребра, информация о достопримечательностях (название, вид, описание, ссылка на веб-ресурс, путь к фотографии на компьютере). Серверное приложение обрабатывает ответ от БД для отправки его клиенту в необходимой форме.

Доказательством удачного подключения клиента к серверу является выполнение запроса (построение маршрута). Пользователь приложения уведомляется в случае неудачной попытки соединения с сервером. Это может произойти по причине отсутствия Интернет-соединения на мобильном устройстве.

Таким образом, был разработан «каркас» для создания приложения. Настроена связь между клиентом и сервером. Установлено соединение между серверным приложением и БД посредством технологии JDBC. Определены возможные варианты запросов и ответов между тремя составляющими приложения.

3.2. Разработка части приложения на стороне клиента

Создание клиентской части приложения осуществлялась в Android Studio с применением современных язык и технологий: Java и расширяемого языка разметки XML.

XML (от англ. eXtensible Markup Language) – это язык разметки, который определяет набор правил кодирования документов в формате, удобочитаемом как для человека, так и для машины. XML-документ представляет собой обычный текстовый файл, в котором последовательность и вложенность элементов интерфейса (виджетов) определяют структуру документа и его содержание (Одиночкина, 2013). У каждого виджета есть свои атрибуты для настройки стиля окна приложения (ширина, высота, расположение, возможность реагирования на нажатие и так далее). Элементы и их атрибуты XML-документа определяли интерфейс приложения.

При разработке интерфейса приложения были добавлены следующие элементы (Рис.47а-в):

– название приложения (номер 1);

– кнопка вызова всплывающего меню для выбора маршрута (номер 2). По умолчанию выбран «самый интересный» маршрут;

– поисковая строка для ввода адреса (номер 3);

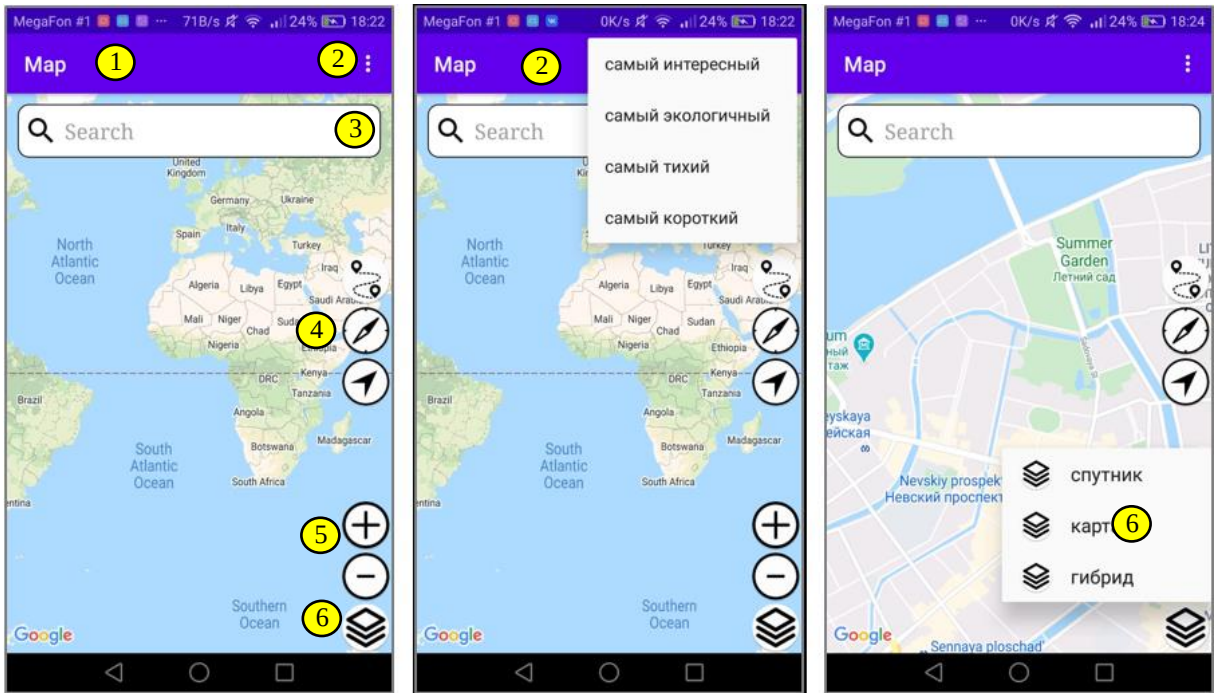
– навигационный блог кнопок (номер 4) (сверху вниз): кнопка для перехода в режим построения маршрута; компас – для возвращения расположения карты в положение север-юг. Стрелка компаса поворачивается вместе с картой при изменении ориентации (Рис.50); кнопка «мое местоположение», имеет три режима: первый – при нажатии устанавливается экстент карты в месте нахождения пользователя, второй – при втором нажатии экстент карты будет изменяться по ходу движения пользователя, третий – при третьем нажатии сохраняется второй режим плюс карта ориентируется по направлению положения мобильного устройства;

– блок кнопок масштабирования карты (номер 5). В дополнении к этим инструментам, пользователь может масштабировать карту двумя пальцами;

– всплывающее меню выбора фоновой карты (номер 6). Реализовано подключение трех веб-карт в приложении для комфортной навигации пользователя (Рис.8);

– значок пешехода (Рис.48). Появляется при переходе в режим построения маршрута. Цвет фигуры говорит о выбранном варианте маршрута. Красный фон

– выбран «самый интересный» маршрут, зеленый – «самый экологичный», синий – «самый тихий», темно-серый – «быстрый».



а) б) в)

Рис.47. Элементы главного окна приложения.

Рис.48. Разновидности значков пешехода.

Написание функций на языке Java позволило сделать окна приложения интерактивными. Например, нажатие на любую кнопку или передвижение маркеров сопровождалось последующими действиями, описанные в коде Java. Все обработчики подобных событий хранятся в одном файле в виде единого класса Java, адрес на который прописан в *.xml* файле с названием AndroidManifest. AndroidManifest предоставляет операционной системе мобильного устройства основную информацию о приложении[18]. В нем содержатся список необходимых разрешений для определения местоположения, доступа к Интернету и API- ключ для подключения веб-карт в приложении.

