

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего
образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление 09.03.01 «Информатика и вычислительная техника»
подготовки:

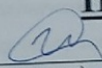
Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

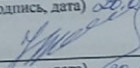
на тему

**СЕРВИС АДРЕСНОГО ГЕОГРАФИЧЕСКОГО
ПОИСКА**

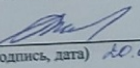
Студент


(подпись, дата) 20.06.2024 Новиков Иван Михайлович

Руководитель

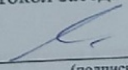

(подпись, дата) 20.06.2024 Слепцов Николай Владимирович

Нормоконтролёр


(подпись, дата) 20.06.2024 Бычков Андрей Станиславович

Работа допущена к защите (протокол заседания. кафедры от 27.05.2024г. № 11)

Заведующий кафедрой


(подпись) М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от

24.06.2024 № 39)

Секретарь ГЭК


(подпись) Гурин Е.И.
(фамилия, инициалы)

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
 М.А. Митрохин
« 14 » 12 20 24 г.

ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Новиков Иван Михайлович гр.20ВВП1 факультета вычислительной техники
2. Тема работы: Сервис адресного географического поиска
Тема утверждена приказом ПГУ № 405/о от 19 апреля 2024 г.
3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ ПГУ Слепцов Н.В.
4. Задание на работу (назначение разработки, исходные данные и т.п.)

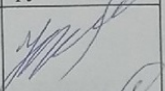
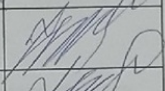
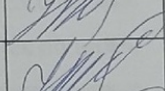
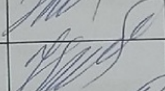
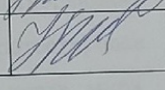
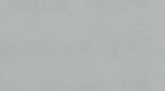
Разработать сервис адресного географического поиска, позволяющий выполнять поиск географического объекта (получить координаты и базовую информацию) на основе данных из базы OSM. Также, необходимо составить карту в растровом формате, для последующего использования в сервисе.

Сервис должен позволять выполнять адресный поиск, поиск по OSM тегам, отдельный адресный поиск, а также предоставлять необходимые картографические данные для визуализации найденного объекта.

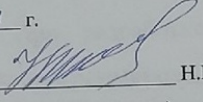
5. Перечень подлежащих разработке вопросов

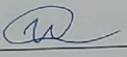
Требуется произвести анализ предметной области: изучить существующие аналоги сервисов, изучить способы проектирования и разработки приложений и виды архитектурных паттернов. Также необходимо изучить основные языки программирования для разработки приложений.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области методов обнаружения объектов на изображении	32	19.04.2024- 22.04.2024	
2	Анализ предметной области методов измерения расстояния	32	23.04.2024- 27.04.2024	
3	Анализ инструментов разработки	65	28.04.2024- 02.05.2024	
4	Разработка программного обеспечения на основе проведенного анализа	94	03.05.2024- 23.05.2024	
5	Отладка и тестирование	49	24.05.2024- 27.05.2024	
6	Оформление выпускной работы	32	28.05.2024- 30.05.2024	

Дата выдачи задания "19" апреля 2024 г.

Руководитель бакалаврской работы  Н.В. Слепцов
(подпись, дата)

Задание к исполнению принял студент  И.М. Новиков
(подпись, дата)

Работу к защите допустить
Декан факультета  24.06.2024 Л.Р. Фионова
(подпись, дата)

Реферат

Работа содержит 95 л., 31 рисунков, 1 таблицы, 7 источников,
2 приложения

C#, VISUAL STUDIO, JAVASCRIPT, POSTGRESQL, QGIS, КЛИЕНТ -
СЕРВЕРНОЕ ПРИЛОЖЕНИЕ

Цель работы – разработать сервис адресного географического поиска.

Сервис должен позволять выполнять адресный поиск, поиск по тегам, раздельный адресный поиск по базе данных OpenStreetMap, также в сервис должна входить карта мира, экспортированная в растровый формат.

Результатом проделанной работы является проект, удовлетворяющий поставленной задаче, а именно, спроектирован, разработан и протестирован сервер, выполняющий адресный поиск, была составлена карта мира, необходимая для визуализации. При разработке использовался ряд технологий, включая в себя языки программирования C#, JavaScript, C# библиотека Npgsql, библиотека OpenLayers, СУБД PostgreSQL, язык гипертекстовой разметки документов HTML, язык описания внешнего вида документа CSS, геоинформационная система QGIS, а также несколько утилит, необходимых для составления кастомизированных баз данных на основе открытой картографической базы OpenStreetMap

ПГУ 1.090301.18.001 ПЗ				
Изм	Лист	№ докум.	Подпись	Дата
Разраб.		Новиков И.М.		20.06.24
Пров.		Слепцов Н.В.		20.06.24
Н. Контр.		Бычков А.С.		20.06.24
Утв.		Митрохин М.А.		20.06.24

Литера	Лист	Листов
	2	

Сервис адресного географического поиска

Пояснительная записка

ФВТ, каф. ВТ
гр. 20ВВП1

Перечень сокращений и обозначений

В данной работе применяются следующие сокращения и обозначения:

БД – база данных;

ГИС – геоинформационная система;

ОС – операционная система;

ПК – персональный компьютер;

ПО – программное обеспечение;

СУБД – система управления базой данных;

API (Application programming interface) – программный интерфейс приложения;

HTTP (Hypertext Transfer Protocol) – протокол передачи гипертекста;

JSON (Java Script Object Notation) – текстовый формат для обмена структурированными данными и для хранения их;

MVC (англ. Model-View-Controller) – паттерн проектирования архитектуры;

MVVM (англ. Model-View-ViewModel) – паттерн проектирования архитектуры;

OSM (OpenStreetMap) - некоммерческий картографический проект по созданию свободной и бесплатной географической карты мира;

SQL (Structured Query Language) - язык структурированных запросов.

					ПГУ 1.090301.18.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Содержание

Перечень сокращений и обозначений	5
1 Анализ предметной области	9
1.1 Анализ существующих аналогов.....	11
1.1.1 Сервис «Google maps»	11
1.1.2 Сервис «Яндекс карты»	14
1.1.3 Сервис «2GIS»	17
1.1.4 Сервис «Nominatim»	20
1.2 Постановка задачи.....	22
1.3 Функционал приложения	23
2 Выбор технологий и средств разработки приложения	25
2.1 Выбор платформы исполнения.....	25
2.2 Выбор языка программирования	25
2.2.1 Java.....	25
2.2.2 C++:.....	25
2.2.3 Python:	25
2.2.4 C#	26
2.3 Выбор СУБД.....	26
2.3.1 MySQL:.....	26
2.3.2 SQLite:	27
2.3.3 PostgreSQL:	27
2.4 Выбор среды разработки	28
2.4.1 Visual Studio Code:	28
2.4.2 JetBrains Rider:	28
2.4.3 Sublime Text:	29
2.4.4 Visual studio:	29
2.5 Выбор геоинформационной системы.....	30
2.5.1 ArcGIS	30
2.5.2 QGIS.....	31
2.6 Выбор дополнительных инструментов разработки.....	31

2.6.1 osmConvert	31
2.6.2 osm2pgsql:	31
2.7 Выбор паттерна проектирования архитектуры.....	32
2.7.1 Модель "Клиент-Сервер" (Client-Server).....	32
2.7.2 Модель "Микросервисы" (Microservices)	32
2.7.3 Модель “Монолитная архитектура” (Monolithic Architecture)	33
2.7.4 Модель “Модель-Вид-Контроллер” (MVC)	33
3 Реализация приложения	36
3.1 Описание работы приложения.....	36
3.2 Структура базы данных	37
3.3 Описание компонентов.....	39
4 Реализация электронной карты мира	45
5 Руководство пользователя.....	48
5.1 Результат работы приложения.....	48
5.1.1 Просмотр главной страницы сервера.....	48
5.1.2 Адресный поиск.	49
5.1.3 Адресный поиск	50
5.1.4 Поиск по тегам	51
6 Безопасность жизнедеятельности.....	52
Заключение	55
Список использованных источников	56
Приложение А	57
Приложение А.1–Диаграмма последовательностей.....	58
Приложение А.2–Диаграмма развертывания	59
Приложение А.3–Диаграмма компонентов.....	60
Приложение А.4–Диаграмма прецедентов.....	61
Приложение Б – Листинг программы	62
Приложение В – Презентация.....	87

Введение

Современный мир становится все больше зависимым от технологий и географические информационные системы (ГИС) не являются исключением. Сервисы географического адресного поиска становятся все более популярными, поскольку они дают удобную возможность выполнять поиск точной и актуальной информации о географических объектах. Также, подобные сервисы обеспечивают доступ к актуальным и всеобъемлющим картографическим данным, что является большим преимуществом для пользователей, использующих подобные сервисы в различных областях, включая логистику, урбанистику и навигацию.

Существует определенное количество сервисов, предоставляющих доступ пользователю к геолокационным данным и картам. Однако, не все из них отличаются открытостью и доступностью для внедрения их в сторонние проекты. В этом контексте разработка сервиса адресного поиска с открытым исходным кодом и свободному доступу к картографическим данным становится не только необходимым действием, но и значимым вкладом в сферу геолокационных технологий.

Сервис адресного поиска, разработанный в рамках данного дипломного проекта, не только облегчит процесс поиска местоположения и информации об объекте, но также обеспечит пользователя картой мира, что повысит удобство и эффективность взаимодействия с географическими данными.

Таким образом, создание сервиса адресного географического поиска является важной и актуальной задачей для современного общества т.к. данный проект может улучшить доступность, а также, в некоторых моментах, качество географической информации, упрощает ее использование в разных прикладных областях. Также, открытый исходный код сервиса и доступ к картографической информации в сервисе адресного поиска, разработанного в данном проекте, могут сыграть важную роль в сфере геолокационных технологий.

В связи с вышесказанным, реализация сервиса адресного географического поиска является актуальной темой для дипломного проекта.

1 Анализ предметной области

Перед разработкой сервиса адресного географического поиска следует провести анализ предметной области для выявления основных потребностей пользователей и особенностей функционирования сервиса.

Таким образом, необходимо изучить текущие ограничения и проблемы уже существующих систем адресного географического поиска, а также изучить требования пользователей к схожим сервисам.

В настоящий момент с системами адресного поиска имеется множество различных проблем, связанных с отсутствием открытости и гибкости систем поиска. Существует много геоинформационных систем, в функционал которых входит поиск по адресам. Однако практически все эти системы имеют сильно ограниченный программный интерфейс, предоставляют неудовлетворительный функционал или имеют закрытый исходный код, что не позволяет разработчикам в полном объеме встроить систему поиска в свое приложение. Кроме того, данные сервисы не позволяют пользователю редактировать географическую базу данных, что исключает возможность своевременного изменения БД, в след за изменением геополитической обстановки в мире. Плюсы и минусы подобных сервисов будут подробнее рассмотрены в следующем разделе.

Конъюнктура современных политических отношений диктует необходимость разработки, создания и поддержания независимых инструментов адресного поиска, основным требованием к которым должно являться наличие исчерпывающего пакета данных для обособленной работы сервиса. Остроту и актуальность проблеме добавляет отсутствие в свободном доступе электронной карты мира с современными границами Российской Федерации, включая недавно вошедшие в состав России новые регионы: Донецкая Народная Республика, Луганская Народная Республика, Запорожская и Херсонская области [1].

Развитие информационных технологий и плотное их внедрение в повседневную жизнь в виде: поисковых систем, маркетплейсов, туристических ресурсов и других подобных сервисов обуславливает потребность в

геоинформационных инструментах, позволяющих внедрять в указанные выше информационные ресурсы удобный и гибкий поиск картографических объектов с базовой информационной аннотацией.

Таким образом, разработка сервиса адресного географического поиска, являющегося темой данной выпускной квалификационной работы, становится обоснованно актуальным и перспективным направлением для дальнейшего развития поисковых инструментов различной направленности.

					ПГУ 1.090301.18.001 ПЗ	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

1.1 Анализ существующих аналогов

В современном информационном мире с развитием цифровых решений и технологий пользователи получают доступ к обширному количеству сервисов и приложений, которые упрощают выполнение определенных задач. Одной из областей, которая имеет место для развития в рамках современных технологий, является область адресного географического поиска.

Для анализа возможностей инструментов и приложений данной области мы сосредоточим внимание на основных сервисах и платформах, предлагающих решения в сфере адресного поиска и навигации.

Среди аналогов можно выделить:

- «Google maps»;
- «Яндекс карты»;
- «2GIS»;
- «Nominatim».

1.1.1 Сервис «Google maps»

На рисунке 1.1 отображена главная страница сервиса «Google maps».

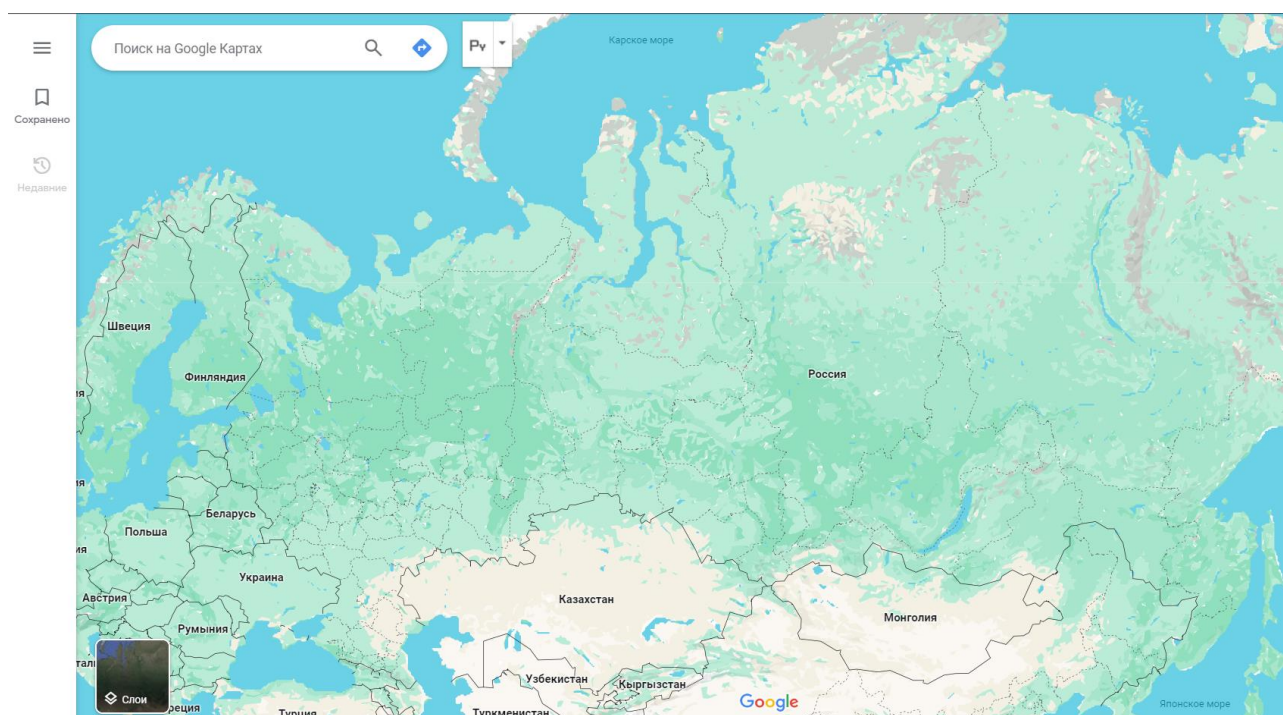


Рисунок 1.1 – Главная страница сервиса «Google maps»

На рисунке 1.2 представлен адресный поиск в сервисе «Google maps».

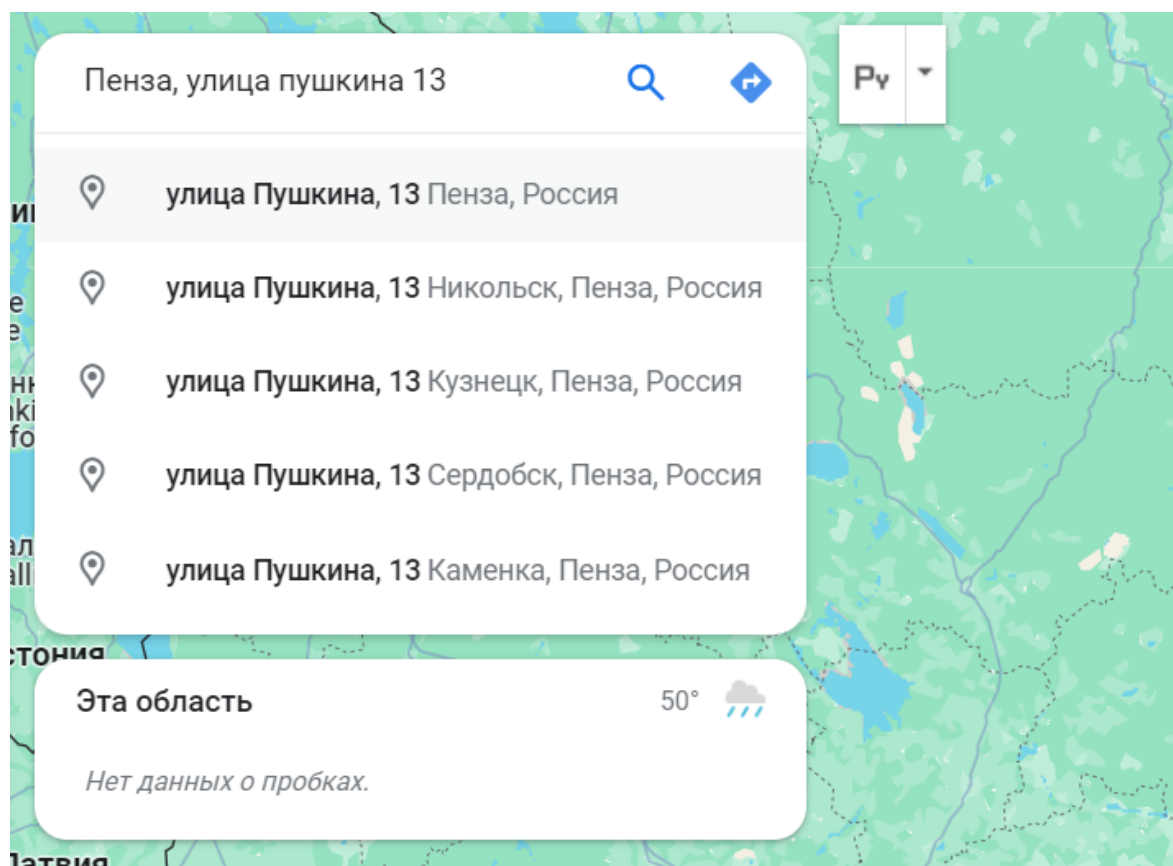


Рисунок 1.2 – Адресный поиск

Результат поиска представлен на рисунке 1.3

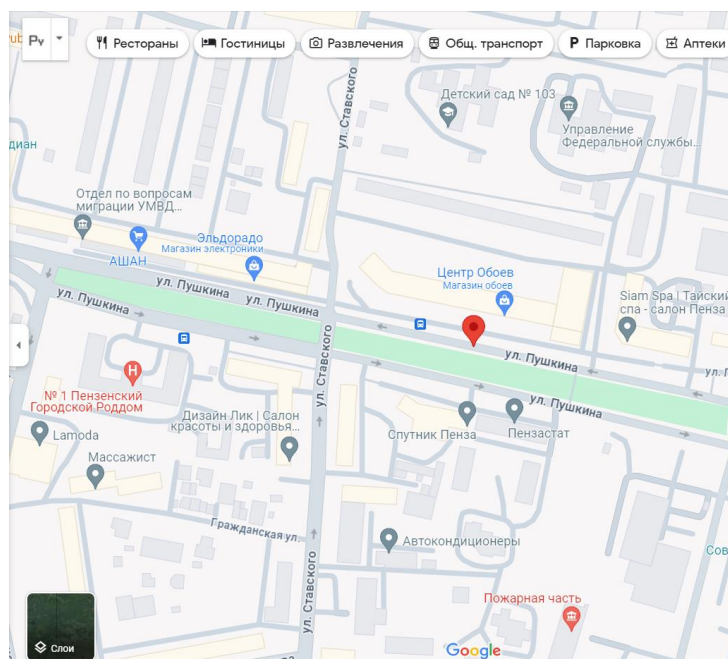


Рисунок 1.3 – Результат поиска сервиса «Google maps»

Информация об объекте поиска представлена на рисунке 1.4.