При написании программного кода на Java со стороны клиента были созданы методы, обеспечивающие:

[18] https://developer.android.com/guide/topics/manifest/manifest-intro

– добавление карт в приложение;

– отображение всплывающего меню выбора фоновой карты;

– добавление/перемещение маркеров (крайних точек маршрута) на карту (Рис.49);

– работу компаса (Рис.50);

– масштабирование, изменение ориентации карты;

– изменение экстента карты при добавлении маркера по введенному адресу;

– функционирование GPS-навигации, отображение местоположения пользователя на карте;

– выбор вида маршрута перед его построением (Рис.51);

– работу прямого и обратного геокодирования. Реализован метод, для того чтобы поменять точки местами, и их адреса в поисковой строке тоже поменяются местами (Рис.52);

– изменение адресов крайних точек при их перемещении;

– построение/перестроение маршрута, изменение цвета его отображения;

– установка экстента карты по линии маршрута, создание анимации при смене экстента;

– отображение объектов (например, достопримечательностей) вдоль маршрута;

– определение значка для отображения каждого объекта по маршруту, исходя из его типа;

– создание всплывающего окна для просмотра информации об объектах вдоль маршрута;

– просмотр дополнительной информации о культурных объектах (текстовое описание, фотографии, веб-ссылка);

– отображение длины, количества достопримечательностей вдоль построенного маршрута;

– подсчет и отображение времени прогулки;

– удаление с карты всех расположенных на ней элементов, очистка адресной строки;

– подключение к серверу;

– отправление запроса, получение ответа от сервера;

– преобразование фотографии и битового формата (в таком формате приходит ответ от серверного приложения) в пиксельное изображение для отображения;

– вывод сообщений о возможных ошибках во время работы приложения.

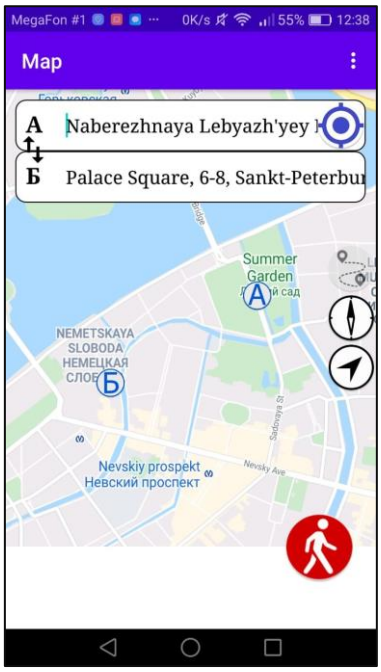


Рис. 49. Добавление маркеров и адресов.

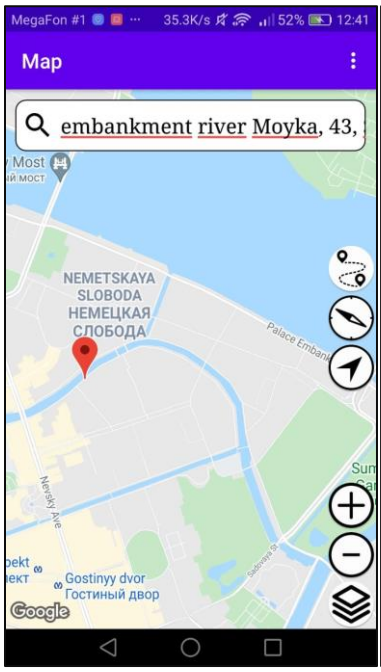


Рис.50. Реализация функции компаса в приложении.

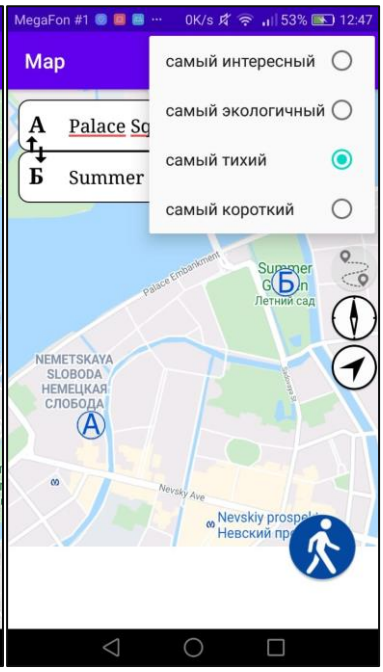
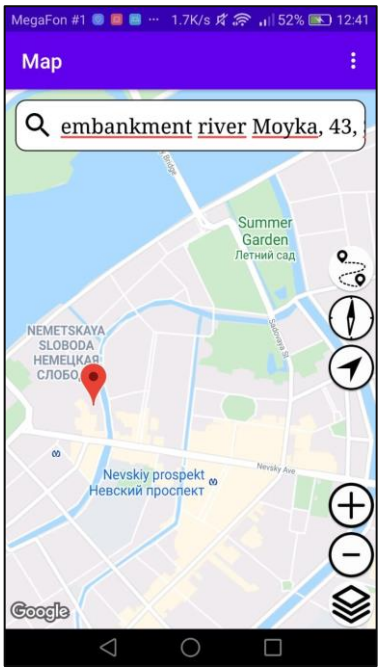


Рис.51. Выбор маршрута.



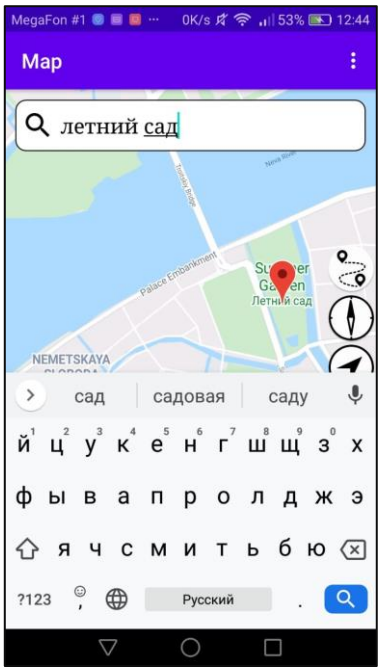


Рис.52. Прямое, обратное геокодирование.