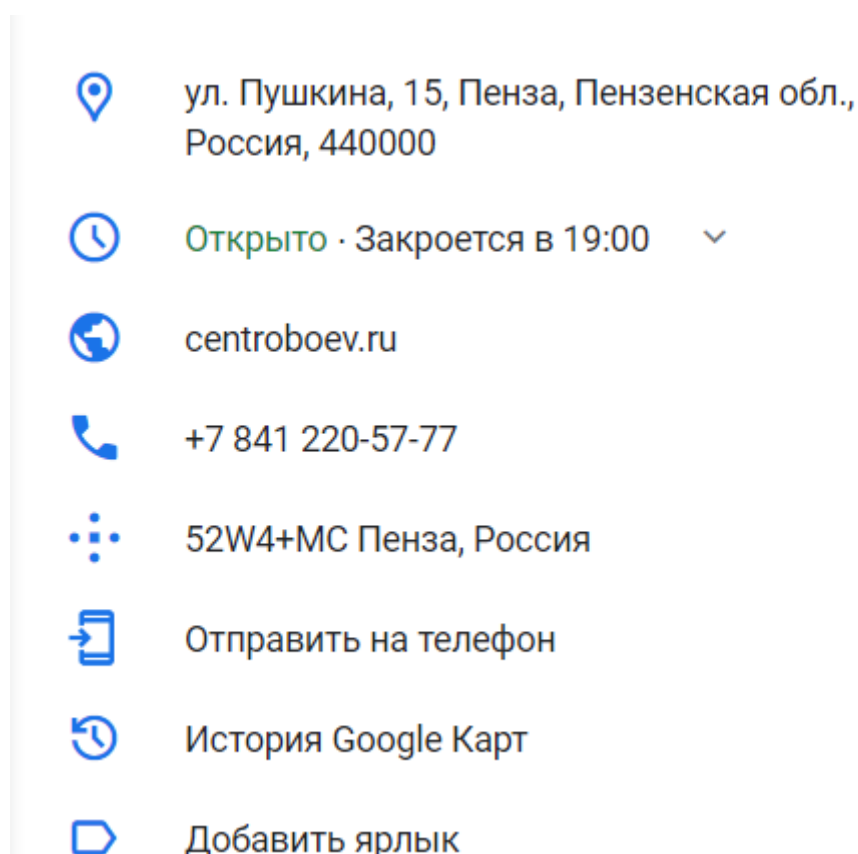


Рисунок 1.4 – Информация об объекте поиска в «Google maps»

Вид карты при использовании API от «Google maps» 1.5.

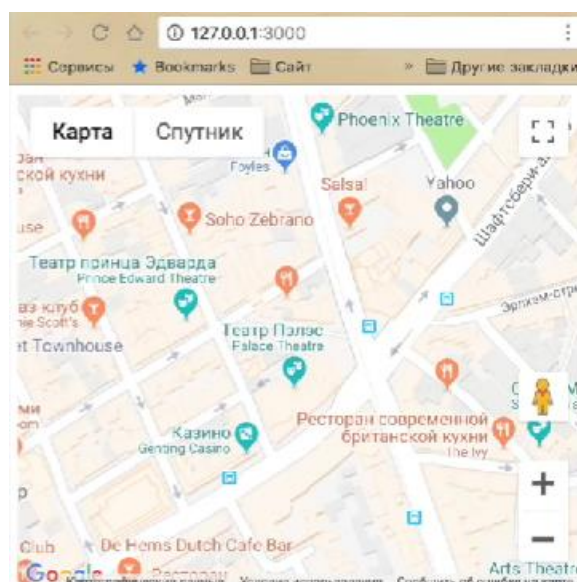


Рисунок 1.5 – Использовании API сервиса «Google maps»

Достоинства сервиса «Google maps»:

– высокая точность данных. Сервис предлагает одни из самых точных геоданных среди остальных инструментов адресного поиска, что обеспечивает надежность и точность результатов поиска объектов;

– широкий функционал. Помимо адресного поиска сервис Google maps предоставляет ряд дополнительных функций, что делает его весьма полезным инструментом для пользователей;

– интеграция с другими сервисами. Google Maps удобно интегрируется с другими сервисами Google, такими как Google Drive и Google Calendar, что обеспечивает удобный и единый интерфейс и для пользователя.

Недостатки сервиса «Google maps»:

– ограничения API. Google maps имеет ограничения на использование API (ограничение по кол-ву запросов, доступа к некоторым функциям), что может стать отталкивающим фактором при выборе разработчиком данный сервис для своего проекта;

– ограничение доступа. Некоторые регионы и страны могут иметь ограниченный доступ к некоторым функциям сервиса (навигация в реальном времени, информация о местоположении общественного транспорта).

1.1.2 Сервис «Яндекс карты»

Части интерфейса, API и основной функционал сервиса «Яндекс карты» представлены на рисунках 1.6, 1.7, 1.8, 1.9.

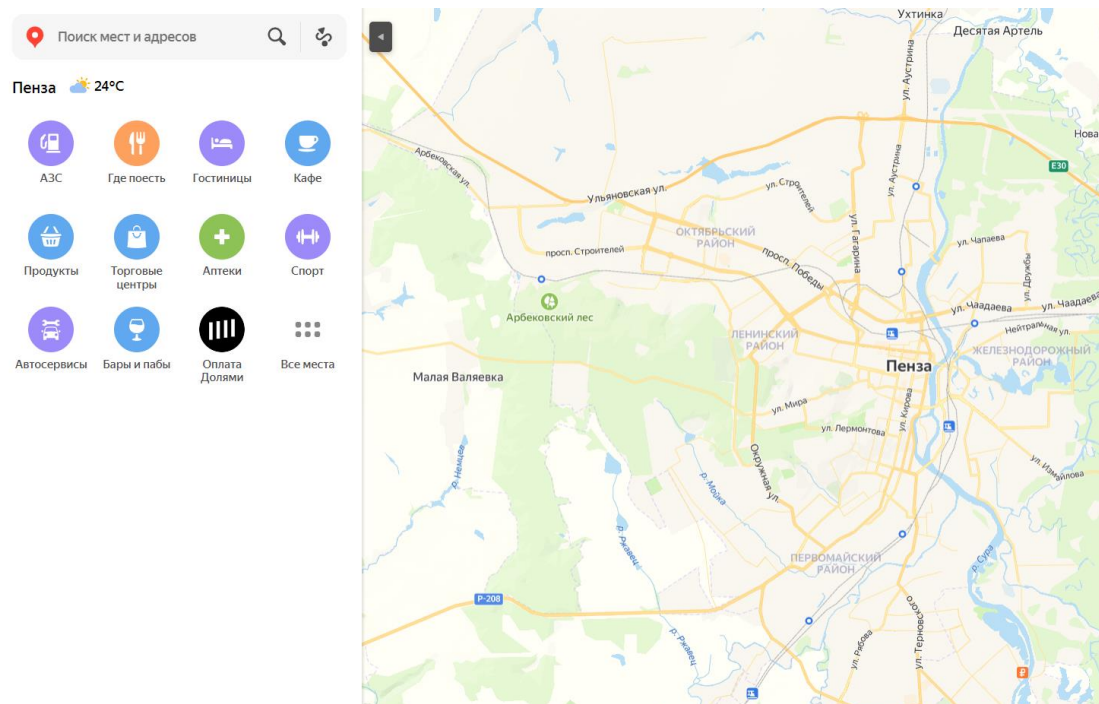


Рисунок 1.6 – Главная страница сервиса «Яндекс карты»

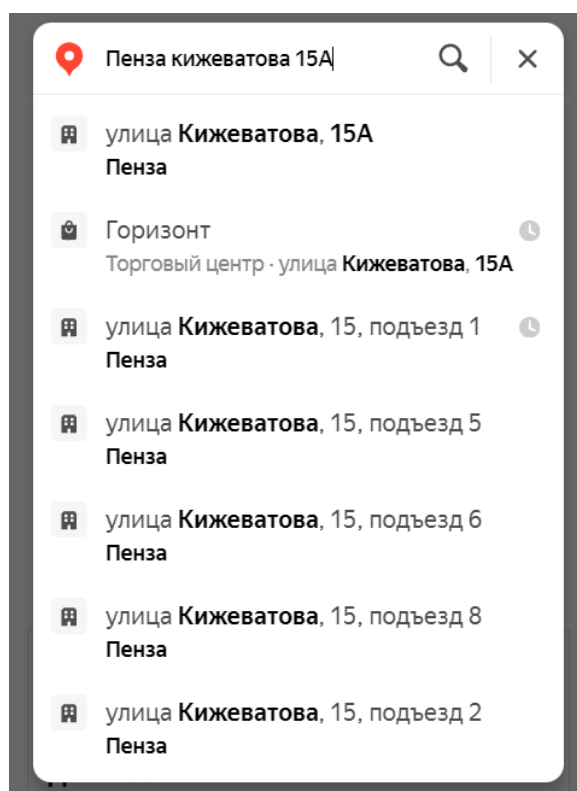


Рисунок 1.7 – Адресный поиск

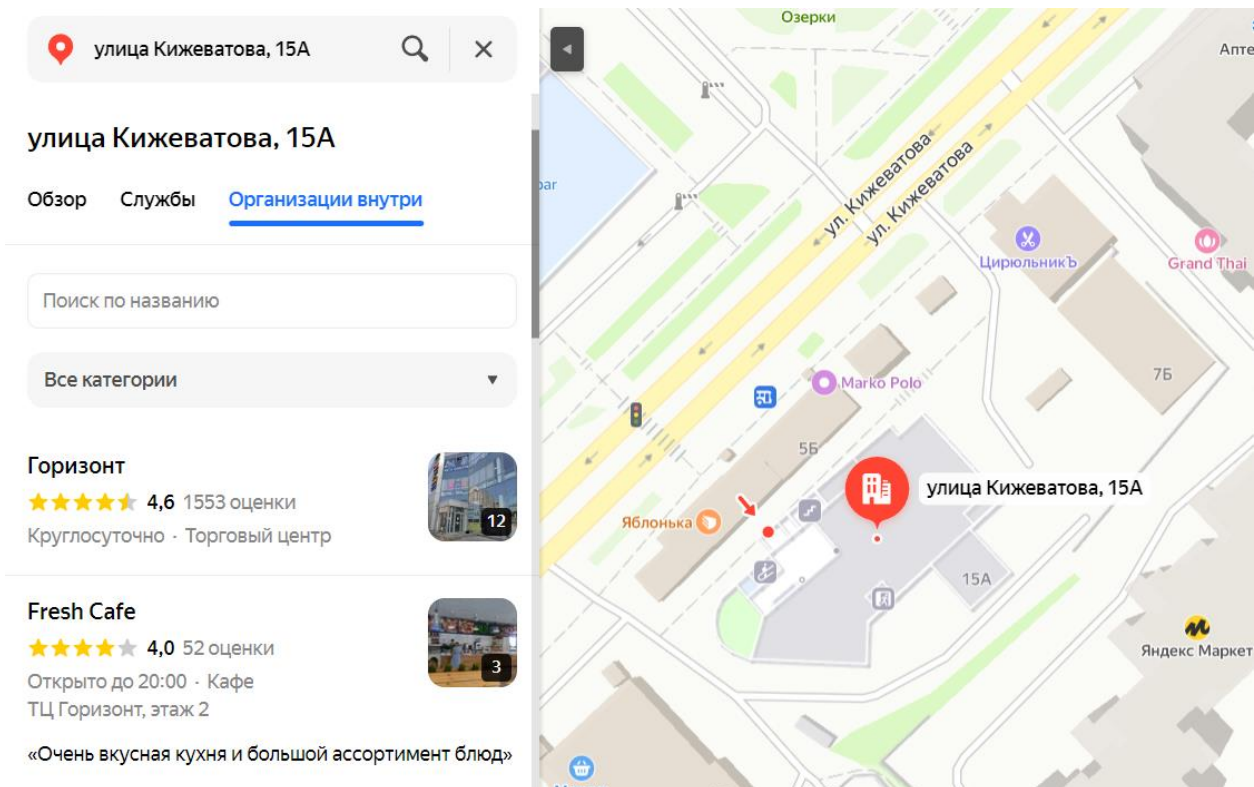


Рисунок 1.8 – Результат поиска и информация об объекте сервиса «Яндекс карты»

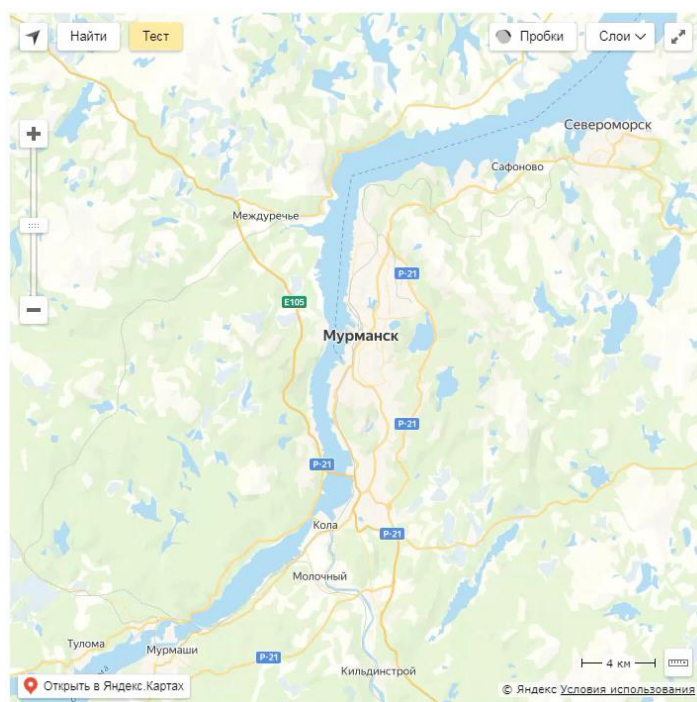


Рисунок 1.9 – Пример использования API сервиса «Яндекс карты»

Достоинства сервиса «Яндекс карты»:

- удобный и функциональный интерфейс;
- широкий функционал. Сервис предоставляет большой набор функций (адресный поиск, прокладку маршрутов для разного видов транспорта, информацию о загруженности дорог, информация об общественном транспорте и др.);
- интеграция с другими сервисами Яндекса. Сервис интегрирован с другими сервисами Яндекса, что обеспечивает пользователей удобным и единым интерфейсом;
- локализованные данные. Сервис предоставляет подробную информацию о местах и объектах Российской Федерации.

Недостатки сервиса «Яндекс карты»:

- ограничения API. У сервиса от Яндекса также есть ограничения на использование своего API, что может затруднить разработчикам интеграцию сервиса в свои проекты;
- недоступность данных за пределами России. Яндекс карты в основном ориентированы на рынок РФ и СНГ, данных за пределами названных регионов меньше и они могут быть менее точными.

1.1.3 Сервис «2GIS»

Части интерфейса, пример использования API и основной функционал сервиса «Яндекс карты» представлены на рисунках 1.10, 1.11, 1.12, 1.13.