Пример кода реализации функции компаса представлен на Рис.53.

```
private void rotateToNorth() {
    CameraPosition cameraPosition = this.googleMap.getCameraPosition();
    int duration = 500;
    this.animateCamera(cameraPosition.target, cameraPosition.zoom, bearing: 0, tilt: 0, duration);
    MapsActivity.this.activityMapsBinding.imageButtonCompassArrow.animate().rotation(0).setDuration(duration).start();

    if (MapsActivity.this.locationState >= 2) {
        MapsActivity.this.locationState = 1;
        MapsActivity.this.activityMapsBinding.imageButtonMyLocation.setImageDrawable(
                MapsActivity.this.getDrawable(R.mipmap.ic_my_location));
    }
}
```

Рис.53. Программный код для реализации функции компаса.

В результате разработки клиентской части приложения, были созданы виджеты главного окна мобильного приложения. Осуществлены методы взаимодействия пользователя с приложением. Список методов включает в себя возможности, начиная от выбора фоновых карт до просмотра информации по маршруту. Реализация выполнялась с использованием языков Java и XML. Примеры построения маршрутов описаны в разделе 3.4.

3.3. Создание серверной части приложения

Серверная часть приложения включает в себя серверное приложение и БД, которые расположены на одном компьютере (Рис.45). Разработка осуществлялась с использованием языков программирования Java и структурированных запросов SQL в ПО IntelliJ IDEA. Серверное приложение состоит из трех классов. Первый из них – Server.java – в нем прописан код для создания подключения между БД и серверным приложением (Рис.54), осуществления ожидания получения запроса от клиентской части приложения (Рис.55). Метод *accept()* является ключевым в этом коде – при его реализации сервер ожидает запроса от клиента через соединение (*this.connection*). В случае успешного создания объекта *worker* устанавливается соединение между сервером и клиентом.

```
public boolean connectDB() {
    this.connection = null;
    boolean status = false;
    try {
        Class.forName("org.postgresql.Driver");
        this.connection = DriverManager.getConnection(this.url, this.user, this.pass);
        status = true;
        System.out.println("connected:" + status);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return status;
}
```

Рис.54. Программный код для соединения с БД.

```
this.serverSoket = new ServerSocket(this.portServer);
System.out.println("Start server on port: " + this.portServer);
while(true) {
    ConnectionWorker worker = null;
    try {
        worker = new ConnectionWorker(this.serverSoket.accept(), this.connection);
        System.out.println("Get client connection");
        Thread t = new Thread(worker);
        t.start();
    } catch (Exception e) {
        System.out.println("Connection error: " + e.getMessage());
    }
}
```

Рис.55. Программный код для осуществления связи между клиентской частью и серверным приложением.

Второй класс серверного приложения – ConnectionWorker.java – выполняет обработку запросов при успешном соединении между всем частями приложения. В нем прописаны запросы к БД (Рис.56) и их обработка для отправления ответа клиенту (Рис.57). Например, на Рис.56 продемонстрирована строка запроса к БД для получения вершин маршрута, его ребер в виде геометрических объектов и длин ребер. Исходными данными для запроса являются идентификаторы крайних точек прогулки и ее тип. На Рис.57 показан процесс создания массивов ребер и вершин маршрута, вычисление его длины.

```
private String getRouteQueryString(String idA, String idB, String cost) {
    return "SELECT ST_AsText(ST_Transform(the_geom,4326)),(ST_Length(geom)) AS length, edge " +
            "FROM ((SELECT *FROM pgr_dijkstra(' " +
            "SELECT id AS id, source AS source, target AS target, " + cost + "::float8 AS cost " +
            "FROM my_shapes.road', " + idA + ", " + idB + ", " +
            "false))  as DJ1 INNER JOIN my_shapes.road as road " +
            "ON DJ1.edge = road.id) AS DJ INNER JOIN my_shapes.road_vertices_pgr AS Tochki  " +
            "ON DJ.node = Tochki.id";
}
```

Рис.56. Запрос к БД.

```
// get points for route:
ResultSet resultRoute = statement.executeQuery(this.getRouteQueryString(idA, idB, routeType));

// parse request:
ArrayList<Long> segmentsID = new ArrayList<~>();
ArrayList<double[]> vertices = new ArrayList<double[]>();
double length = 0.0;
while (resultRoute.next()) {
    String lengthSegment = resultRoute.getString( columnLabel: "length");
    length += Double.parseDouble(lengthSegment);

    String point = resultRoute.getString( columnLabel: "st_astext");
    String[] tokens = point.substring(6, point.length() - 1).split( regex: " ");
    double[] coords = new double[2];
    for (int i = 0; i < 2; ++i) {
        coords[i] = Double.parseDouble(tokens[i]);
    }
    vertices.add(coords);
    segmentsID.add(Long.parseLong(resultRoute.getString( columnLabel: "edge")));
}
```

Рис.57. Обработка результата запроса от БД.

Третий класс – Launcher.java – запуск серверного приложения.

Помимо запроса, представленного на Рис.56, к БД осуществлялись следующие запросы: определение идентификатора, координат крайней точки маршрута, получение названия, описания достопримечательности, пути расположения фотографии на компьютере. Достопримечательности отображаются вдоль маршрута, если они находятся от него на расстоянии, не превышающее 20-ти метров. Был написан запрос,

создающий временную буферную зону вокруг маршрута. Затем находится геометрическое пересечение точечных объектов (достопримечательностей) с полигоном (буферной зоной). После выделения нужных точечных объектов определяется необходимая информация о них. Фотография достопримечательности преобразуется в битовый формат и отправляется клиентской части приложения, где она преобразуется в пиксельное изображение.

Прогулки в мобильном приложении строятся на основе алгоритма Дэйкстры, описанный в источнике (Кормен, 2005) (Рис.58).