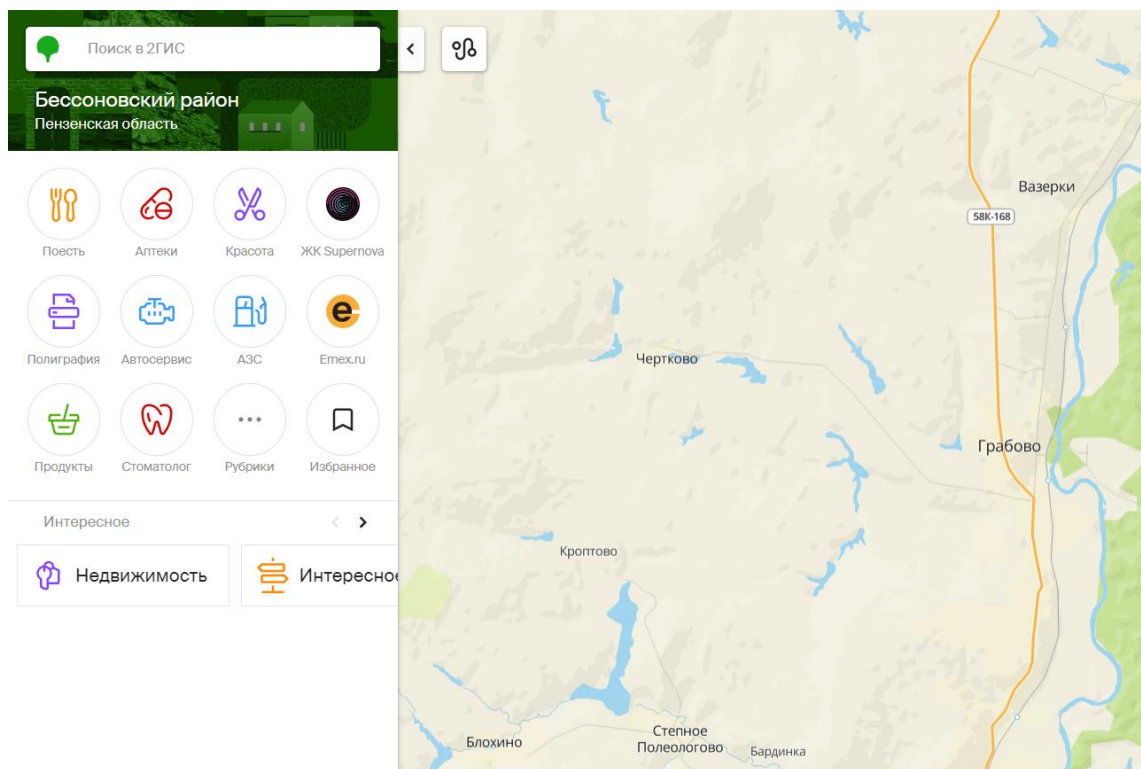


Рисунок 1.10 – Главная страница сервиса «2GIS»

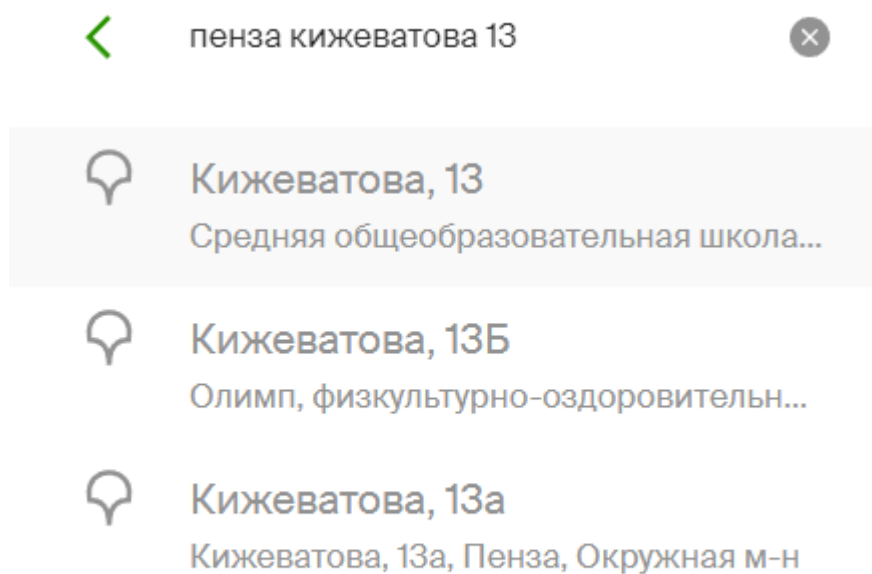


Рисунок 1.11 – Адресный поиск

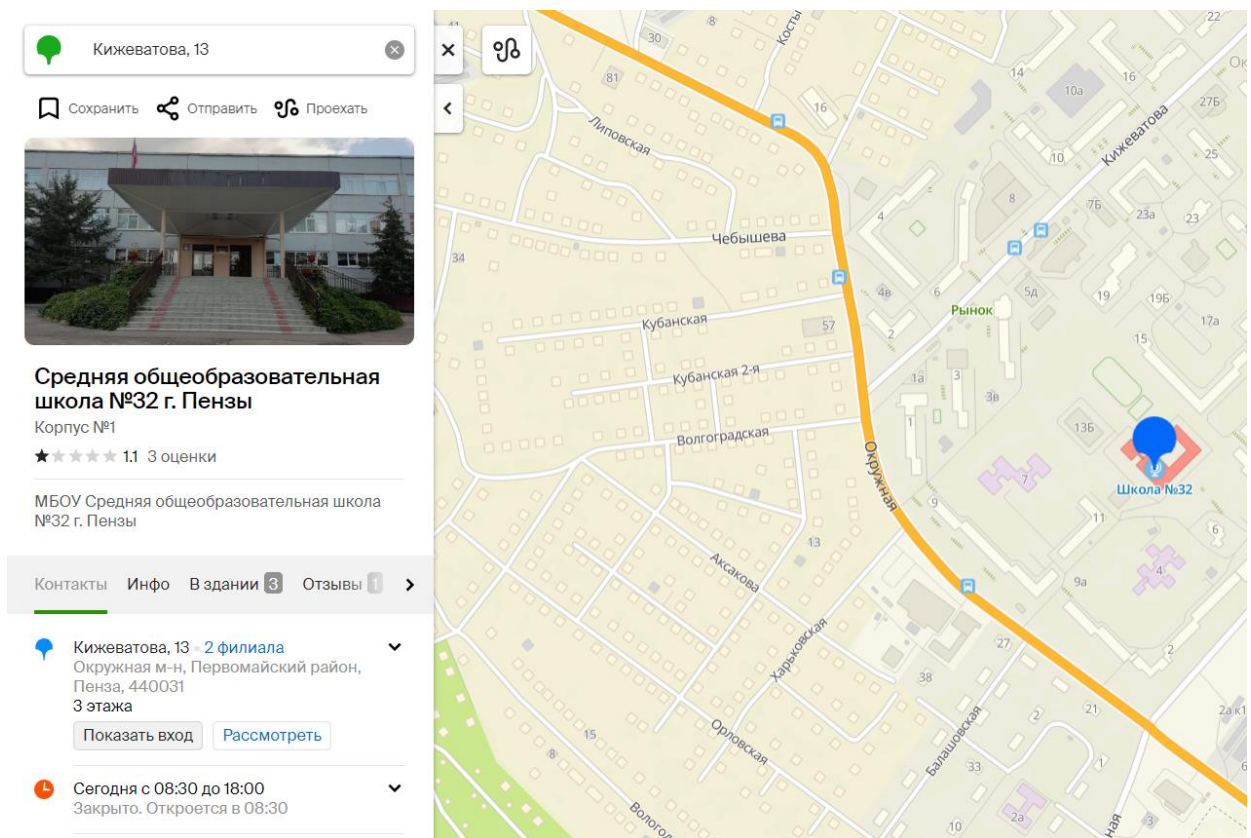


Рисунок 1.12 – Результат поиска сервиса «2gis»

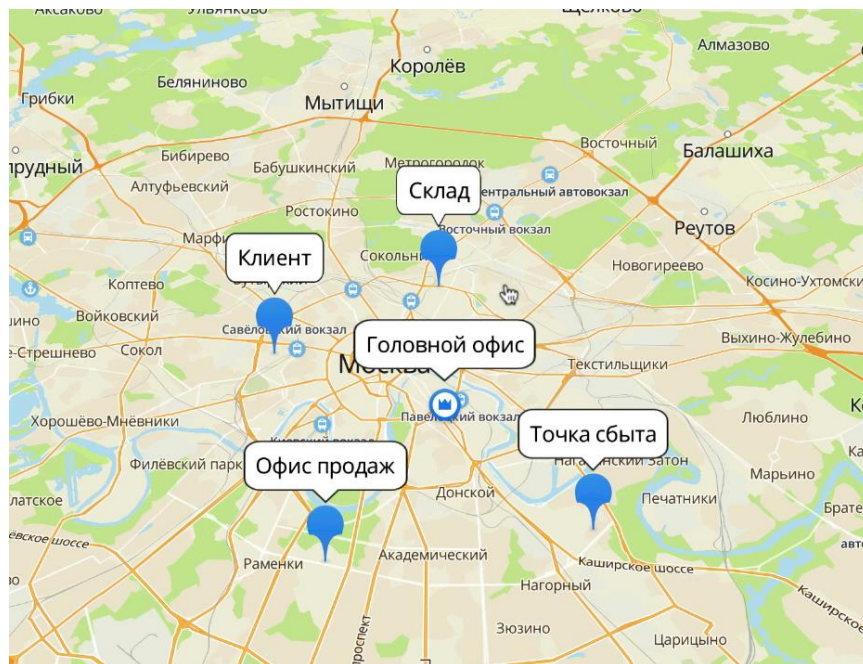


Рисунок 1.13 – Пример API сервиса «2GIS»

Достоинства сервиса «2GIS»:

- функциональный и удобный интерфейс пользователя, интуитивно понятная навигация;
- обширный функционал: Сервис 2GIS предоставляет большой набор инструментов и функций, позволяющий назвать данный сервис полноценной геоинформационной системой;
- интеграция с базами данных и другими сервисами, что улучшает качество поиска;
- локализованные региональные данные: Сервис предлагает точные данные о местоположении и базовой информации географических объектов РФ.

Недостатки сервиса «2gis»:

- платный API. программный интерфейс данного сервиса, в большей его части, является платным, что сильно ограничивает удобность использования API;
- ограничение географического покрытия. 2GIS имеет точные данные об объектах, находящихся в пределах России, однако остальные регионы имеют менее точную информацию.

1.1.4 Сервис «Nominatim»

Интерфейс, работа приложения и основной функционал инструмента «Nominatim» представлены на рисунках 1.14, 1.15.

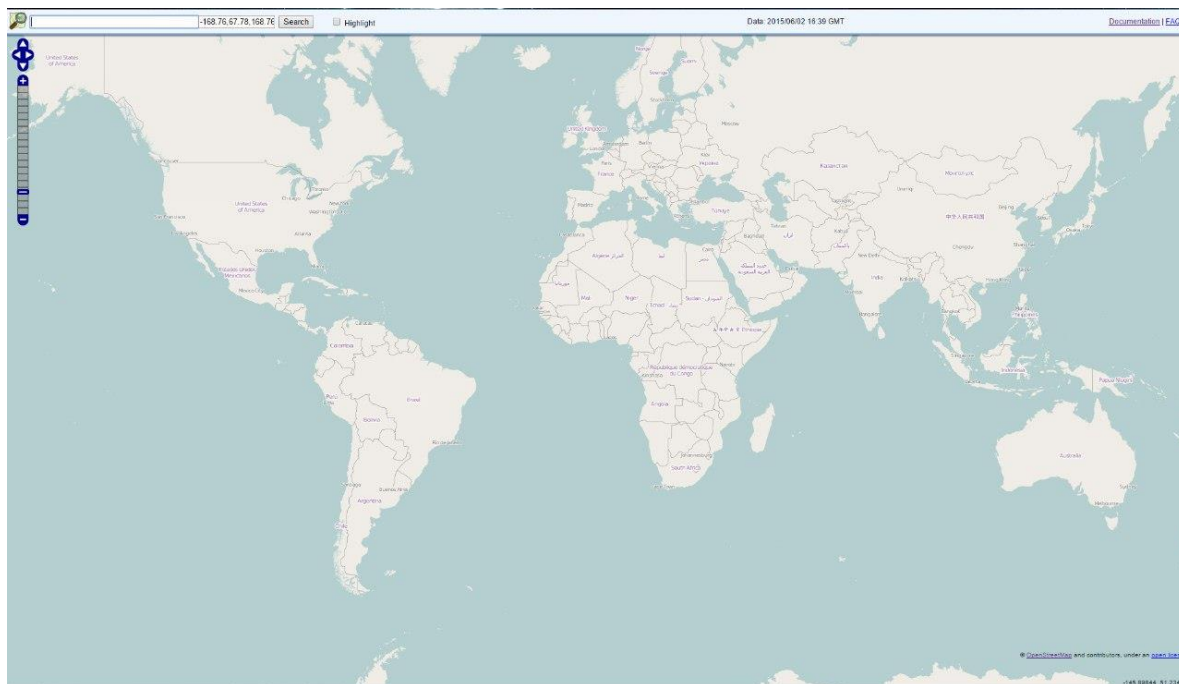


Рисунок 1.14 – Главная старница

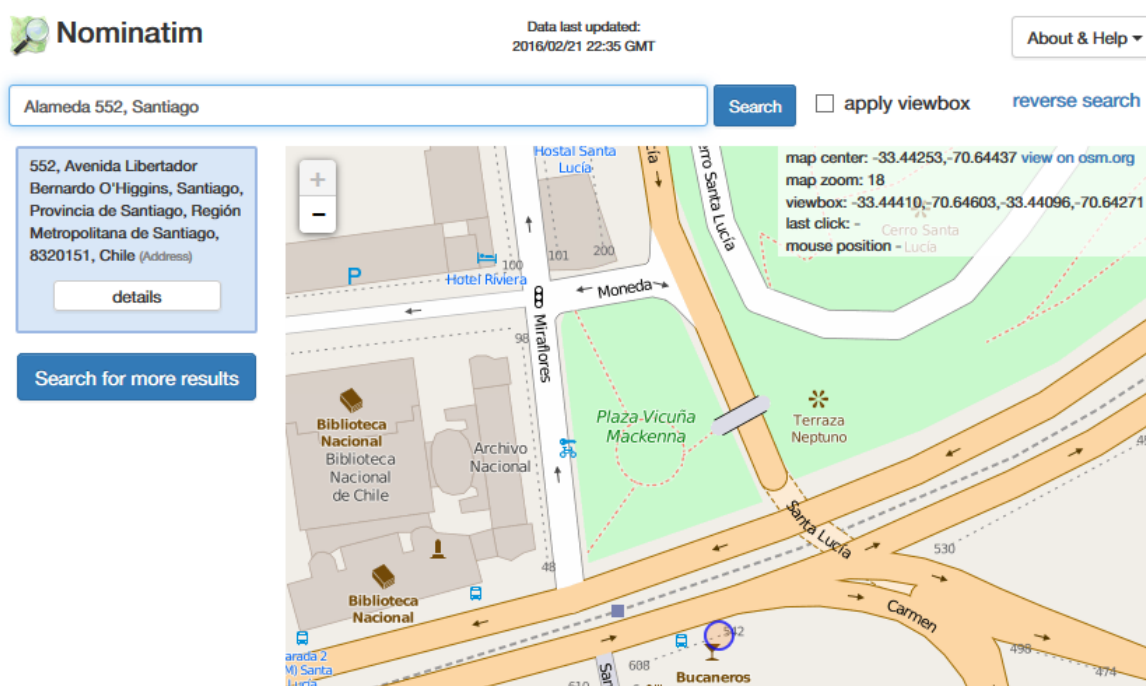


Рисунок 1.15 – Результат поиска сервиса «Nominatim»

Достоинства сервиса «Nominatim»:

- открытость и бесплатность: Nominatim является бесплатным и открытым инструментом, это позволяет использовать данный сервис без каких-либо ограничений;
- точность геокодирования: Сервис имеет высокую точность определения координат по адресу объекта;
- широкое покрытие: Nominatim использует базу OSM для выполнения поиска, данная база данных с равномерной точностью охватывает весь мир;
- наличие API и документации: Сервис предлагает пользователю или разработчику обширную документацию, а также программный интерфейс, что позволяет легко встроить адресный поиск в различные проекты.

Недостатки сервиса «Nominatim»:

- скорость работы: моментами неудовлетворительная скорость работы сервиса;
- Ограниченный функционал по сравнению с коммерческими аналогами: Несмотря на свою бесплатность, Nominatim может иметь ограниченный функционал по сравнению с коммерческими аналогами, что может быть недостатком для некоторых пользователей.

1.2 Постановка задачи

Необходимо разработать HTTP-сервера на языке программирования C#, который будет осуществлять адресный географический поиск с использованием данных открытого проекта OpenStreetMap (OSM), а также необходимо составить актуальную карту мира, с подробной региональной информацией РФ.

Основными задачами проекта являются:

- разработка HTTP-сервера. Реализация серверной части приложения, способной принимать запросы от клиентов, отправлять информацию и осуществлять поиск в базе данных по различным критериям;

– использование данных OpenStreetMap. Импорт данных из проекта OpenStreetMap и конвертация их в БД для последующего использования в поиске объектов по их географическим атрибутам;

– множественные виды поиска. Реализация нескольких видов поиска, включая адресный поиск, поиск по тегам OSM;

– разработка веб-интерфейса. Создание небольшой веб-части для демонстрации работы сервиса, обеспечивая пользователей возможностью использовать функционал сервиса через веб-браузер;

– составление карты мира в среде QGIS, для визуализации результатов поиска и навигации.

Минимальные системные требования:

– браузер Mozilla Firefox, Google Chrome, Opera (с поддержкой CSS3, HTML5);

– минимальная скорость интернета: 2-3 Мбит/сек;

– минимальный объем оперативной памяти: 4 ГБ (рекомендуется 8 ГБ и выше);

– Жесткий диск: SSD объемом не менее 256 Гб;

– Процессор: Intel Core i5 или AMD Ryzen 5 и выше;

– операционная системы Windows (не ниже 10 версии) или Windows server (не ниже Windows server 12).

1.3 Функционал приложения

Основные возможности сервиса:

– адресный поиск. Поиск необходимого объекта, путем разбивания цельной строки с адресом, на отдельные элементы и поиск их в БД;

– поиск по OSM тегам. Поиск географического объекта по парам ключ – значение, которые используются в базе данных проекта OSM;

– отдельный адресный поиск;

– визуализация результатов поиска.

Диаграмма прецедентов приведена на рисунке 1.16.



Рисунок 1.16 – Диаграмма прецедентов.

2 Выбор технологий и средств разработки приложения

В этом разделе проведен анализ сред разработки, языков программирования, СУБД и дополнительных инструментов. Исходя из анализа выбраны необходимые технологии, которые обеспечивают наибольшее удобство и гибкость разработки, а также дальнейшую поддержку приложения.

2.1 Выбор платформы исполнения

Выбор платформы, на которой будет работать сервис является важным аспектом разработки, так как он определяет то, какие технологии необходимо применять в ходе разработки.

Было принято решение использовать ПК или любую другую высокопроизводительную компьютерную систему (сервер), поддерживающую ОС Windows или Windows server, в качестве основной платформы исполнения.

2.2 Выбор языка программирования

При выборе технологий для разработки HTTP-сервера, следует учитывать такие факторы, как производительность, удобство разработки, доступность библиотек и фреймворков, а также опыт команды разработчиков.

2.2.1 Java

Преимущества: кроссплатформенность, обширные возможности для масштабирования, широкое сообщество и доступность библиотек.

Недостатки: высокое потребление ресурсов, многословность в синтаксисе.

2.2.2 C++:

Преимущества: высокая производительность, полный контроль над памятью и ресурсами, возможность оптимизации.

Недостатки: сложность в разработке и отладке, возможность ошибок на уровне памяти.

2.2.3 Python:

Преимущества: простой синтаксис, богатый выбор библиотек, высокая скорость разработки.

Недостатки: относительно низкая производительность, особенно в сравнении с компилируемыми языками.

2.2.4 C#

C# представляет собой отличный выбор для разработки HTTP-сервера в данном проекте по нескольким причинам:

- простота и удобство разработки. C# обладает чистым и понятным синтаксисом, что делает разработку приложений более удобной и эффективной;
- богатая экосистема и инструментарий. C# имеет широкий выбор библиотек и фреймворков для разработки серверных приложений, включая ASP.NET Core, который предоставляет мощные инструменты для работы с HTTP;
- производительность. C# обеспечивает высокую производительность благодаря своей оптимизированной виртуальной машине и возможности компиляции в машинный код;
- интеграция с существующими технологиями. C# легко интегрируется с другими технологиями и платформами, что позволяет использовать его в различных сценариях разработки.

В результате проведенного анализа, выбор пал на C# как наиболее подходящий язык программирования для разработки HTTP-сервера в данном проекте, благодаря его простоте, производительности и богатой экосистеме инструментов [2].

Также в качестве инструментов разработки фронт-стороны приложения были выбраны язык разметки HTML и язык стилей CSS и язык программирования JavaScript.

2.3 Выбор СУБД

2.3.1 MySQL:

Преимущества: широкое распространение, простота в установке и настройке, хорошая производительность для простых запросов.

Недостатки: Меньшая функциональность по сравнению с другими СУБД, ограничения в поддержке сложных запросов.

2.3.2 SQLite:

Преимущества: простота встроенной реализации, не требует отдельного сервера, хорошая производительность для небольших проектов.

Недостатки: не подходит для масштабируемых проектов с большими объемами данных, ограниченная функциональность.

2.3.3 PostgreSQL:

Преимущества: мощный и расширяемый, поддерживает сложные типы данных и функции, высокая производительность при работе с большими объемами данных. Также, обладает необходимым расширением, поддерживающим возможность работы с геоинформационными данными.

Недостатки: более сложная в установке и настройке по сравнению с MySQL, требует больше ресурсов.

PostgreSQL является наиболее подходящим выбором для хранения и обработки данных OpenStreetMap в данном проекте по нескольким причинам:

- мощность и расширяемость. PostgreSQL обладает богатым набором функций, включая поддержку географических данных и расширений для работы с ними, что делает его идеальным для хранения данных OSM;

- производительность. PostgreSQL обеспечивает высокую производительность при выполнении сложных запросов и обработке больших объемов данных, что важно для работы с географическими данными;

- надежность и безопасность. PostgreSQL известен своей надежностью и стабильностью, а также обеспечивает широкие возможности для обеспечения безопасности данных;

- активное сообщество и поддержка. PostgreSQL имеет активное сообщество разработчиков и пользователей, что обеспечивает доступ к обширной документации, инструментам и ресурсам для разработки и поддержки проекта.

Использование базы данных PostgreSQL совместно с расширением PostGIS обеспечивает надежное хранение и эффективный доступ к геоинформационным

данным [3]. PostgreSQL хорошо известна своей производительностью и масштабируемостью, а PostGIS добавляет функции для работы с географической информацией.

Таким образом, PostgreSQL является наилучшим выбором для базы данных в данном проекте, обеспечивая высокую производительность, расширяемость и надежность при работе с данными OpenStreetMap.

Объекты базы данных OpenStreetMap хранятся в формате PBF (Protocolbuffer Binary Format), который обеспечивает компактное и эффективное хранение геоинформационных данных. Для работы с данными OSM и перевода их в более удобный для использования формат применяются следующие инструменты:

2.4 Выбор среды разработки

2.4.1 Visual Studio Code:

Преимущества:

- Бесплатная и легкая в использовании среда разработки;
- Поддержка различных языков программирования и расширений;
- Удобная интеграция с системами контроля версий и различными облачными сервисами;
- Работает на различных операционных системах;

Недостатки:

- Меньший функционал по сравнению с Visual Studio;
- Может потреблять больше ресурсов компьютера при работе с большими проектами;

2.4.2 JetBrains Rider:

Преимущества:

- Поддержка языка C# и других популярных языков программирования;
- Удобная отладка и профилирование кода;

– Интеграция со средой разработки ReSharper для автоматической проверки и исправления кода;

– Поддержка плагинов и расширений.

Недостатки:

– Платная лицензия для полноценного использования;

– Малое сообщество, имеет низкую популярность в сравнении с остальными средами разработки.

2.4.3 Sublime Text:

Преимущества:

– Легкая и быстрая среда разработки;

– Поддержка различных языков программирования;

– Широкие возможности настройки интерфейса и функционала;

– Бесплатный пробный период для ознакомления.

Недостатки:

– Ограниченный функционал без использования плагинов;

– Не имеет интегрированных инструментов для разработки на C#.

2.4.4 Visual studio:

Visual Studio - это одна из самых мощных и популярных интегрированных сред разработки (IDE) от Microsoft. Ее выбор как основного инструмента разработки обусловлен рядом причин:

– мощный функционал. Visual Studio предоставляет широкие возможности для разработки различных типов проектов, включая приложения для Windows, веб-сайты, мобильные приложения и многое другое. Ее большой экосистеме позволяет работать с различными языками программирования, такими как C#, C++, Python, JavaScript, и другими;

– интуитивный интерфейс. Интерфейс Visual Studio дружелюбен к разработчикам, обладает удобным расположением инструментов, хорошей навигацией по проектам, интегрированной системой управления версиями и отладки;

– широкие возможности отладки. Visual Studio обладает продвинутыми инструментами отладки, которые упрощают процесс нахождения и исправления ошибок в коде;

– поддержка различных платформ. Visual Studio позволяет разрабатывать для различных платформ, таких как Windows, Android, iOS, MacOS, и многих других;

– богатая экосистема плагинов. Visual Studio имеет обширную библиотеку расширений и плагинов, которые позволяют дополнительно настраивать среду разработки под свои нужды.

Именно из-за комбинации всех этих преимуществ и удобства использования Visual Studio является одним из лучших выборов для разработчиков ПО.

2.5 Выбор геоинформационной системы

2.5.1 ArcGIS

Преимущества:

– широкий функционал и возможности для анализа пространственных данных;

– мощные инструменты для создания карт и визуализации;

– поддержка различных форматов данных и интеграция с базами данных.

Недостатки:

– высокая стоимость лицензий;

– требует специальных знаний для работы с программой.

2.5.2 Google Earth

Преимущества:

– простота использования и доступность для пользователей;

– большая база геоданных и возможность добавления собственных данных;

– интеграция с другими сервисами Google.

Недостатки:

- ограниченный функционал в сравнении с геоинформационными системами;
- ее всегда точное отображение данных.

2.5.3 QGIS

Почему было выбрано именно QGIS:

- бесплатное и открытое программное обеспечение;
- широкие возможности для работы с различными форматами данных;
- активное сообщество пользователей и наличие обучающих материалов;
- гибкость и возможность настраивать программу под свои потребности;
- возможность создания красивых и информативных карт.

Именно из-за своей бесплатности, удобства использования, функциональности и возможности визуализации данных было принято решение использовать QGIS для создания карты мира в рамках дипломного проекта.

2.6 Выбор дополнительных инструментов разработки

2.6.1 osmConvert

osmConvert — это утилита командной строки, позволяющая конвертировать данные OpenStreetMap из одного формата в другой. Например, с её помощью можно эффективно преобразовать данные из формата PBF в текстовый формат OSM, который более удобен для последующей обработки и анализа.

2.6.2 osm2pgsql:

osm2pgsql — это популярная утилита для импорта данных OSM в базу данных PostgreSQL с поддержкой географических данных PostGIS. С её помощью можно эффективно загружать и индексировать пространственные данные, что позволяет выполнять мощные запросы к базе данных для поиска адресов и других геоинформационных данных.

Таким образом, для разработки проекта были выбраны следующие технологии:

- язык программирования C#;
- СУБД PostgreSQL и расширение PostGis;
- среда разработки Visual Studio Code.

2.7 Выбор паттерна проектирования архитектуры

При разработке современного программного обеспечения одним из ключевых аспектов является выбор правильной архитектуры. Архитектурный выбор напрямую влияет на функциональность, производительность, масштабируемость и поддерживаемость системы. В рамках данной дипломной работы рассматривается разработка HTTP сервера на C# для адресного поиска по базе OpenStreetMap (OSM).

2.7.1 Модель "Клиент-Сервер" (Client-Server)

Модель "Клиент-Сервер" предполагает разделение на клиентскую и серверную части. Сервер хранит данные и логику обработки запросов, а клиент выполняет запросы к серверу и отображает результаты пользователю.

Используется в большинстве веб-приложений и мобильных приложений, где клиентское приложение взаимодействует с серверной частью через HTTP/HTTPS.

Преимущества:

- четкое разделение задач между клиентом и сервером;
- масштабируемость: можно развивать серверную часть независимо от клиентской.

Недостатки:

- уязвимость к сетевым проблемам;
- может потребовать сложную инфраструктуру на серверной стороне.

2.7.2 Модель "Микросервисы" (Microservices)

Микросервисная архитектура предполагает разделение системы на отдельные, автономные сервисы, каждый из которых отвечает за конкретную бизнес-логику и может развертываться независимо.

Подходит для больших и сложных систем, где важна гибкость и возможность масштабирования отдельных компонентов.

Преимущества:

- легкость масштабирования отдельных сервисов;
- независимое обновление и разворачивание сервисов;
- четкое разграничение ответственности между командами разработчиков.

Недостатки:

- усложнённое управление сетью и взаимодействием между сервисами;
- требует сложной системы мониторинга и логирования.

2.7.3 Модель “Монолитная архитектура” (Monolithic Architecture)

В рамках монолитной архитектуры приложение разрабатывается как единый целостный блок, где все компоненты тесно связаны друг с другом.

Подходит для небольших приложений или приложений с минимальными требованиями к масштабированию.

Преимущества:

- простота разработки и деплоя;
- не требует сложного управления взаимодействием между компонентами.

Недостатки:

- плохо масштабируется;
- сложность в поддержке и расширении при увеличении функциональности.

2.7.4 Модель “Модель-Вид-Контроллер” (MVC)

MVC (Model-View-Controller) архитектура разделяет приложение на три отдельных компонента: модель (хранение данных и бизнес-логика), вид

(представляет данные пользователю), контроллер (обрабатывает ввод и обновляет модель).

Часто используется для веб-приложений, где требуется отделить логику работы с данными от представления их пользователю.

Преимущества:

- четкое разделение ответственности между компонентами;
- облегчает тестирование и поддержку;
- возможность независимого обновления компонентов.

Недостатки:

- может усложнить разработку небольших приложений;
- требует понимания и соблюдения паттерна;

Обоснование выбора модели MVC

Для разработки HTTP сервера на C# для сервиса адресного поиска по базе OSM, модель MVC представляет собой оптимальный выбор по следующим причинам:

– четкое разделение обязанностей. MVC модель разделяет бизнес-логику, пользовательский интерфейс и контроль над данными на три отдельных компонента. Это облегчит вам разработку, тестирование и поддержку приложения, так как каждый компонент будет отвечать за свою конкретную задачу;

– масштабируемость и поддержка. MVC позволяет легко добавлять новые функции и изменять существующие без существенной реорганизации кода. Это важно для потенциального расширения вашего сервиса;

– соответствие веб-технологиям. MVC хорошо интегрируется с веб-технологиями и является стандартным подходом в разработке веб-приложений на C#, особенно с использованием ASP.NET MVC, который предоставляет мощный фреймворк для простого создания и поддержки веб-ресурсов;

– тестируемость. Компоненты MVC можно тестировать отдельно, что улучшает качество кода и облегчает процесс отладки;

– примеры успешного использования. Многие успешные веб-приложения разрабатываются с использованием модели MVC, что подтверждает её практическую ценность и эффективность.

Таким образом, модель MVC является оптимальным выбором для моего сервиса адресного поиска, обеспечивая баланс между сложностью разработки и возможностями масштабирования и поддержки.

					ПГУ 1.090301.18.001 ПЗ	Лист
						35
Изм.	Лист	№ докум.	Подпись	Дата		

3 Реализация приложения

В данном разделе описан процесс реализации приложения «Сервис адресного географического поиска». Здесь рассмотрены базовые компоненты, методы взаимодействия и структура БД, чтобы предоставить полную информацию о разработке приложения

3.1 Описание работы приложения

Топология сервиса и распределение ее компонентов по физическим узлам представлены на рисунке 3.1.

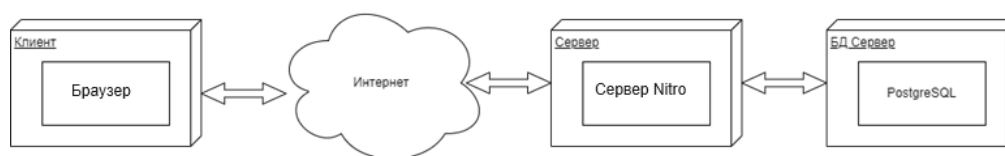


Рисунок 3.1 – UML-диаграмма развертывания системы

Алгоритм работы приложения заключается в том, что пользователь взаимодействует с клиентской частью либо направляет запросы напрямую к серверу. Сервер обрабатывает поступающие запросы, осуществляет поиск в базе данных и предоставляет ответы в виде HTTP-ответов или отображает результаты на фронтенд-части приложения

Последовательность выполняемых действий представлена на рисунке 3.2.

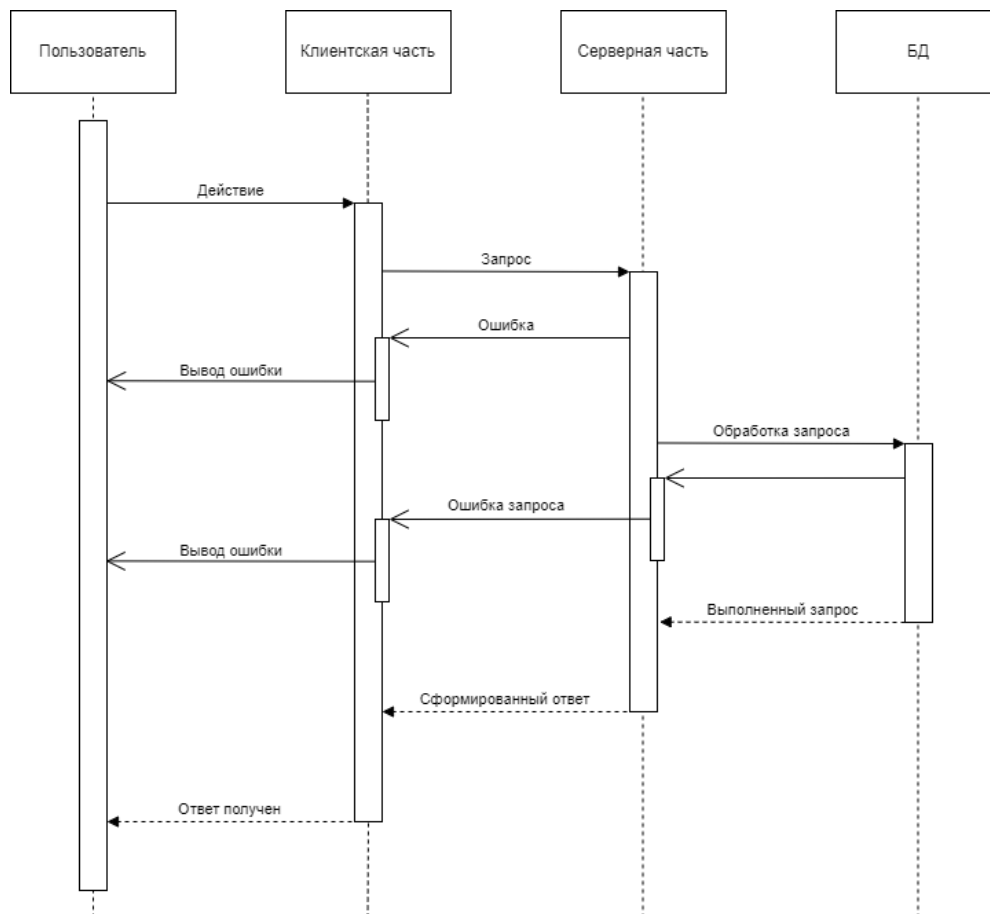


Рисунок 3.2 – UML-диаграмма последовательности исполнения приложения

3.2 Структура базы данных

Для реализации клиент-серверного приложения, способного хранить геоинформационные данные информацию, необходимо провести анализ и определить основные сущности и связи, которые будут присутствовать или отсутствовать в базе данных. Структура базы данных играет важную роль в обеспечении эффективной работы приложения.

Сервис должен предоставлять пользователю возможность быстрого и удобного поиска. База данных OSM была переведена, с помощью инструмента Osm2Pgsql, в базу данных PostgreSQL, стандартные настройки экспорта были изменены, чтобы получить необходимые для выполнения поиска элементы БД. Также, в базе данных остались дополнительные столбцы, оставшиеся после экспорта, их мы не будем рассматривать

Таблица planet_osm_point. В этой таблицы содержится информация об точечных географических объектах

Таблица 1 – Authors

Поле	Тип данных	Описание
osm_id	TEXT	Идентификатор объекта
addr:city	TEXT	Название города
addr:street	TEXT	Название улицы
addr:housenumber	TEXT	Номер дома/строения
name	TEXT	Общее название объекта
lat	TEXT	Долгота
lon	TEXT	Широта
...	...	...

Графическое представление базы данных показана на рисунке 3.3.

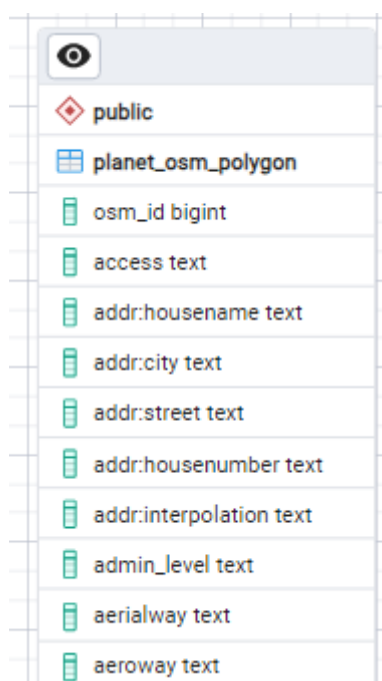


Рисунок 3.3 – Структура базы данных

На данном рисунке можно заметить более сложную структуру таблицы, однако как было указано выше, большинство столбцов необходимы инструменту osm2pgsql, он их создает автоматически.

3.3 Описание компонентов

В соответствии с архитектурой MVC, сервис имеет три основные части:

3.3.1 Model (модель):

Модель представляет собой компонент, отвечающий за обработку данных и бизнес-логику приложения. Модель содержит информацию о структуре данных, их хранении, обновлении и доступе к ним. Она представляет собой абстракцию данных и содержит методы для работы с ними, включая чтение, запись, обновление и удаление. Модель отвечает за валидацию данных, их проверку на корректность и целостность.

3.3.2 View (представление):

Представление отвечает за визуализацию данных для пользователя. Оно отображает информацию, предоставленную моделью, в удобном для восприятия пользователем виде. Представление может быть в виде веб-страницы, графического интерфейса пользователя или другого способа представления данных.

3.3.3 Controller (контроллер):

Контроллер является посредником между моделью и представлением. Он отвечает за обработку пользовательского ввода, вызов методов модели для получения данных и передачу этих данных на представление для отображения. Контроллер принимает запросы от пользователя, выполняет соответствующие действия и определяет, какие данные необходимо передать на представление.

UML-диаграмма компонентов приложения представлена на рисунке 3.4.