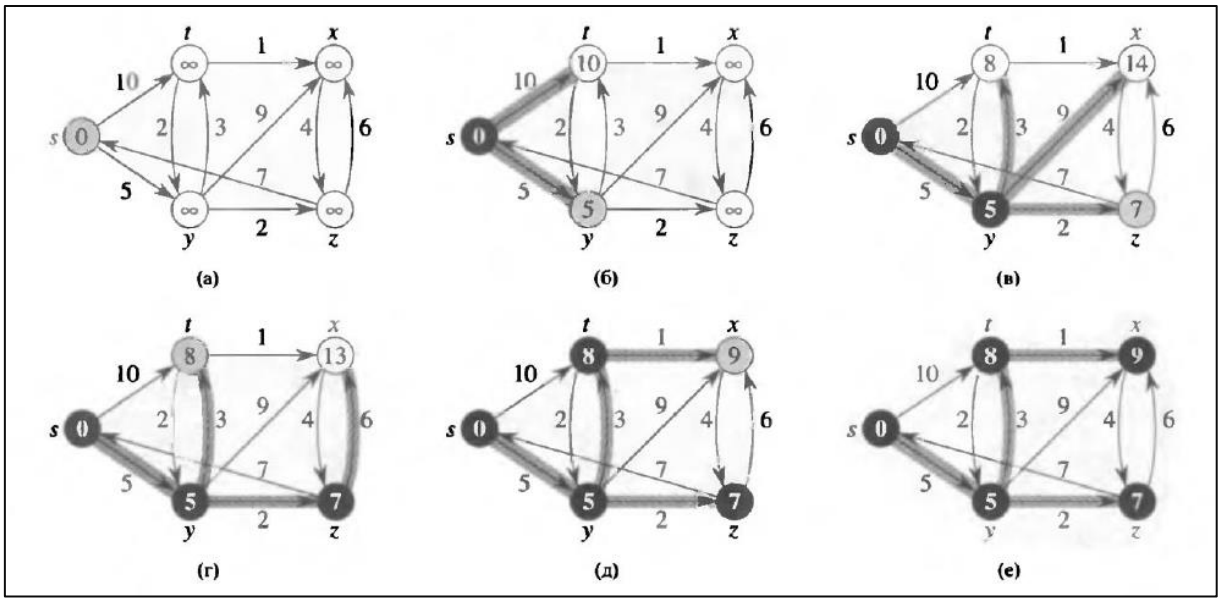


Рис.58. Алгоритм Дейкстры. Источник: (Кормен, 2005).

В результате разработки серверного приложения, была настроена связь между БД и серверным приложением, серверным приложением и клиентской частью. Осуществлена возможность запуска серверного приложения. Реализованы методы обработки запросов и ответов.

3.4. Примеры построений пешеходных маршрутов в приложении

После открытия приложения пользователю предлагается выбрать крайние точки прогулки. Для этого он вводит адрес конечной точки в адресной строке либо указывает точку на карте (Рис.59). Во втором случае адрес появляется в поисковой строке автоматически.

Находясь в режиме построения маршрута, адрес красного маркера становится адресом конечной точки «Б» (Рис.60). При этом кнопка для перехода в режим построения маршрута становится неактивной. Курсор в поисковой строке перемещается на место ввода адреса точки «А», и пользователю предлагается выбрать начальную точку на карте

или указать ее адрес. В качестве начальной точки можно выбрать местоположение пользователя, нажав кнопку в адресной строке для точки «А», справа. После указания местоположения на карте появляется начальная точка «А» (Рис.61).

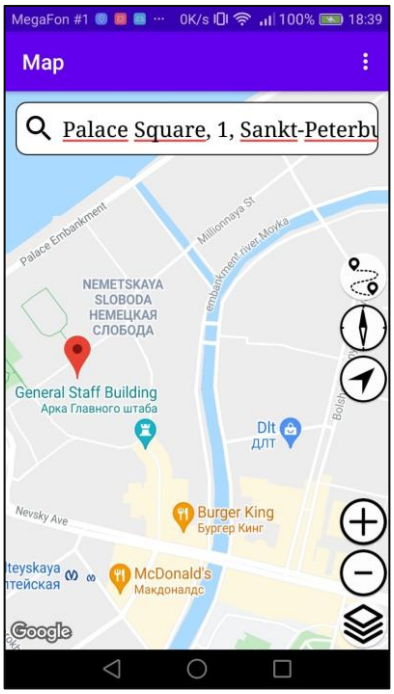


Рис. 59. Выбор точки на карте.

Рис.60. Приглашение к вводу точки «А».

Рис.61. Выбор крайних точек маршрута.

При нажатии на значок пешехода на экране внизу справа маршрут будет построен. Информация о типе маршрута, его длине и времени прохождения отображается внизу экрана (Рис.62,63). На карте появляются местоположения объектов вдоль маршрута – достопримечательности, озелененные территории и так далее. Пользователь может ознакомиться с условными обозначениями объектов на карте, выдвинув экран вверх (Рис.64).

Нажав на один из значков на карте, пользователь наблюдает более подробную информацию в дополнительном окне приложения внизу экрана (Рис.65,66). Здесь он может ознакомится с кратким описанием объекта, его фотографией. В этом окне у пользователя есть возможность перейти по веб-ссылке (Рис.67) с более подробной информацией об объекте, представленной на веб-ресурсе[19] (Рис.68).

[19] https://ru.wikipedia.org/wiki/

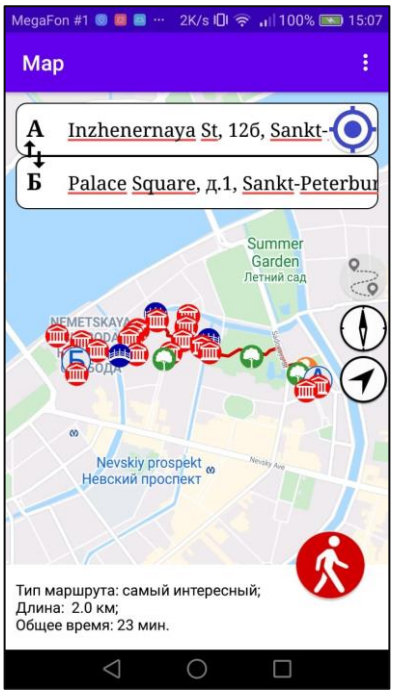


Рис. 62. Результат построения «самой интересной» прогулки.