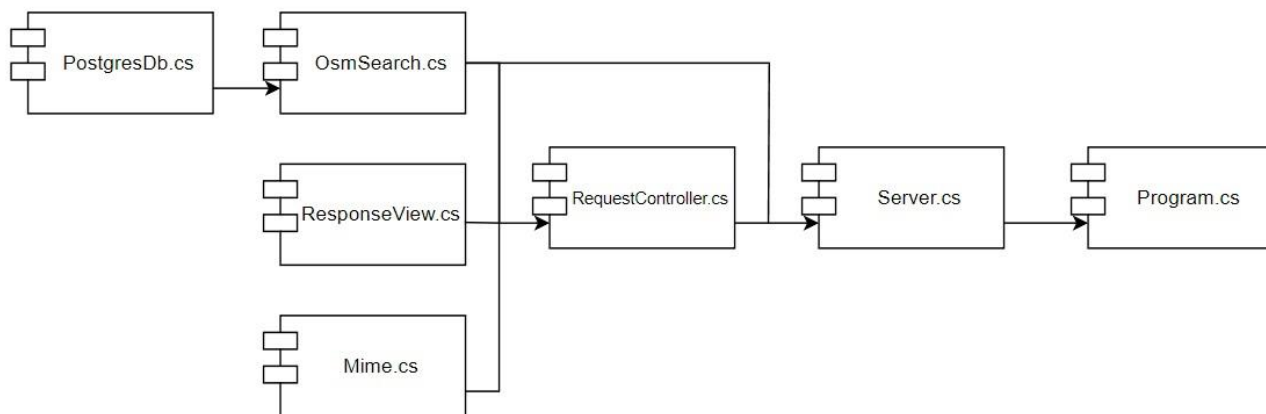


Рисунок 3.4 – UML-диаграмма компонентов приложения

Модель содержит функции, реализующие HTTP запросы к БД. Функция SeqAddrSearch выполняет отдельный адресный поиск.

Листинг 1. Функция SeqAddrSearch

```

public async Task<string> SeqAddrSearch(string city, string
street, string houseNumber)
{
    string connectionString =
"Host=localhost;Username=postgres;Password=123;Database=gis";
    string query = $"SELECT * FROM public.planet_osm_point
WHERE \"addr:city\" ILIKE '{city}' AND \"addr:street\" ILIKE
'{street}' AND \"addr:housenumber\" ILIKE '{houseNumber}';";

    PostgresDb pgsq1 = new PostgresDb(connectionString);

    DataTable resultTable = await pgsq1.ExecuteQuery(query);

    string jsonResult =
JsonConvert.SerializeObject(resultTable, Formatting.Indented);

    return jsonResult;
}
  
```

Функция AddrSearch выполняет полный адресный поиск, путем разбиения на определенные элементы поиска, а далее выполняет поиск по базе данных, функция FinInColumn выполняет поиск слов в ДБ, для их последующего разбиения.

Листинг 2. Функции AddrSearch() и FindInColumn()

```

public async Task<string> AddrSearch(string input)
{
    string[] words = input.Split(' ',
StringSplitOptions.RemoveEmptyEntries);
  
```

```

        string city = null;
        string street = null;
        string houseNumber = null;

        PostgresDb pgsql = new PostgresDb(connectionString);

        foreach (string word in words)
        {
            if (city == null)
            {
                city = await FindWordInColumn(pgsql, word,
"\"addr:city\"");
            }

            if (street == null)
            {
                street = await FindWordInColumn(pgsql, word,
"\"addr:street\"");
            }

            if (houseNumber == null)
            {
                houseNumber = await FindWordInColumn(pgsql, word,
"\"addr:housenumber\"");
            }

            if (city != null && street != null && houseNumber != null)
            {
                return await SeqAddrSearch(city, street, houseNumber);
            }

            return "{}";
        }

        private async Task<string> FindWordInColumn(PostgresDb pgsql,
string word, string columnName)
        {
            string query = $"SELECT DISTINCT {columnName} FROM
public.planet_osm_point WHERE {columnName} ILIKE '{word}'";
            DataTable resultTable = await pgsql.ExecuteQuery(query);

            if (resultTable.Rows.Count > 0)
            {
                return resultTable.Rows[0][columnName].ToString();
            }

            return null;
        }

```

Функция TagsSearch выполняет поиск объекта по набору OSM тегов

					ПГУ 1.090301.18.001 ПЗ	Лист
						41
Изм.	Лист	№ докум.	Подпись	Дата		

Листинг 3. TagsSearch()

```
public async Task<string> TagsSearch(List<string> tags)
{
    // Проверка на наличие тегов
    if (tags == null || tags.Count == 0)
    {
        return "{}";
    }

    // Создание объекта для работы с базой данных
    PostgresDb pgsql = new PostgresDb(connectionString);

    // Формирование SQL-запроса
    StringBuilder queryBuilder = new StringBuilder("SELECT
* FROM public.planet_osm_point WHERE ");
    for (int i = 0; i < tags.Count; i++)
    {
        queryBuilder.Append($"tags ILIKE '%{tags[i]}%'");
        if (i < tags.Count - 1)
        {
            queryBuilder.Append(" OR ");
        }
    }

    string query = queryBuilder.ToString();

    // Выполнение запроса
    DataTable resultTable = await
pgsql.ExecuteQuery(query);

    // Преобразование результата в JSON
    string jsonResult =
JsonConvert.SerializeObject(resultTable, Formatting.Indented);

    return jsonResult;
}
```

Далее рассмотрим функцию `HandleDefaultReq()` класса `RequestController`, она позволяет серверу отображать фронтенд страницы, а также позволяет серверу обрабатывать большое количество файлов.

Листинг 4. HandleDefaultReq ()

```
public async Task HandleDefaultReq(HttpListenerRequest request,
HttpListenerResponse response)
{
    // Получаем путь к выполняемой сборке
    string exePath = Assembly.GetExecutingAssembly().Location;
    string binPath = Path.GetDirectoryName(exePath);

    // Путь к корню проекта (относительно папки bin)
```

					ПГУ 1.090301.18.001 ПЗ	Лист
						42
Изм.	Лист	№ докум.	Подпись	Дата		

```

        string projectRoot =
Directory.GetParent(binPath).Parent.Parent.FullName;

        // Формируем полный путь к запрашиваемому файлу
        string reqUrl = (request.Url.AbsolutePath == "/" ) ?
"/pages/startPage/start.html" : request.Url.AbsolutePath;
        string filePath = Path.Combine(projectRoot,
reqUrl.TrimStart('/'));

        try
        {
            // Чтение содержимого файла и отправка его в ответе
            using (Stream stream = new FileStream(filePath,
FileMode.Open))
            {
                byte[] buffer = new byte[1024 * 16];
                int nbytes;
                string mime;

                response.ContentType =
MimeType.mimeTypeMappings.TryGetValue(Path.GetExtension(filePath),
out mime) ? mime : "application/octet-stream";
                response.ContentLength64 = stream.Length;
                response.AddHeader("Date",
DateTime.Now.ToString("r"));
                response.AddHeader("Last-Modified",
File.GetLastWriteTime(filePath).ToString("r"));

                while ((nbytes = await stream.ReadAsync(buffer, 0,
buffer.Length)) > 0)
                {
                    await response.OutputStream.WriteAsync(buffer,
0, nbytes);
                }
            }
            catch (Exception ex)
            {
                await ResponseView.SendErrorResponse(response,
ex.Message);
            }

            response.Close();
        }

```

Данная функция обращается к классу Mime для получения необходимого типа данных при HTTP запросе.

Листинг 5. Mime

```

public class MimeType
{

```

```

        public static IDictionary<string, string> mimeTypeMappings
=
        new Dictionary<string,
string>(StringComparer.InvariantCultureIgnoreCase)
        {
            {".asf", "video/x-ms-asf"},
            {".asx", "video/x-ms-asf"},
            {".avi", "video/x-msvideo"},
            {".bin", "application/octet-stream"},
            {".cco", "application/x-cocoa"},
            {".crt", "application/x-x509-ca-cert"},
            {".css", "text/css"},
            {".deb", "application/octet-stream"},
            {".der", "application/x-x509-ca-cert"},
            {".dll", "application/octet-stream"},
            {".dmg", "application/octet-stream"},
            {".ear", "application/java-archive"},
            {".eot", "application/octet-stream"},
            {".exe", "application/octet-stream"},
            {".flv", "video/x-flv"},
            {".gif", "image/gif"},
            {".hqx", "application/mac-binhex40"},
            {".htc", "text/x-component"},
            {".htm", "text/html"},
            {".html", "text/html"},
            {".ico", "image/x-icon"},
            {".img", "application/octet-stream"},
            {".iso", "application/octet-stream"},
            {".jar", "application/java-archive"},
            {".jardiff", "application/x-java-archive-
diff"},
            {".jng", "image/x-jng"},
            {".jnlp", "application/x-java-jnlp-file"},
            {".jpeg", "image/jpeg"},
            {".jpg", "image/jpeg"},
            {".js", "application/x-javascript"},
            {".mml", "text/mathml"},
            {".mng", "video/x-mng"},
            {".mov", "video/quicktime"},
            {".mp3", "audio/mpeg"},
            {".mpeg", "video/mpeg"},
            {".mpg", "video/mpeg"},
            {".msi", "application/octet-stream"},
            {".msm", "application/octet-stream"},
            {".msp", "application/octet-stream"},
            {".pdb", "application/x-pilot"},
            {".pdf", "application/pdf"},
            {".pem", "application/x-x509-ca-cert"},
            {".pl", "application/x-perl"},
            {".pm", "application/x-perl"},
            {".png", "image/png"},
            {".prc", "application/x-pilot"},
            {".ra", "audio/x-realaudio"},

```

```

manager"},
        {".rar", "application/x-rar-compressed"},
        {".rpm", "application/x-redhat-package-
manager"},
        {".rss", "text/xml"},
        {".run", "application/x-makeself"},
        {".sea", "application/x-sea"},
        {".shtml", "text/html"},
        {".sit", "application/x-stuffit"},
        {".swf", "application/x-shockwave-flash"},
        {".tcl", "application/x-tcl"},
        {".tk", "application/x-tcl"},
        {".txt", "text/plain"},
        {".war", "application/java-archive"},
        {".wbmp", "image/vnd.wap.wbmp"},
        {".wmv", "video/x-ms-wmv"},
        {".xml", "text/xml"},
        {".xpi", "application/x-xpinstall"},
        {".zip", "application/zip"},
    };
}

```

4 Реализация электронной карты мира

Фронтенд сторона приложения позволяет использовать формат XYZ tiles для отображение карты, в данном разделе я опишу создание карты, экспорт в растровый формат и использование ее в web-странице.

Была взята из открытых источников карта мира «Natural earth» [6], однако в ней отсутствуют новые регионы РФ. Используя инструменты редактирования векторных слоев, была проведена новая граница России, охватывающая новые регионы, что показано на рисунке 4.1.

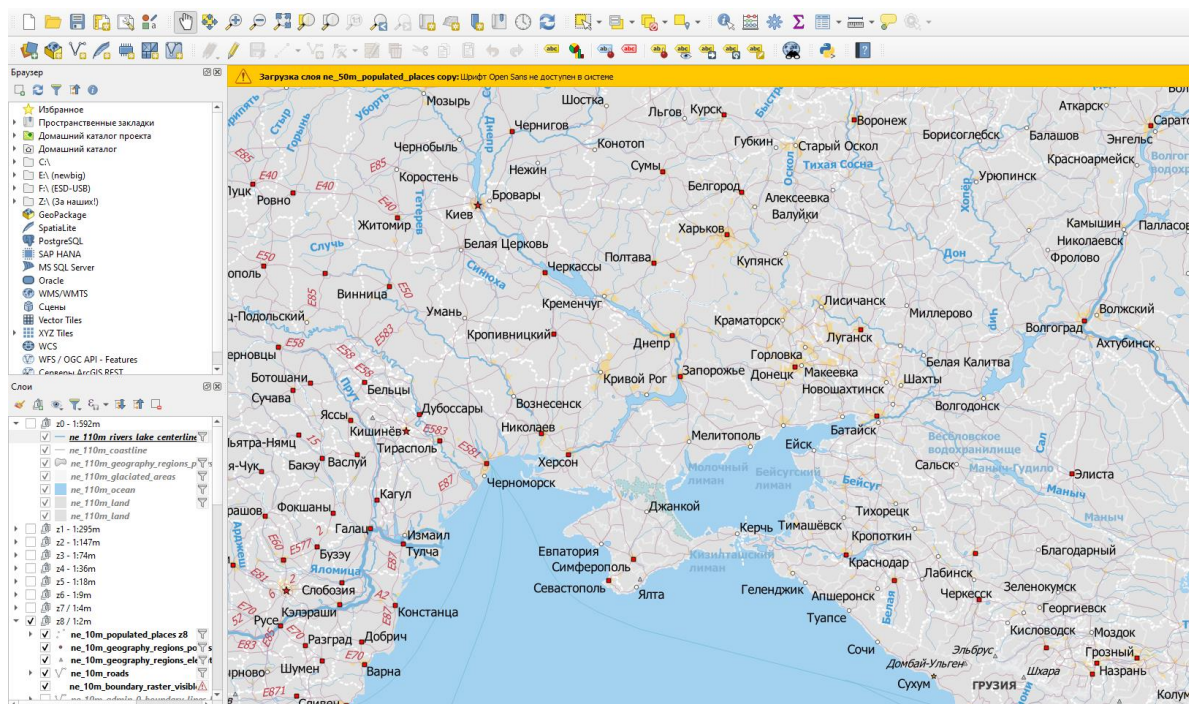


Рисунок 4.1 – Интерфейс QGIS, отредактированная карта мира.

Далее, с помощью инструмента растеризации карты в формат XYZ tiles карта была экспортирована в множество отдельных квадратов формата png. Подобный формат хранения карт используется во многих ГИС. Также, такой формат воспринимает JS библиотека OpenLayers, куда ее и можно импортировать. Фронтенд сервиса тоже использует данную библиотеку.

На рисунках 4.2 и 4.3 показан процесс экспорта карты и ее финальный вид как набор растровых изображений.

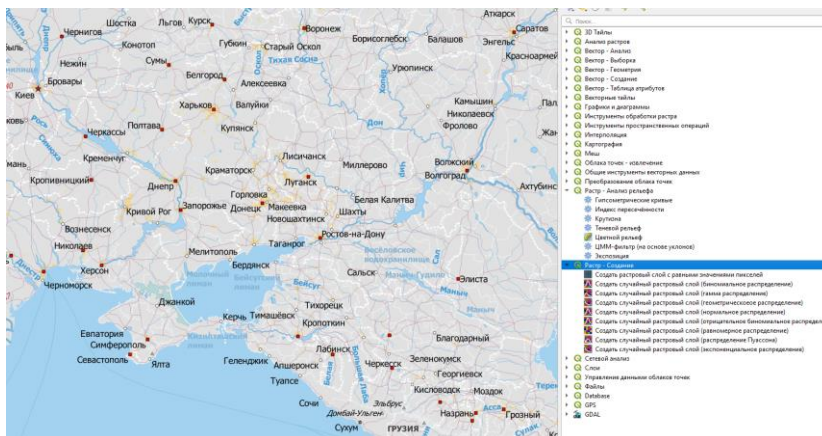


Рисунок 4.2 – Процесс преобразования карты в растровый формат

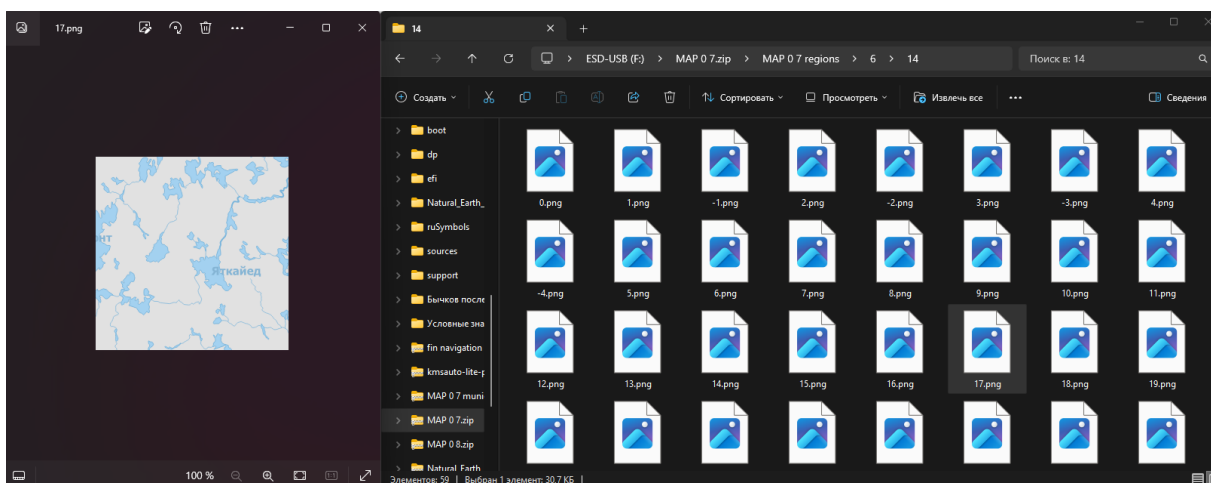


Рисунок 4.3 – Финальный вид карты

Изм.	Лист	№ докум.	Подпись	Дата

5 Руководство пользователя

5.1 Результат работы приложения

Чтобы запустить сервер, необходимо собрать приложение и выполнить .exe файл. После выполненных действий откроется консоль, сообщающая нам об успешном запуске сервера, что можно наблюдать на рисунке 5.1.

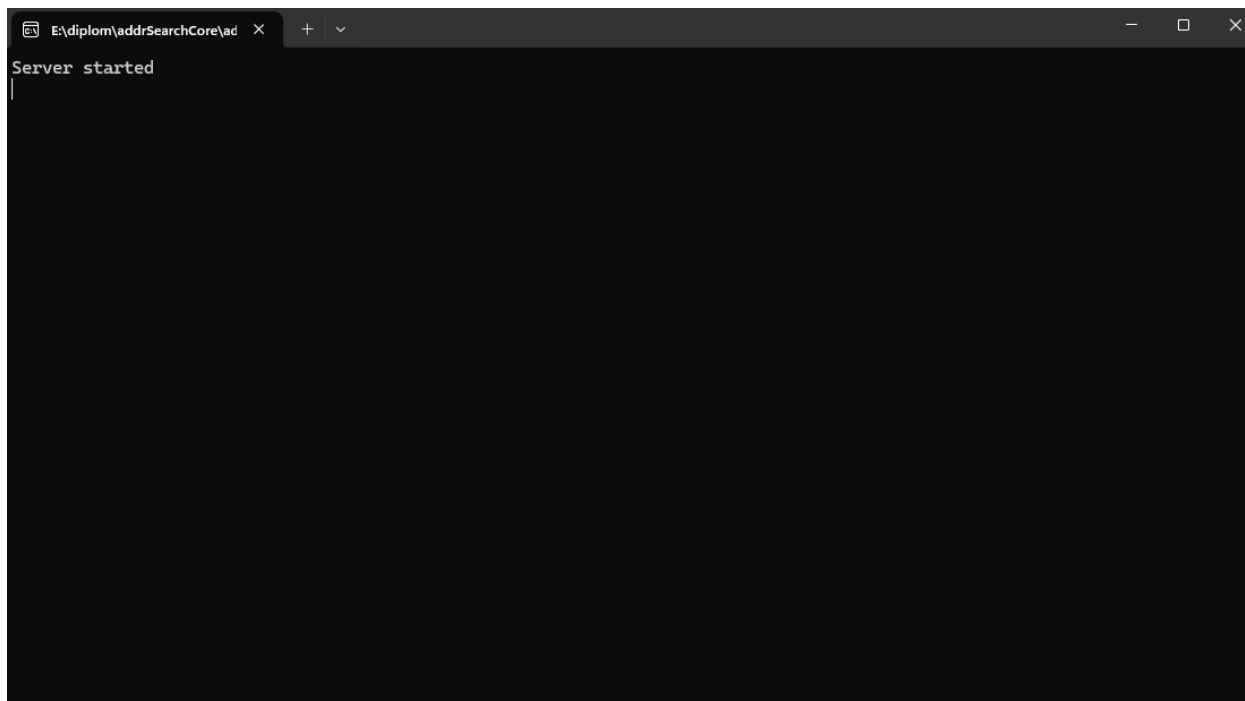


Рисунок 5.1 – Консоль сервера

5.1.1 Просмотр главной страницы сервера.

Переходим в браузере по следующему адресу: «ip адрес компьютера:4444». Мы увидим начальную страницу сервера, она представлена на рисунке 5.2.

Pages

- [searchPage](#)

Api

- [apiDocs](#)

Рисунок 5.2 – Начальная страница сервера

5.1.2 Адресный поиск.

Перейдем на страницу поиска (SearchPage) и выполним поиск адресный поиск, для этого перейдем на вкладку «Адресный поиск». Данная страница представлена на рисунке 5.3.

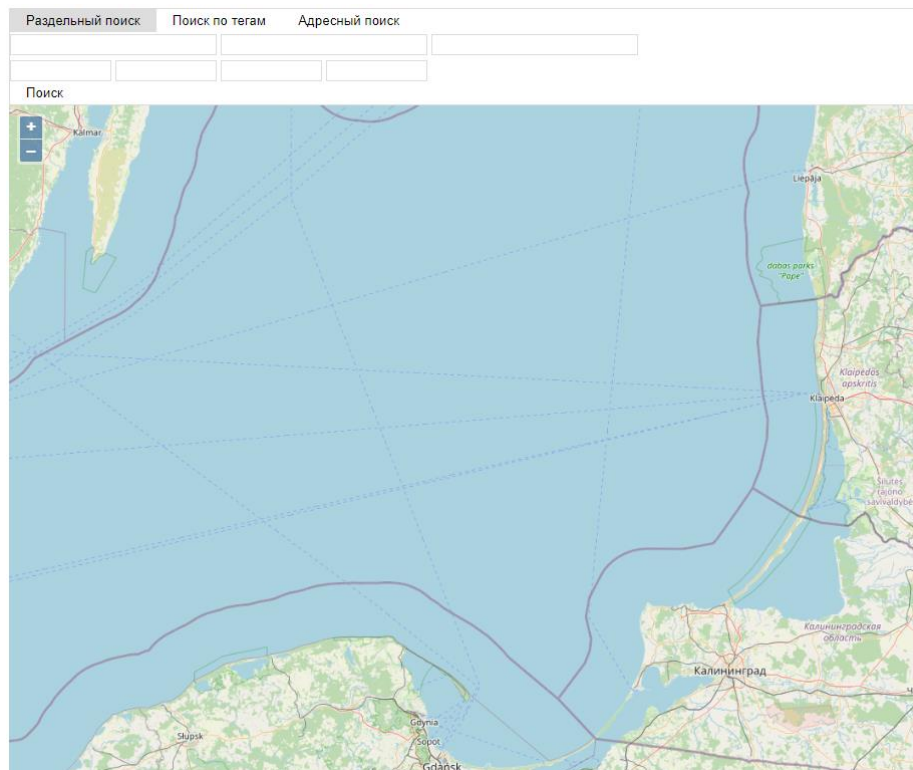


Рисунок 5.3 – Вкладка адресного поиска

Далее, введем данные. Введенные данные представлены на рисунке 5.4.

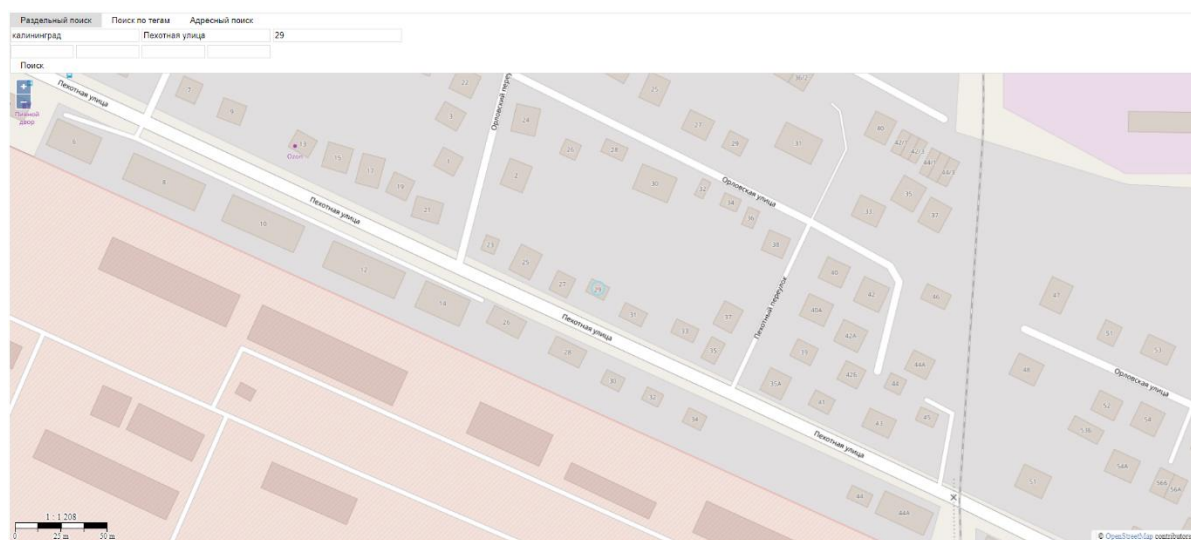


Рисунок 5.4 – Страница поиска с введенными данными

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.18.001 ПЗ

Лист

49

Нажимаем на кнопку «Поиск», Карта приближается до объекта, который мы искали, а также объект помечается знаком. Результат поиска представлен на рисунке 5.5.

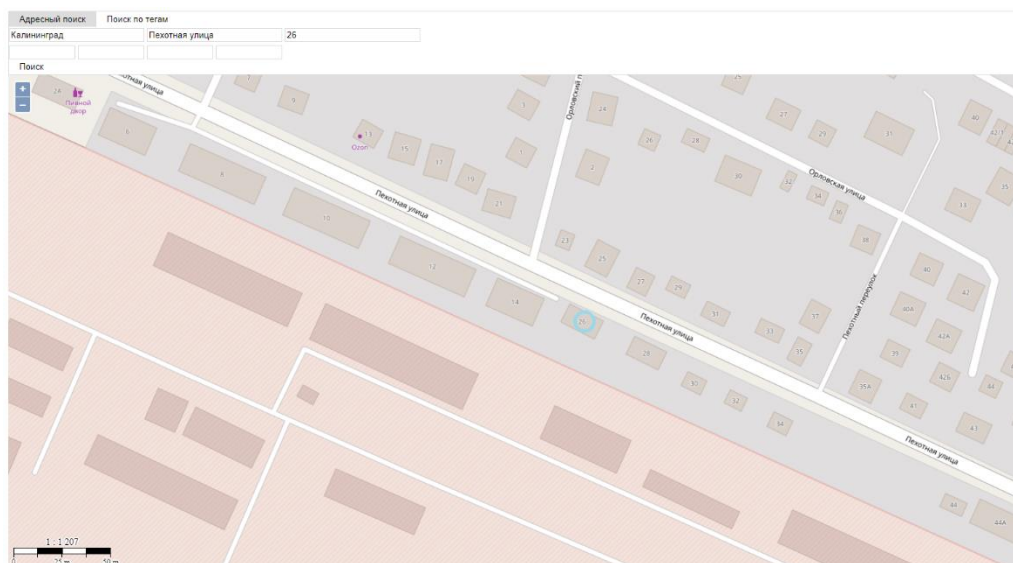


Рисунок 5.5 – Найденный объект

5.1.3 Адресный поиск

Чтобы выполнить адресный поиск, необходимо перейти на вкладку адресный поиск и ввести в строку поиска необходимый адрес и нажать на кнопку «Поиск». Результат поиска представлен на рисунке 5.6.

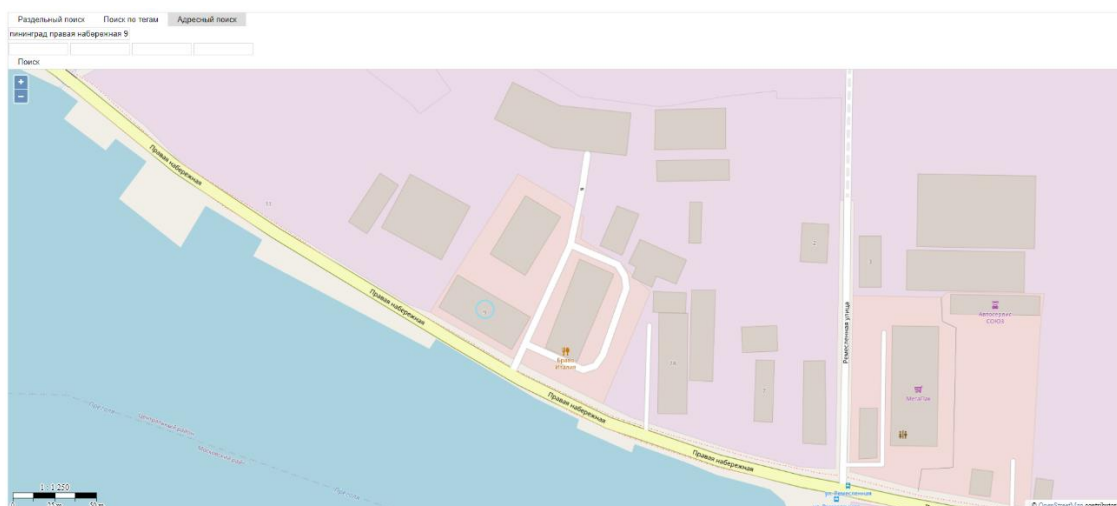


Рисунок 5.6 – Найденный объект

5.1.4 Поиск по тегам

Чтобы выполнить тестирование поиска по тегам удобнее всего будет воспользоваться какой-либо утилитой для тестирования API. В данном примере будет использоваться утилита «Postman».

Необходимо послать запрос на адрес «http://localhost:4444/tags_search», и в тело этого адреса необходимо поместить json строку определенной структуры. Пример запроса показан на рисунке 5.7.

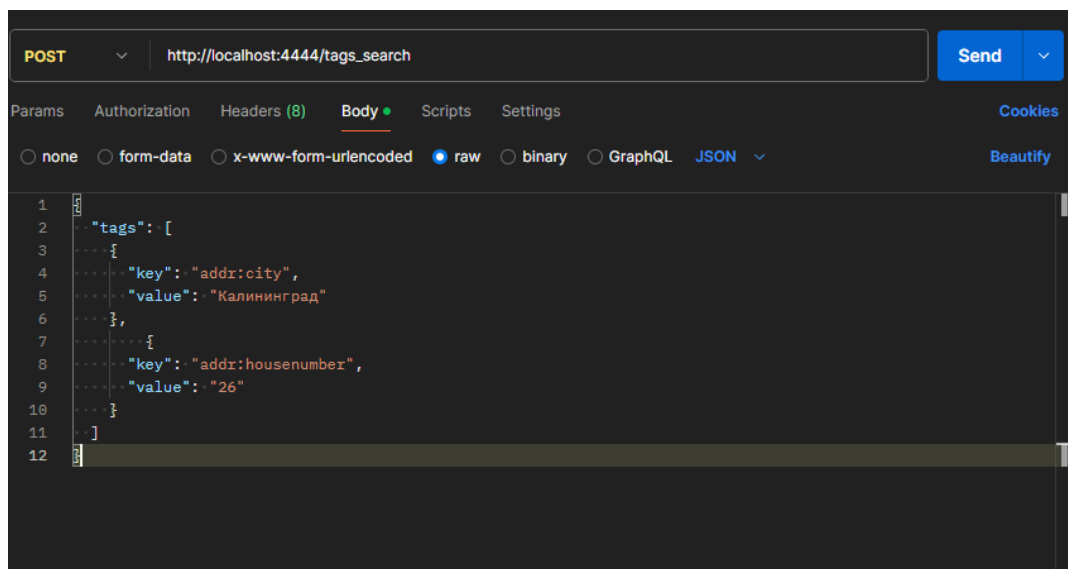


Рисунок 5.7 – Тело запроса tags_search

Далее, посылаем запрос и смотрим ответ сервера. В ответе – json строка, в которой список всех объектов, подходящих под условия поиска. Результат поиска показан на рисунке 5.8



Рисунок 5.8 – Результат поиска по тегам

6 Безопасность жизнедеятельности

При взаимодействии с компьютером, необходимо соблюдать меры безопасности, чтобы не было неприятных последствий для здоровья после использования. Если пользователь ПК будет пренебрегать техникой безопасности, то проблемы со здоровьем будут неизбежны.

Искусственное освещение, используемое в помещениях, где сотрудники непосредственно связаны с персональными электронно-вычислительными машинами, обязано обеспечивать условия для безопасности зрения, а также соответствовать оптимальным условиям зрительное восприятие, какое бывает при естественном освещении. Для этого выделяют следующие требования к освещению на рабочих местах, оборудованных ПЭВМ:

- освещенность рабочего места должна быть порядка 300 – 500 лк. Запрещается работать в темном или полутёмном пространстве, а также в помещении с избыточным количеством света;

- искусственное освещение в помещениях для эксплуатации ПЭВМ должно быть распределено равномерно;

- естественный свет должен падать преимущественно слева от рабочего стола. Нельзя располагать оборудованное дисплеем рабочее место таким образом, чтобы в поле зрения оператора не попадали окна или осветительные приборы, а также они не должны находиться непосредственно за спиной пользователя;

- яркость бликов на экране ПЭВМ не должна превышать 40 кд / м² и яркость потолка не должна превышать 200 кд / м².

В качестве источников общего искусственного освещения лучше использовать люминесцентные лампы типа ЛБ с рассеивателями или экранирующими решетками, поскольку они создают равномерную освещенность путем рассеянного или отраженного распределения света и исключают блики на экране монитора и клавиатуре. Соотношение яркости экрана и непосредственно ближайшего окружения не должно превышать 3:1.

Кроме того, на рабочие места необходимо устанавливать локальные источники света – настольные лампы с регулируемой яркостью и изменением расположения плафона.

Эти требования необходимо учитывать, поскольку без правильного размещения источников света на рабочих местах зрение сотрудников может падать, что приводит к хронической переутомляемости и усталости, а также утрате заинтересованности в результатах работы.

Помимо условий относительно освещенности помещения существуют правила организации рабочего стола:

- расстояние между рабочими столами с видеомониторами должно быть не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов – не менее 1,2 м;

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

- конструкция рабочего стола должна обеспечивать оптимальное размещение на рабочей поверхности используемого оборудования с учетом его количества и конструктивных особенностей, характера выполняемой работы;

- рабочее кресло должно быть подъемно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья, при этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию;

- сиденье и спинка стула должны быть покрыты не электризующимися полумягкими материалами;

Кроме того, применение дисплеев с антибликовым, антирадиационным покрытием или защитных экранов поможет снизить вредное воздействие на здоровье человека при использовании персонального компьютера.

Для комфортной работы с персональным компьютером необходимо обеспечить оптимальные значения температуры воздуха в помещении (19-23 градуса цельсия), а также относительную влажность воздуха (55%). Помещение,

где находятся компьютеры, должно быть достаточно просторным и хорошо проветриваемым.

В связи с напряженностью трудового процесса у пользователей ПЭВМ рекомендуется организовать рабочую смену путем чередования работ с использованием ПЭВМ и без него. Продолжительность непрерывной работы не должна превышать 1 часа, а продолжительность перерыва – от 10 до 15 минут. При работе с ПЭВМ в ночную смену (с 22 до 6 ч.) продолжительность регламентированных перерывов следует увеличить на 30%.

Помимо этого, рекомендуется проводить упражнения на глаза, снижающих напряжение. Поскольку, смотря в монитор, люди реже моргают, слезистая оболочка не получает требуемого количества влаги и пересыхает.

Существует общие рекомендации, позволяющие снизить усталость зрительного аппарата:

- часто поморгайте в течение пары минут;
- открывайте и закрывайте глаза с периодичностью 3-5 секунд;
- легкими движениями нажимайте на веки;
- переводите взгляд с удалённых объектов на близко расположенные предметы, фокусируя взгляд на протяжении пяти секунд.

Заключение

В ходе выполнения выпускной квалификационной работы реализовано клиент-серверное приложение, для чего использован современный стек технологий в области геоинформационной разработки, включающий инструменты и библиотеки для создания и редактирования географических баз данных.

Результатом работы является клиент-серверное приложение «Сервис Адресного географического поиска», а также пояснительная записка дипломного проекта, в которой представлены описание разработки и обзор его функциональных возможностей.

Разработанное приложение предоставляет пользователю удобную возможность адресного поиска, который позволяет пользователю получать информацию о местоположении географических объектов на карте.

Задание на дипломное проектирование выполнено в полном объеме.

В дальнейшем приложение может быть усовершенствовано, путём расширения функционала: полноценный ГИС интерфейс, возможность поиска пути, более точная электронная карта.