Рис.63. Реализация функции компаса в приложении.

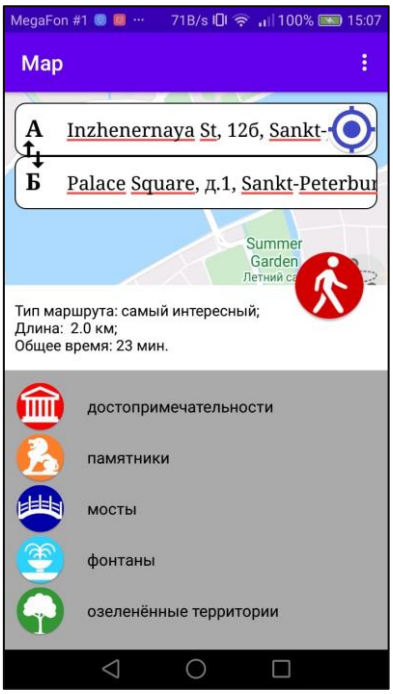


Рис.64. Условные обозначения.

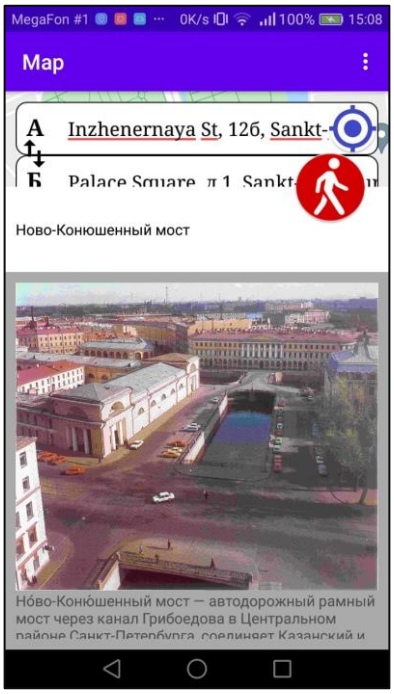


Рис. 65. Информация и мосте.

Рис.66. Информация об озелененной территории.

Рис.67. Возможность перехода по веб-ссылке.

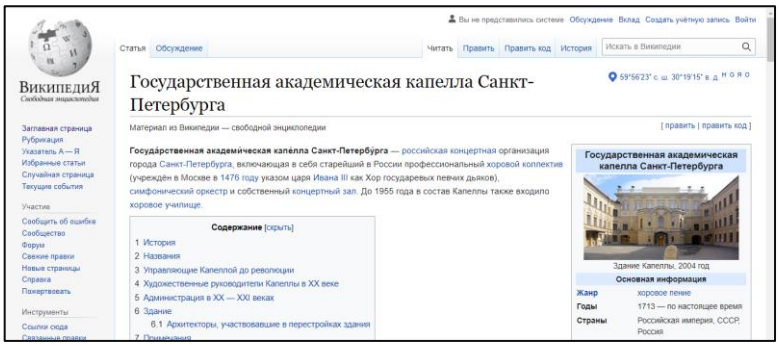


Рис.68. Результат работы веб-ссылки для просмотра более подробной информации об объекте.

Выбрав иной тип прогулки, маршрут перестраивается (Рис.69,70,73). Изменяется цвет значка пешехода, тип, длина, время маршрута в информационном окне. Пользователь может управлять отображением объектов на карте, нажав на соответствующий значок в окне условных обозначений. Выключение отображения объекта на карте сопровождается изменением яркости значка в легенде. Есть возможность выключить визуализацию всех или нескольких достопримечательностей (Рис.71,72,73).

При передвижении по маршруту режим навигации будет активен при условии предоставления разрешения пользователем на определение местоположения мобильного устройства. В режим навигации входит: компас, определение местоположения (в том числе при нажатии на кнопку в адресной строке для точки «А»), три режима навигации при разных количествах нажатии на значок «мое местоположение».

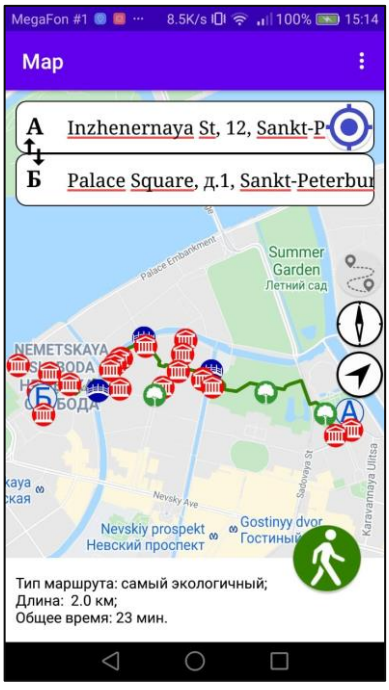


Рис.69. «Самый экологичный» маршрут.

Рис.70. «Самый тихий» маршрут.

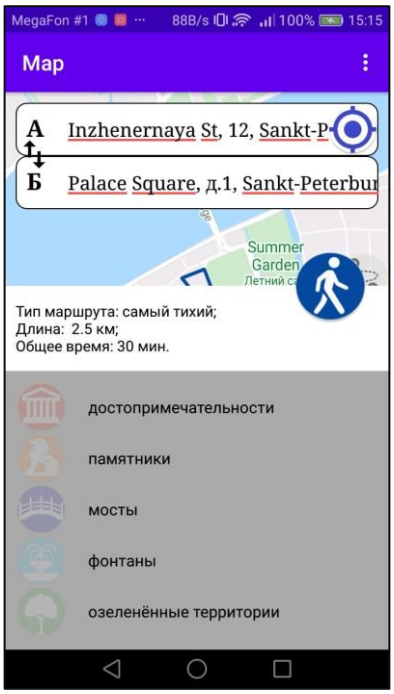


Рис. 71. Добавление маркеров и адресов.

Рис.72. Реализация функции компаса в приложении.

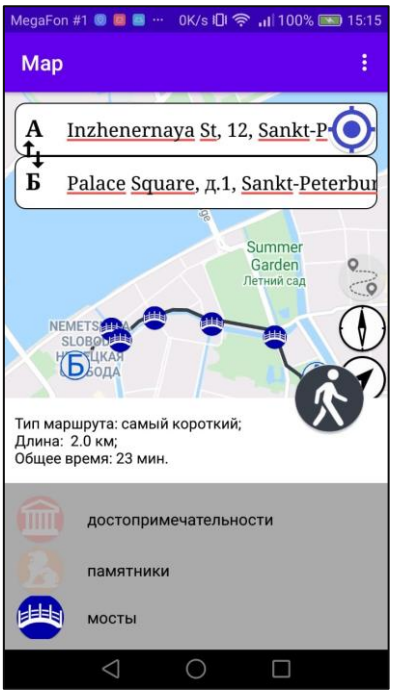


Рис.73. Выбор маршрута.

В ходе тестирования было замечено, что не все достопримечательности имеют фотографии в описании. Более того, не у всех есть текстовое описание (Рис.74). Причина этого явления описана в разделе 2.1.

Рис.74. Пример отсутствия описания достопримечательности.

Таким образом, тестовое построение четырех типов маршрутов показало работоспособность мобильного приложения. Продемонстрировано изображение оформления окон приложения, их функциональность, интерактивность элементов

приложения по нажатию на них. Показаны примеры отображения подробной информации о достопримечательностях, краткой информации о маршруте.

Таким образом, разработка клиент-серверной архитектуры обеспечила создание по отдельности двух главных частей приложения. Была выполнена разработка клиентской части с использованием возможностей двух языков, каждый из которых выполняет свою функцию в приложении. Создание серверного приложения и подключение к БД обеспечило решение главной задачи созданного мобильного приложения – построение различных пешеходных маршрутов. В конечном итоге, построение четырех видов пешеходных прогулок в приложении является результатом его функционирования.

Заключение

Подводя итог, в процессе создания прототипа мобильного приложения было выполнено следующее:

- проанализированы исследования и готовые сервисы по построению прогулок;
- выбрано программное обеспечение и языки программирования для реализации проекта на различных стадиях;
- собраны данные по 600 объектам, отображаемым вдоль строящихся пешеходных маршрутов;
- получены и проанализированы данные из трех источников;
- построены тестовые маршруты для выявления особенностей исходных данных;
- разработан алгоритм применения данных о фотографиях социальной сети для определения весовых значений элементов сетевой модели данных пешеходных дорог;
- созданы программные коды на языке Python для проведения предварительной обработки исходных данных;
- разработана клиент-серверная архитектура приложения;
- написан программный код для функционирования клиентской части приложения средствами Java и XML;
- создана серверная часть приложения.

Созданное приложение предоставляет возможность построения четырех типов маршрутов на тестовом участке центра города Санкт-Петербург: «самый интересный», «самый экологичный», «самый тихий», «самый короткий». Дополнительно пользователь может выбирать фоновую карту окна приложения, пользоваться функциями геокодирования. В процессе работы приложения пользователь получает информацию о необходимом для прогулки времени, длине пути. Главное отличие приложения от созданного ранее веб-приложения заключается в реализации его в мобильном формате. Такое представление обеспечит удобное пользование приложением на улице, по пути следования по маршруту. В дополнении к этому, в мобильном приложении реализованы новые функции, такие как просмотр информации о достопримечательностях (текстового описания, фотографии), возможность перехода по веб-ссылки для детального ознакомления с объектом, навигации по маршруту с использованием встроенного GPS-приемника в мобильном устройстве. Пользователь может просматривать местоположения пунктов проката велосипедов и самокатов. В отличии от веб-версии, в созданном приложении используются обновленные данные о достопримечательностях,

применены новые источники данных. Это способствовало улучшению построения «самых интересных» прогулок и реализации новых функций. Разработанное приложение может функционировать на любых устройствах с операционной системой Android.

В рамках развития проекта планируется усовершенствовать функцию просмотра информации о достопримечательности путем добавления недостающего текстового описания, его исправление, добавления отсутствующих фотографий. Предполагается реализация мобильного приложения в качестве социального проекта по развитию туризма в Санкт-Петербурге.

Литература

1. Al Shammas, T., Escobar, F. Comfort and Time-Based Walkability Index Design: A GIS-Based Proposal // Int. J. Environ. Res. Public Health, 2019. 16 p. DOI 10.3390/ijerph16162850.
2. Blair, S.N. Physical inactivity: The biggest public health problem of the 21st century // Br. J. Sports Med, 2009, Vol. 43, P. 1–2.
3. Brownson, R., et. al. Measuring the Built Environment for Physical Activity. State of the Science // Am. J. Prev. Med, 2009, Vol. 36, P. 99–123. DOI 10.1016/j.amepre.2009.01.005.
4. Butler, E., et. al. Identifying GIS measures of the physical activity built environment through a review of the literature // Journal of Physical Activity and Health, 2011. Vol. 1, P. 91–97. DOI 10.1123/jpah.8.s1.s91.
5. Coffee, N., et. al. Is walkability associated with a lower cardiometabolic risk? // Health & Place, 2013. Vol. 21, P. 163 – 169. DOI 10.1016/j.healthplace.2013.01.009.
6. Dannenberg, A., Cramer, T., Gibson, C. Assessing the walkability of the workplace: A New Audit Tool // The American Journal of Health Promotion, 2004. Vol. 20, P. 39 – 45. DOI 10.4278/0890-1171-20.1.39.
7. Debyser, A. Urban Mobility. Shifting towards Sustainable Transport Systems, 2014.
8. Ewing, R., et. al. Relationship between urban sprawl and physical activity, obesity, and morbidity // The American Journal of Health Promotion, 2003. Vol. 18. P. 567 – 582. DOI 10.4278/0890-1171-18.1.47.
9. Ewing, R., Handy, S. Measuring the unmeasurable: Urban design qualities related to walkability // Journal of Urban Design, 2009. Vol. 14, P. 65 – 84. DOI 10.1080/13574800802451155.
10. Forsyth, A. What is a walkable place? The walkability debate in urban design // Urban Des. Int, 2015. Vol. 20, P. 274 – 292.
11. Forsyth, A., et. al. Standards for Environmental Measurement Using GIS: Toward a Protocol for Protocols // Journal of Physical Activity and Health, 2006. Vol. 3, P. 241 – 257. DOI 10.1123/jpah.3.s1.s241.
12. Frank, L., et. al. The development of a walkability index: Application to the neighborhood quality of life study // British Journal of Sports Medicine, 2010. Vol. 44. P. 924 – 933. DOI 10.1136/bjsm.2009.058701.
13. Freeman, L., et al. Neighborhood walkability and active travel (walking and cycling) in New York City // J. Urban Health, 2012. Vol. 90. P. 575 – 585.
14. Giles-Corti, B., et. al. How Walkable is Melbourne? The Development of a Transport Walkability Index for Metropolitan Melbourne. University of Melbuorne: Melbourne, Australia, 2014.