					ПГУ 1.090301.18.001 ПЗ	Лист
						55
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. Официальный ресурс публикации правовых актов [Электронный ресурс]/Режим доступа: <http://publication.pravo.gov.ru/>, свободный. (Дата обращения 26.04.2024)
2. Документация C# [Электронный ресурс]/Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/csharp/>, свободный. (Дата обращения 28.04.2024)
3. Документация PostGis [Электронный ресурс]/Режим доступа: <https://postgis.net/documentation/>?, свободный. (Дата обращения 29.04.2024)
4. Джон Дакетт. "HTML и CSS: дизайн и создание веб-сайтов". 2011 – 512 с
5. Флэнаган Д. JavaScript. Карманный справочник. Сделайте веб-страницы интерактивными! / Перевод А.Г. Сысолюк. — Москва.: Издательский дом "Вильямс", 2015. — 320 с.
6. Ресурс, предоставляющий векторные и растровые карты [Электронный ресурс]/Режим доступа: <https://www.naturalearthdata.com/downloads/>, (Дата обращения 29.04.2024)
7. Некоммерческий веб-картографический проект [Электронный ресурс]/Режим доступа: <https://wiki.openstreetmap.org/>, (Дата обращения 30.04.2024)

Приложение А

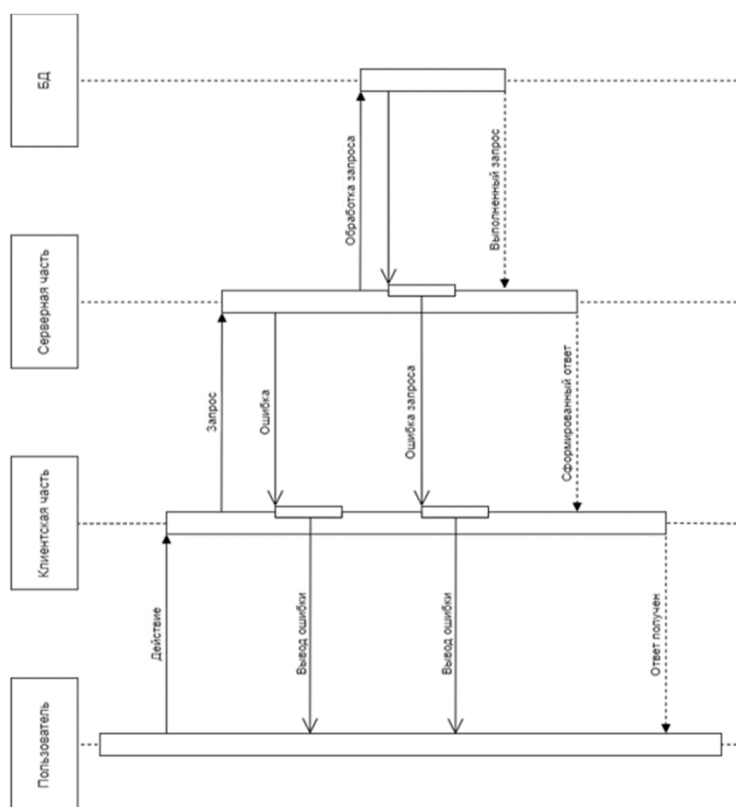
(Обязательное)

Клиент серверное приложение «Сервис адресного географического поиска»

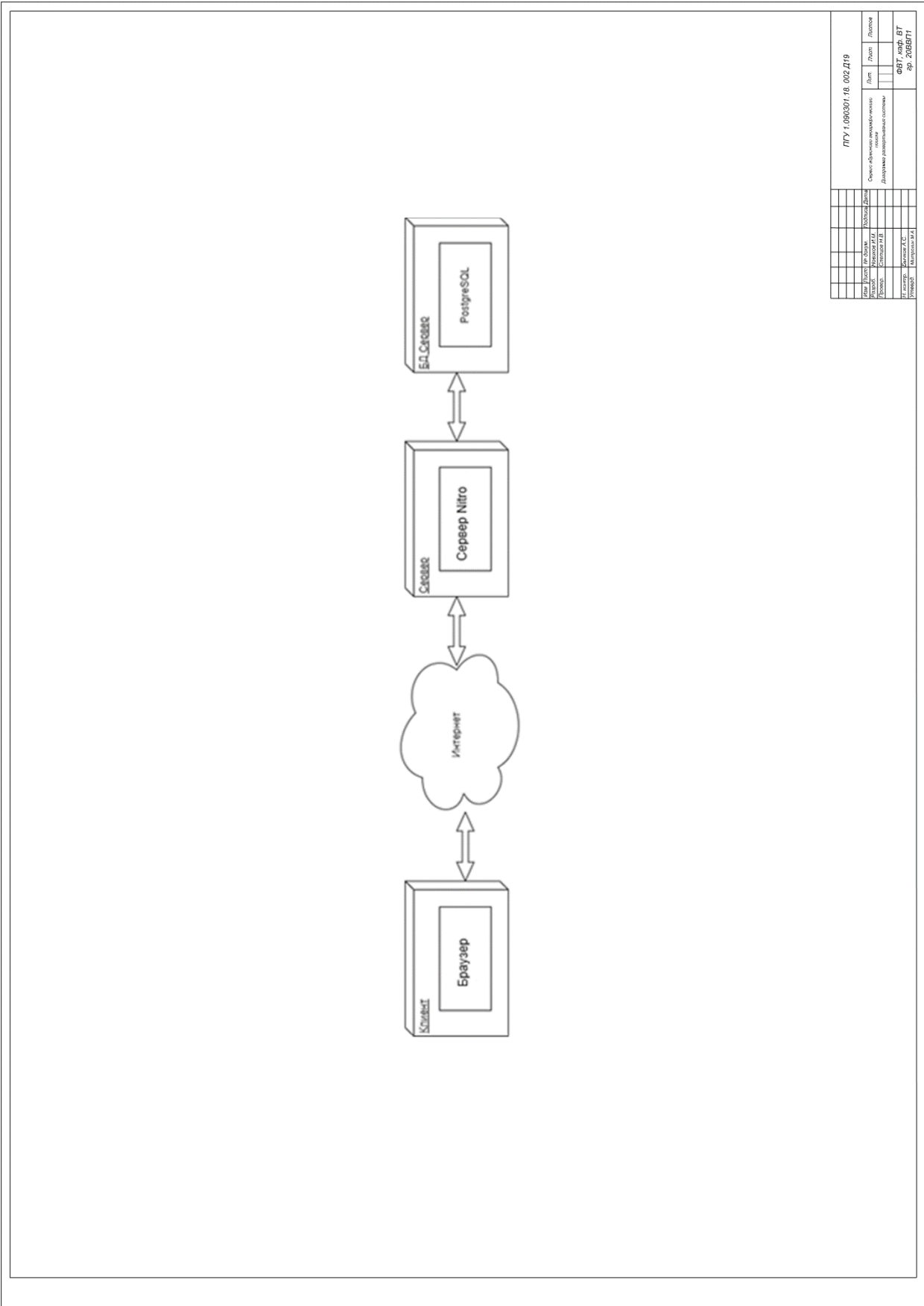
UML-диаграммы

					ПГУ 1.090301.18.001 ПЗ	Лист
						57
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1–Диаграмма последовательностей

[illegible]

Приложение А.2–Диаграмма развертывания

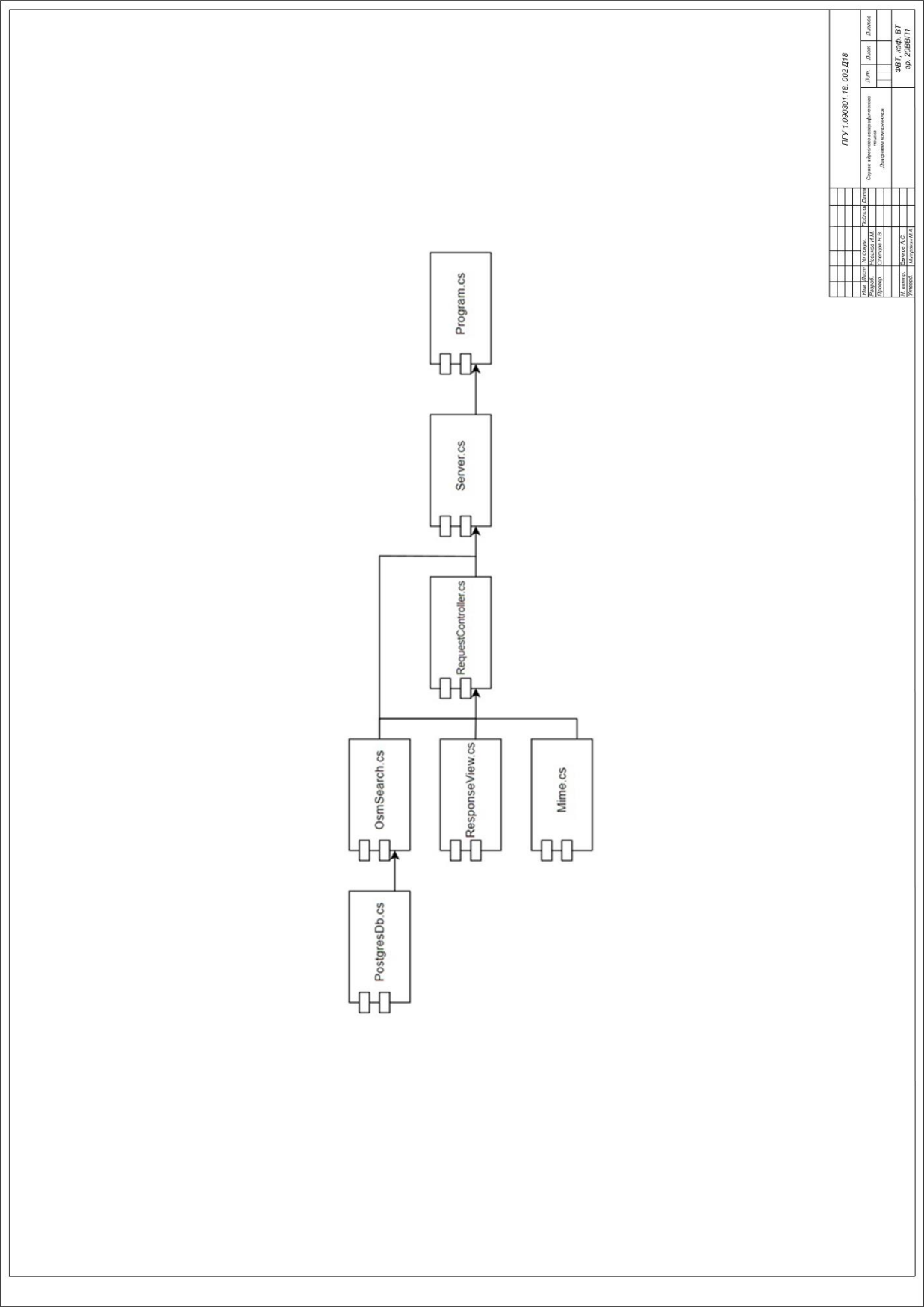


ПГУ 1.090301.18.002 Д19									
Имя	Должность	Подпись	Дата	Лист	Сервер: объект, на котором выполняется задание				
Имя	Должность	Подпись	Дата	Лист	Данные: информация о состоянии				
Имя	Должность	Подпись	Дата	Лист	Данные: информация о состоянии				
Имя	Должность	Подпись	Дата	Лист	Данные: информация о состоянии				
Имя	Должность	Подпись	Дата	Лист	Данные: информация о состоянии				
Имя	Должность	Подпись	Дата	Лист	Данные: информация о состоянии				
Имя	Должность	Подпись	Дата	Лист	Данные: информация о состоянии				
Имя	Должность	Подпись	Дата	Лист	Данные: информация о состоянии				
Имя	Должность	Подпись	Дата	Лист	Данные: информация о состоянии				

Изм.	Лист	№ докум.	Подпись	Дата

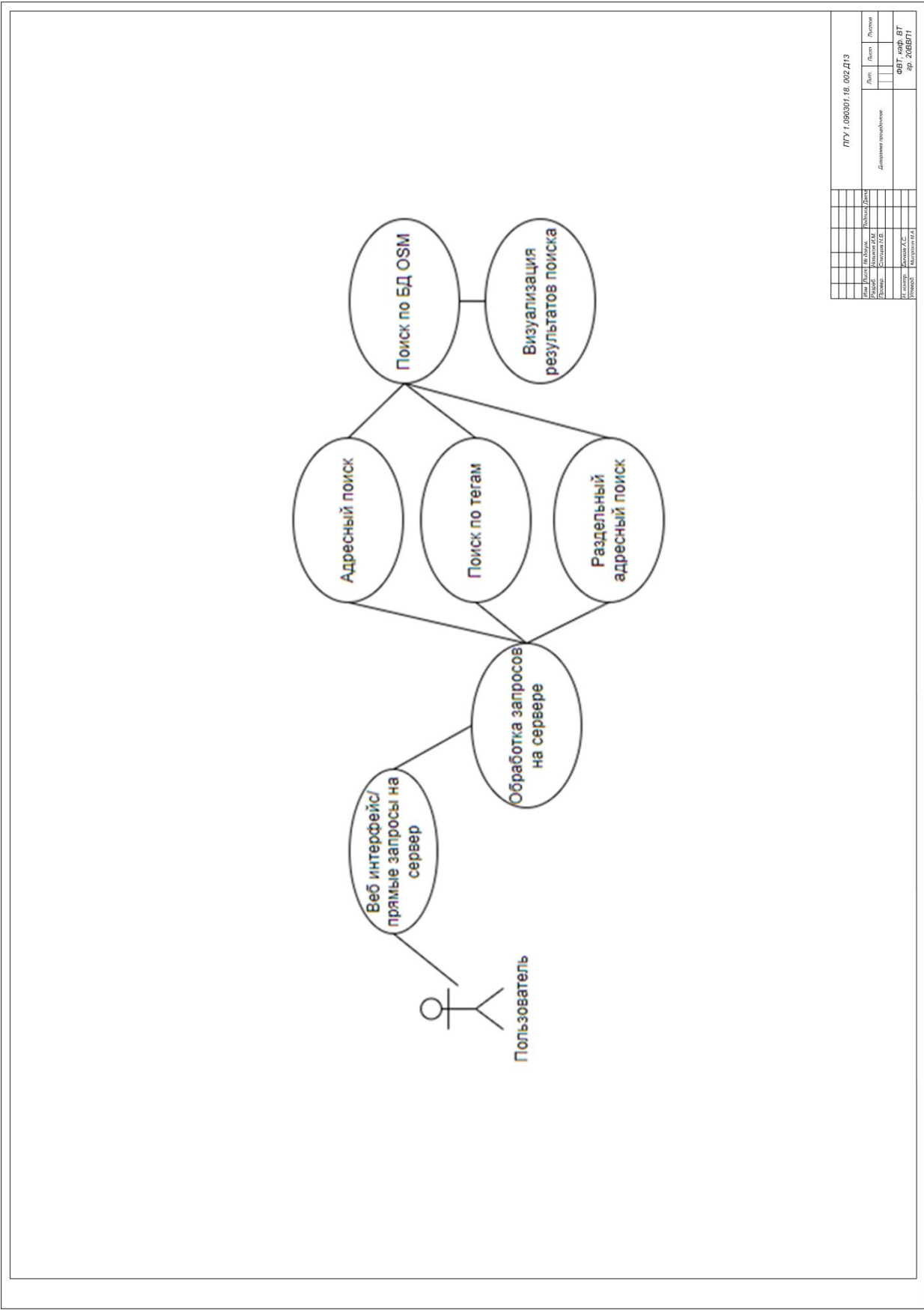
ПГУ 1.090301.18.001 ПЗ

Приложение А.3–Диаграмма компонентов



ПГУ 1.090301.18.002 Д18									

Приложение А.4—Диаграмма прецедентов



ПГУ 1.090301.18.002 Д13									
						</			

Изм.	Лист	№ докум.	Подпись	Дата

Приложение Б

(Обязательное)

Сервис адресного географического поиска

Листинг программы

					ПГУ 1.090301.18.001 ПЗ	Лист
						62
Изм.	Лист	№ докум.	Подпись	Дата		

Файл index.html

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8" />
  <title>search</title>
  <script src="./js/ol/ol.js"></script>
  <script src="./js/jquery-3.7.1.min.js"></script>
  <script src="./js/jquery-ui.js"></script>
  <script src="./js/gioApi.js"></script>
  <script src="./main.js"></script>
  <link rel="stylesheet" href="./css/jquery-ui.css" />
  <link rel="stylesheet" href="./css/styles.css" />
  <link rel="stylesheet" href="./js/ol/ol.css" />
</head>

<body>

  <div class="tabs">
    <div>
      <button id="btnTab1" class="tabllinks"
onclick="openTab(event, 'tab1')">Раздельный поиск</button>
      <button id="btnTab2" class="tabllinks"
onclick="openTab(event, 'tab2')">Поиск по тегам</button>
      <button id="btnTab3" class="tabllinks"
onclick="openTab(event, 'tab3')">Адресный поиск</button>
    </div>

    <div id="tab1" class="tabcontent">
      <input type="text" id="cityInput"
class="inputColumn">
      <input type="text" id="streetInput"
class="inputColumn">
      <input type="text" id="numberInput"
class="inputColumn">
      <ul id="cityList" class="columnList"></ul>
      <br />
      <input type="text" onkeyup="this.value =
this.value.replace(/\D+/g, '');" id="lonInput1" class="inputPoints">
      <input type="text" id="latInput1"
class="inputPoints">
      <input type="text" id="lonInput2"
class="inputPoints">
      <input type="text" id="latInput2"
class="inputPoints">
      <!--<input type="text" id="streetInput"
class="inputColumn">
      <input type="text" id="numberInput"
class="inputColumn">-->
      <!--<ul id="autoList" class="ui-widget"></ul>-->
      <br>
    </div>
  </div>
</body>
</html>
```

```

        <button id="searchBtn">Поиск</button>
    </div>
    <div id="tab2" class="tabcontent">
        <div>
            <div class="tags_inputs">
                <input type="text" id="key_input" />
                <input type="text" id="value_input" />
            </div>
            <div class="tags_inputs">
                <input type="text" id="cast_type_input" />
                <input type="text" id="operation_input" />
            </div>
        </div>
        <div>
            <div class="tags_inputs">
                <div class="tags_inputs2">
                    <input type="text" id="rect_input" />
                    <textarea
id="tags_text_area"></textarea>
                </div>
            </div>
        </div>
    </div>
    <div id="tab3" class="tabcontent">
        <input type="text" id="cityInput"
class="inputColumn">
        <ul id="cityList" class="columnList"></ul>
        <br />
        <input type="text" onkeyup="this.value =
this.value.replace(/\D+/g, ''); " id="lonInput1" class="inputPoints">
        <input type="text" id="latInput1"
class="inputPoints">
        <input type="text" id="lonInput2"
class="inputPoints">
        <input type="text" id="latInput2"
class="inputPoints">
        <!--<input type="text" id="streetInput"
class="inputColumn">
        <input type="text" id="numberInput"
class="inputColumn">-->
        <!--<ul id="autoList" class="ui-widget"></ul>--
>
        <br>
        <button id="searchBtn">Поиск</button>
    </div>
</div>
<div>