15. Glazier, R., et. al. Development and Validation of an Urban Walkability Index for Toronto. Canada: Research Gate, 2012. P. 1 – 21. DOI 10.1016/j.socscimed.2008.05.028.
16. Gullón, P., Bilal, U., Cebrecos, A., Badland, H.M.; Galán, I.; Franco, M. Intersection of neighborhood dynamics and socioeconomic status in small-area walkability: The Heart Healthy Hoods project // Int. J. Health Geogr, 2017. Vol. 16. P. 1 – 9.
17. Guo, Q., et al. The development of urban night tourism based on the nightscape lighting projects-a Case Study of Guangzhou // Energy Procedia 5. 2010 International Conference on Energy, Environment and Development, 2011. P. 477 –481. ISSN 1876-6102.
18. Habibian, M., Hosseinzadeh, A. Walkability index across trip purposes // Sustainable Cities and Society Journal, 2018. Vol. 42. P. 216 – 225. DOI 10.1016/j.scs.2018.07.005.
19. Hall, C., Ram, Y. Walk score and its potential contribution to the study of active transport and walkability: A critical and systematic review // Transportation Research Part D: Transport and Environment, 2018. Vol. 61. P. 310 – 324. DOI 10.1016/j.trd.2017.12.018.
20. Handy, S., et al. How the built environment a ects physical activity: Views from urban planning // Am. J. Prev. Med, 2002. Vol. 23. P. 64 – 73. DOI 10.1016/S0749-3797(02)00475-0/
21. Kelly, C., Tight, M., Hodgson, F., Page, M. A comparison of three methods for assessing the walkability of the pedestrian environment // J. Transp. Geogr, 2011. Vol. 19. P. 1500 – 1508. DOI 10.1016/j.jtrangeo.2010.08.001.
22. Kikuchi, H., et. al. Objectively Measured Neighborhood Walkability and Change in Physical Activity in Older Japanese Adults: A Five-Year Cohort Study // Int. J. Environ. Res. Public Health, 2018. Vol. 15. P. 1814. DOI 10.3390/ijerph15091814.
23. Krambeck, H. The Global Walkability Index // Environmental Engineering Journal, 2006.
24. Leslie, E., et.al. Walkability of local communities: Using geographic information systems to objectively assess relevant environmental attributes // Health & Place, 2007. Vol.13. P. 111 – 122. DOI 10.1016/j.healthplace.2005.11.001.
25. Litman, T. Economic Value of Walkability // Transp. Res. Rec, 2003. Vol. 1828. P. 3 – 11. DOI 10.3141/1828-01.
26. Lo, R. Walkability: What is it? J. Urban // Int. Res. Placemaking Urban Sustain, 2009. Vol. 2. P. 145 – 166. DOI 10.1080/17549170903092867.
27. Maghelal, P., Capp, C. Walkability: A review of Existing Pedestrian Indices // J. Urban Reg. Inf. Syst. Assoc., 2011. Vol. 23. P. 5 – 19. DOI 10.1017/CBO9781107415324.004.
28. Mayne, D., Morgan, G., Jalaludin, B., Bauman, A. Area-level walkability and the geographic distribution of high body mass in Sydney, Australia: A spatial analysis using the 45

and up study // Int. J. Environ. Res. Public Health, 2019. Vol. 16. 664p. DOI 10.3390/ijerph16040664.

29. Miura, H. et al. A Study on Navigation System for Pedestrians Based on Street Illuminations // Knowledge-Based and Intelligent Information and Engineering Systems, 2011. P. 49-55. DOI 10.1007/978-3-642-23854-3_6.

30. Mostafa Refat, I. A Parametric Study of the Effect of Building Distributions and Size on the Propagation of Sound in the Urban Environment // Journal of Architectural Engineering Technology 3.1, 2014. P. 1-8. ISSN 2168-9717.

31. Mukhina, K., Rakitin, S., Visheratin, A. Detection of tourists attraction points using Instagram profiles // Procedia Computer Science, 2017. P. 2378-2382. ISSN 1877-0509.

32. Oja, P., Vuori, I., Paronen, O. Daily walking and cycling to work: Their utility as health-enhancing physical activity // Patient Educ. Couns, 1998. Vol. 33. P. 87 – 94. DOI 10.1016/S0738-3991(98)00013-5.

33. Pikora, T., et al. Developing a reliable audit instrument to measure the physical environment for physical activity // Am. J. Prev. Med, 2002. Vol. 23. P. 187 – 194. DOI 10.1016/S0277-9536(02)00163-6.

34. Pikora, T.; Giles-Corti, B.; Bull, F.; Jamrozik, K.; Donovan, R. Developing a framework for assessment of the environmental determinants of walking and cycling // Soc. Sci. Med, 2003. Vol. 56. P. 1693 – 1703. DOI 10.1016/S0277-9536(02)00163-6.

35. Pikora, T., et. al. Neighborhood environmental factors correlated with walking near home: Using SPACES // Med. Sci. Sports Exerc, 2006. Vol. 38. P. 708 – 714. DOI 10.1249/01.mss.0000210189.64458.f3.

36. Quercia, D., Rossano S., Aiello L. The Shortest Path to Happiness: Recommending Beautiful, Quiet, and Happy Routes in the City // Proceeding HT '14 Proceedings of the 25th ACM conference on Hypertext and social media, 2014. P. 116-125. arXiv:1407.1031.

37. Ribeiro, A.I., Homann, E. Development of a Neighbourhood Walkability Index for Porto Metropolitan Area. How Strongly Is Walkability Associated with Walking for Transport? // Int. J. Environ. Res. Public Health, 2018. Vol. 15. 2767p. DOI 10.3390/ijerph15122767.

38. Saelens, B.E., Sallis, J.F., Frank, D.L. Environmental correlates of walking and cycling: Findings from the transportation, urban design, and planning literatures // Soc. Behav. Med, 2003. Vol. 25. P. 80 – 91. DOI 10.1207/S15324796ABM2502_03.

39. Sallis, J.F., Owen, N., Fisher, E.B. Ecological models of health behavior. In Health Behavior and Health Education: Theory, Research, and Practice, Jossey-Bass A Wiley Imprint: San Francisco, CA, USA, 2008, pp. 465 – 485. DOI 10.7326/0003-4819-116-4-350_1.

40. Tsiompras, A., Photis, Y., What matters when it comes to “walk and the city”? Defining a weighted GIS-based walkability index // Transp. Res. Procedia, 2017. Vol. 24. P. 523 – 530. DOI 10.1016/j.trpro.2017.06.001.

41. Vale, D., Saraiva, M., Pereira, M. Active accessibility: A review of operational measures of walking and cycling accessibility // J. Transp. Land Use, 2015. P. 209 – 235. DOI 10.5198/jtlu.2015.593.

42. Yasufumi, T. et al. Walking Route Recommender for Supporting a Walk as Health Promotion // IEICE Transactions on Information and Systems, 2017. P. 671-681. DOI 10.1587/transinf.2016DAP0006

43. Yen, Jin Y. Finding the K Shortest Loopless Paths in a Network // Management Science, 1971. P. 712-716.

44. Zhong, W., Chen, F. et al. SAFEBIKE: A Bike-sharing Route Recommender with Availability Prediction and Safe Routing, 2017. arXiv:1712.01469.

45. Zhu, X., Lee, C. Walkability and Safety Around Elementary Schools. Economic and Ethnic Disparities // Am. J.Prev. Med, 2008. Vol. 34. P. 282 – 290. DOI 10.1016/j.amepre.2008.01.024.

46. Блинов, И.Н. Java. Промышленное программирование: практ. пособие / И.Н. Блинов, В.С. Романчик. Минск: УниверсалПресс, 2007, 704 с. ISBN 978-985-6699-63-7.

47. Кормен, Т., Лейзерсон, Ч., Ривест, Р., Штайн, К. Алгоритмы: построение и анализ. Под ред. И. В. Красикова. М.: Вильямс, 2005, 1296 с.

48. Лутц М. Изучаем Python, 4-е изд. / пер. с англ. СПб: Символ-Плюс, 2011, 1280 с.

49. Одиночкина С.В. Основы технологий XML. СПб: НИУ ИТМО, 2013, 56 с.

50. Шилдт, Герберт. Java 8. Полное руководство / пер. с англ. 9-е изд. М. : ООО "И.Д. Вильяме", 2015. 1 376 с. : цв. ил. ISBN 978-5-8459-1918-2.

Ресурсы сети Интернет:

51. http://desktop.arcgis.com/ru/arcmap/latest/tools/spatial-analyst-toolbox/euclidean-distance.htm – Евклидово расстояние в ArcGIS. (дата обращения: 01.04.2020).

52. http://developer.android.com/guide/topics/manifest/manifest-intro – App Manifest Overview. (дата обращения: 01.08.2021).

53. http://developer.android.com/studio – Android Studio. (дата обращения: 01.08.2021).

54. http://developers.google.com/maps/documentation/android-sdk/get-api-key?hl=ru – Использование ключей API Maps SDK for Android. (дата обращения: 01.08.2021).

55. http://educba.com/jdbc-architecture/ – Introduction to JDBC Architecture. (дата обращения: 01.12.2021).
56. http://jdbc.postgresql.org/documentation/head/index.html – The PostgreSQL JDBC Interface. (дата обращения: 01.02.2021).
57. http://jetbrains.com/ru-ru/idea/ – обзор IntelliJ IDEA. (дата обращения: 01.02.2021).
58. http://krasnakarta.ru/spot/map – интерактивная карта Красногвардейского района Санкт-Петербурга. (дата обращения: 01.02.2020).
59. http://maps.me/ – мобильное приложение Maps.me (дата обращения: 16.04.2020).
60. http://medium.com/@urbica/walkstreets-5a41b22ae104 – описание приложения Walkstreets. (дата обращения: 06.11.2019).
61. http://pgrouting.org/ – расширение pgRouting. (дата обращения: 01.11.2019).
62. http://postgis.net/ – геопространственная БД Postgis. (дата обращения: 07.11.2019).
63. http://postgresql.org/about/ – обзор PostgreSQL. (дата обращения: 01.02.2021).
64. http://ru.wikipedia.org/wiki/. (дата обращения: 11.02.2021).
65. http://tech.yandex.ru/direct/doc/dg/concepts/about-docpage/ – документация API Яндекс.Директа. (дата обращения: 09.11.2019).
66. http://tech.yandex.ru/maps/doc/geosearch/concepts/about-docpage/ – сервис поиска по организациям компании «Яндекс». (дата обращения: 10.11.2019).
67. http://travelpath.ru/ – сервис построения туристических маршрутов TravelPath. (дата обращения: 04.11.2019).
68. http://vk.com/dev/first_guide – обзор API ВКонтакте. (дата обращения: 01.02.2021).
69. http://vk.com/dev/photos.search – обзор метода photos.search. (дата обращения: 01.02.2021).
70. http://yandex.ru/maps – поисково-информационная картографическая служба «Яндекс.Карты». (дата обращения: 10.11.2019).
71. Semenov A. Development of service suggesting walking routes // MSc Dissertation. W., Wuhan University, 2018. 45p. URL: https://www.researchgate.net/publication/325567805_Development_of_service_suggesting_walking_routes (дата обращения: 02.11.2019).