    <div id="map" class="map"></div>
</div>
</body>

</html>

```

Файл main.js

```
const url = "http://localhost:16000";
const serverUrl = "http://3.1.48.33";
const layer_world_id =
'map_rus_karta_mira_7f5ad9f68f01ac9eab0e32ba854f1e93_land';
const layer_geotiff_id = 'geotiff_main';
const layer_osm_id =
'osm_src_full_4_8f362df92b1ebc4cd570e80d71b354f5';
const layer_XYZ_name = 'world 0 10';

var map;

var webgio = {
  url: 'http://3.1.48.33:16000/',
  proj: 'EPSG:900913',
};

function openTab(evt, tabName) {
  let i, tabcontent, tablinks;

  tabcontent = document.getElementsByClassName("tabcontent");
  for (i = 0; i < tabcontent.length; i++) {
    tabcontent[i].style.display = "none";
  }
  tablinks = document.getElementsByClassName("tablinks");
  for (i = 0; i < tablinks.length; i++) {
    tablinks[i].className = tablinks[i].className.replace("
active", "");
  }
  document.getElementById(tabName).style.display = "block";
  evt.currentTarget.className += " active";
}

function createMap() {

  const osmLayer = new ol.layer.Tile({
    source: new ol.source.OSM()
  });

  map = new ol.Map({
    controls: ol.control.defaults().extend([new
ol.control.ScaleLine({ bar: true, text: true })]),
    interactions: ol.interaction.defaults().extend([new
ol.interaction.DragRotateAndZoom()]),

    layers: [
      osmLayer

    ],

    target: 'map',
    view: new ol.View({
```

```

        center: [0, 0],
        zoom: 2,
    })
    })
}

function addPoint(lon, lat, zoomFlag) {

    map.getLayers().forEach(layer => {
        if (layer.get('name') == 'vectorLayer') {
            map.removeLayer(layer)
        }
    });

    var geomPoint = new ol.geom.Point(ol.proj.fromLonLat([lon,
lat]));

    const point = new ol.Feature({
        geometry: geomPoint
    });

    const vectorSource = new ol.source.Vector({
        features: [point]
    });

    const vectorLayer = new ol.layer.Vector({
        source: vectorSource,
        crossOrigin: "Anonymous",
        name: "vectorLayer"
    });

    map.addLayer(vectorLayer);

    if (zoomFlag) {
        const view = map.getView();
        view.setCenter(ol.proj.fromLonLat([lon, lat]));
        view.setZoom(10);
    }

}

async function listenerFunc(el) {
    const osmObj = await getOsmByAoguid(el.aoguid, el.region,
el.house_num);
}

async function autocompleteFunc(userInput, inputObj, column,
filter) {
    const ulList = document.getElementById("cityList");

    if (userInput.length > 1) {
        const result = await columnSearch(column, userInput,
10, filter);

```

```

let source = [];
let elements = [];
ulList.innerHTML = '';

for (let i in result) {

    source[i] = result[i];
    elements[i] = document.createElement("li");
    elements[i].innerText = source[i];
    elements[i].class = "columnList";
    ulList.appendChild(elements[i]);
    elements[i].addEventListener('click', async
function (event) {
        //listenerFunc(result[i]);
        ulList.innerHTML = '';
        inputObj.value = source[i];
    });
}
console.log(source);
}
else
    ulList.innerHTML = '';
}

function objToXml(tag, obj) {
    let xml = "<" + tag + ">";
    for (let key in obj) {
        xml += "\n <" + key + ">" + obj[key] + "</" + key +
">";
    }
    xml += "\n</" + tag + ">";

    return (xml);
}

async function seqSearch(city, street, housenumber, limit) {
    try {

        const addrObj = {
            city: city,
            street: street,
            housenumber: housenumber,
            limit: limit
        }

        const bodyStr = JSON.stringify(addrObj);

        const response = await fetch(serverUrl +
"/osm_seq_search", {
            method: "POST",
            headers: {
                "Content-Type": "application/json; charset=utf-
8"

```

```

        },
        body: bodyStr
    });

    if (!response.ok) {
        throw new Error("seqSearch request failed")
    }

    const responseData = await response.json();
    console.log("responseData: ", responseData);
    addPoint(responseData[0].lon, responseData[0].lat,
true);

    return responseData;
}
catch (err) {
    console.log(err);
}
}

async function columnSearch(column, string, limit, filter) {
    try {

        const bodyStr = createColumnJsonObj(column, string,
limit, filter);

        console.log("bodyStr " + bodyStr);

        const response = await fetch(serverUrl +
"/column_search", {
            method: "POST",
            headers: {
                "Content-Type": "application/json; charset=utf-
8"

            },
            body: bodyStr
        });

        if (!response.ok) {
            throw new Error("seqSearch request failed")
        }

        const responseData = await response.json();
        return responseData;
    }
    catch (err) {
        console.log(err);
    }
}

function createColumnJsonObj(column, string, limit, addObj) {
    var jsonStr = "";

    jsonStr += "{ \"column\": \"" + column + "\", ";

```

```

jsonStr += "\"string\": \"" + string + "\", ";
jsonStr += "\"limit\": " + limit;

if (addObj.length > 0) {

    jsonStr += ", \"filter\": {";

    var nullCount = 0;

    addObj.forEach(function (el, i) {
        jsonStr += el;

        if (i < addObj.length - 1)
            jsonStr += ", ";
    })

}

//for (let i = 0; i < addObj.length; i++) {

//    jsonStr += addObj[i];
//    console.log(addObj[i]);

//    //const item = addObj[i];
//    //const key = Object.keys(item)[0];
//    //const value = item[key];

//    //jsonStr += "\"" + key + "\": \"" + value + "\"";
//    if (addObj.length > 0)
//        jsonStr += ", ";
//}

jsonStr += "}}"

if (nullCount == addObj.length)
    return null
else
    return jsonStr;

}

function sendColumnReq(event, field, addObj) {

    var filter = []
    var currentField;

;

    console.log(addObj);

    addObj.forEach(function (el) {
        switch (el.id) {
            case 'cityInput':
                //filter.push({ city: el.value });
                if (el.value != "")

```

```

        filter.push("\"city\": " + "\"" + el.value
+ "\"");
        break;
    case 'streetInput':
        //filter.push({ street: el.value });
        if (el.value != "")
            filter.push("\"street\": " + "\"" +
el.value + "\"");
        break;
    case 'numberInput':
        //filter.push({ housenumber: el.value });
        if (el.value != "")
            filter.push("\"housenumber\": " + "\"" +
el.value + "\"");
        break;
    }
})

switch (field.id) {
    case 'cityInput':
        currentField = "city";
        break;
    case 'streetInput':
        currentField = "street";
        break;
    case 'numberInput':
        currentField = "housenumber";
        break;
}

autocompleteFunc(event.target.value, field, currentField,
filter);
}

$(document).ready(function () {
    //http://3.1.48.33/pages/searchPage/index.html
    createMap();

    let tabObj = document.getElementById("tab1");          //
показ 1й вкладки по умолчанию
    tabObj.style.display = "block";

    const searchBtn = document.getElementById("searchBtn");
    const cityField = document.getElementById("cityInput");
    const streetField = document.getElementById("streetInput");
    const numberField = document.getElementById("numberInput");
    const lanField = document.getElementById("lanInput");
    const lotField = document.getElementById("lotInput");

    const map = document.getElementById("map");

    cityField.addEventListener('input', async function (event)
{

```

```

        let addObj = [];
        addObj.push(streetField);
        addObj.push(numberField);
        sendColumnReq(event, cityField, addObj);
    });

    streetField.addEventListener('input', async function
(event) {
        let addObj = [];
        addObj.push(cityField);
        addObj.push(numberField);
        sendColumnReq(event, streetField, addObj);
    });

    numberField.addEventListener('input', async function
(event) {
        let userInput = event.target.value;
        if (userInput.length > 1) {
            let addObj = [];
            addObj.push(streetField);
            addObj.push(cityField);
            sendColumnReq(event, numberField, addObj);
        }
    });

    searchBtn.addEventListener("click", function (event) {
        var pointObj = seqSearch(cityField.value,
streetField.value, numberField.value, 1);
    });

    map.addEventListener("click", function (event) {
        const ulList = document.getElementById("cityList");
        ulList.innerHTML = '';
    });

});

```

Файл styles.html

```

.autoInput {

}

.columnList {
    width: 200px;
    list-style-type: none;
    position: absolute;
    z-index: 100;
    padding-left: 0;
    margin-top: 0;
}

```

```

.inputColumn {
    width: 200px;
    padding: 2px;
    border: 1px solid;
    outline: none;
    border-color: #dddddd;
}

.inputPoints {
    margin-top: 5px;
    width: 95px;
    padding: 2px;
    border: 1px solid;
    outline: none;
    border-color: #dddddd;
}

li {
    padding: 2px;
    border: 1px solid;
    width: 200px;
    border-color: #dddddd;
    border-top: none;
    background-color: #ffffff;
    cursor: pointer;
}

li:hover {
    background-color: #f1f1f1;
}

.map {
    left: 0;
    position: relative;
    width: 100%;
    height: 750px;
}

.tabBtn {
    border: 1px solid;
    border-color: #dddddd;
    border-bottom: none;
    background-color: #ffffff;
    cursor: pointer;
}

.tabBtn:hover {
    background-color: #f1f1f1;
}

.inputDiv{
    height: 75px;

```

```

}

.seqInput{
    margin: 2px;
}

.tabs {
    overflow: hidden;
    border: 1px solid #dddddd;
    background-color: #ffffff;
}

.tabs button{
    background-color: inherit;
    float: left;
    border: none;
    outline: none;
    cursor: pointer;
    padding: 4px 16px;
    transition: 0.1s;
}

.tabs button:hover {
    background-color: #dddddd;
}

.tabs button.active {
    background-color: #dddddd;
}

.tabcontent {
    display: none;
    padding-top: 25px;
}

.tags_inputs{
    padding: 5px;
}

```

Файл index.html (главная страница)

```

<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8" />
    <title>GioServer</title>
</head>

<body>

```

```

<div>
    <h3>Pages</h3>
    <ul>
        <li> <a
href="/pages/searchPage/index.html">searchPage</a> </li>
    </ul>
    <h3>Api</h3>
    <ul>
        <li> <a
href="/pages/startPage/docs.html">apiDocs</a> </li>
    </ul>
</div>
</body>

</html>

```

Файл Mime.cs

```

using System;
using System.Collections.Generic;
using System.Text;
namespace addrSearchCore
{
    public class MimeType
    {
        public static IDictionary<string, string>
mimeTypeMappings =
        new Dictionary<string,
string>(StringComparer.InvariantCultureIgnoreCase)
        {
            {".asf", "video/x-ms-asf"},
            {".asx", "video/x-ms-asf"},
            {".avi", "video/x-msvideo"},
            {".bin", "application/octet-stream"},
            {".cco", "application/x-cocoa"},
            {".crt", "application/x-x509-ca-cert"},
            {".css", "text/css"},
            {".deb", "application/octet-stream"},
            {".der", "application/x-x509-ca-cert"},
            {".dll", "application/octet-stream"},
            {".dmg", "application/octet-stream"},
            {".ear", "application/java-archive"},
            {".eot", "application/octet-stream"},
            {".exe", "application/octet-stream"},
            {".flv", "video/x-flv"},
            {".gif", "image/gif"},
            {".hqx", "application/mac-binhex40"},
            {".htc", "text/x-component"},
            {".htm", "text/html"},
            {".html", "text/html"},
            {".ico", "image/x-icon"},
            {".img", "application/octet-stream"},
            {".iso", "application/octet-stream"},

```

```

diff"},
        {" .jar", "application/java-archive"},
        {" .jardiff", "application/x-java-archive-
diff"},
        {" .jng", "image/x-jng"},
        {" .jnlp", "application/x-java-jnlp-file"},
        {" .jpeg", "image/jpeg"},
        {" .jpg", "image/jpeg"},
        {" .js", "application/x-javascript"},
        {" .mml", "text/mathml"},
        {" .mng", "video/x-mng"},
        {" .mov", "video/quicktime"},
        {" .mp3", "audio/mpeg"},
        {" .mpeg", "video/mpeg"},
        {" .mpg", "video/mpeg"},
        {" .msi", "application/octet-stream"},
        {" .msm", "application/octet-stream"},
        {" .msp", "application/octet-stream"},
        {" .pdb", "application/x-pilot"},
        {" .pdf", "application/pdf"},
        {" .pem", "application/x-x509-ca-cert"},
        {" .pl", "application/x-perl"},
        {" .pm", "application/x-perl"},
        {" .png", "image/png"},
        {" .prc", "application/x-pilot"},
        {" .ra", "audio/x-realaudio"},
        {" .rar", "application/x-rar-compressed"},
        {" .rpm", "application/x-redhat-package-
manager"},
        {" .rss", "text/xml"},
        {" .run", "application/x-makeself"},
        {" .sea", "application/x-sea"},
        {" .shtml", "text/html"},
        {" .sit", "application/x-stuffit"},
        {" .swf", "application/x-shockwave-flash"},
        {" .tcl", "application/x-tcl"},
        {" .tk", "application/x-tcl"},
        {" .txt", "text/plain"},
        {" .war", "application/java-archive"},
        {" .wbmp", "image/vnd.wap.wbmp"},
        {" .wmv", "video/x-ms-wmv"},
        {" .xml", "text/xml"},
        {" .xpi", "application/x-xpinstall"},
        {" .zip", "application/zip"},
    };
}
public class SearchParams
{
    public string City { get; set; }
    public string Street { get; set; }
    public string Housenumber { get; set; }
    public string Input { get; set; }
}
}

```

Файл OsmSearch.cs

```
using Npgsql;
using System;
using System.Collections.Generic;
using System.Data;
using System.Text;
using System.Threading.Tasks;
using System.Xml;

namespace addrSearchCore
{
    public class OsmSearch
    {
        public async Task<string> TagsSearch(List<string> tags)
        {
            // Проверка на наличие тегов
            if (tags == null || tags.Count == 0)
            {
                return "{}";
            }

            // Создание объекта для работы с базой данных
            PostgresDb pgsql = new
            PostgresDb(connectionString);

            // Формирование SQL-запроса
            StringBuilder queryBuilder = new
            StringBuilder("SELECT * FROM public.planet_osm_point WHERE ");
            for (int i = 0; i < tags.Count; i++)
            {
                queryBuilder.Append($"tags ILIKE
            '%{tags[i]}%'");
                if (i < tags.Count - 1)
                {
                    queryBuilder.Append(" OR ");
                }
            }

            string query = queryBuilder.ToString();

            // Выполнение запроса
            DataTable resultTable = await
            pgsql.ExecuteQuery(query);

            // Преобразование результата в JSON
            string jsonResult =
            JsonConvert.SerializeObject(resultTable, Formatting.Indented);

            return jsonResult;
        }
    }
}
```

```

    }

    public async Task<string> SeqAddrSearch(string city, string
street, string houseNumber)
    {
        string connectionString =
"Host=localhost;Username=postgres;Password=123;Database=gis";
        string query = $"SELECT * FROM
public.planet_osm_point WHERE \"addr:city\" ILIKE '{city}' AND
\"addr:street\" ILIKE '{street}' AND \"addr:housenumber\" ILIKE
'{houseNumber}';";

        PostgresDb pgsql = new
PostgresDb(connectionString);

        DataTable resultTable = await
pgsql.ExecuteQuery(query);

        string jsonResult =
JsonConvert.SerializeObject(resultTable, Formatting.Indented);

        return jsonResult;
    }

    public async Task<string> AddrSearch(string input)
    {

        string[] words = input.Split(' ',
StringSplitOptions.RemoveEmptyEntries);

        string city = null;
        string street = null;
        string houseNumber = null;

        PostgresDb pgsql = new
PostgresDb(connectionString);

        foreach (string word in words)
        {
            if (city == null)
            {
                city = await FindWordInColumn(pgsql, word,
\"\"addr:city\"");
            }

            if (street == null)
            {
                street = await FindWordInColumn(pgsql,
word, \"\"addr:street\"");
            }

            if (houseNumber == null)
            {

```

```

        houseNumber = await FindWordInColumn(pgsql,
word, "\"addr:housenumber\"");
    }
}

if (city != null && street != null && houseNumber
!= null)
{
    return await SeqAddrSearch(city, street,
houseNumber);
}

return "{}";
}

private async Task<string> FindWordInColumn(PostgresDb
pgsql, string word, string columnName)
{
    string query = $"SELECT DISTINCT {columnName} FROM
public.planet_osm_point WHERE {columnName} ILIKE '{word}';";
    DataTable resultTable = await
pgsql.ExecuteQuery(query);

    if (resultTable.Rows.Count > 0)
    {
        return
resultTable.Rows[0][columnName].ToString();
    }

    return null;
}
}
}

```

Файл PostgresDB.cs

```

using System;
using System.Collections.Generic;
using System.Data;
using Npgsql;

namespace addrSearchCore
{
    public class PostgresDb
    {
        private readonly string _connectionString;

        public PostgresDb(string connectionString)
        {
            _connectionString = connectionString;
        }
    }
}

```

```

        public void OpenConnection(out NpgsqlConnection
connection)
        {
            connection = new
NpgsqlConnection(_connectionString);
            connection.Open();
        }

        public void CloseConnection(NpgsqlConnection
connection)
        {
            if (connection != null && connection.State ==
ConnectionState.Open)
            {
                connection.Close();
            }
        }

        public int ExecuteNonQuery(string query,
Dictionary<string, object> parameters = null)
        {
            using (var conn = new
NpgsqlConnection(_connectionString))
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand(query,
conn))
                {
                    AddParameters(cmd, parameters);
                    return cmd.ExecuteNonQuery();
                }
            }
        }

        public object ExecuteScalar(string query,
Dictionary<string, object> parameters = null)
        {
            using (var conn = new
NpgsqlConnection(_connectionString))
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand(query,
conn))
                {
                    AddParameters(cmd, parameters);
                    return cmd.ExecuteScalar();
                }
            }
        }

        public DataTable ExecuteQuery(string query,
Dictionary<string, object> parameters = null)

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.18.001 ПЗ

Лист

79

```

        {
            using (var conn = new
NpgsqlConnection(_connectionString))
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand(query,
conn))
                {
                    AddParameters(cmd, parameters);
                    using (var adapter = new
NpgsqlDataAdapter(cmd))
                    {
                        var dataTable = new DataTable();
                        adapter.Fill(dataTable);
                        return dataTable;
                    }
                }
            }
        }

        private void AddParameters(NpgsqlCommand cmd,
Dictionary<string, object> parameters)
        {
            if (parameters != null)
            {
                foreach (var parameter in parameters)
                {
                    cmd.Parameters.AddWithValue(parameter.Key,
parameter.Value);
                }
            }
        }
    }
}

```

Файл Program.cs

```

using System;

namespace addrSearchCore
{
    internal class Program
    {
        static void Main()
        {
            string[] prefixes = { "http://localhost:4444/" };
            Server server = new Server(prefixes);

            server.Start();
            Console.ReadLine();
            server.Stop();
        }
    }
}

```

Файл RequestControllers.cs

```
using System;
using System.IO;
using System.Net;
using System.Reflection;
using System.Text.Json;
using System.Threading.Tasks;

namespace addrSearchCore
{
    public class RequestController
    {
        private readonly OsmSearch _osmSearch;

        public RequestController()
        {
            _osmSearch = new OsmSearch();
        }

        public async Task HandleRequest(HttpListenerRequest
request, HttpListenerResponse response)
        {
            string path = request.Url.LocalPath;
            System.Console.WriteLine("Request from: " +
request.RemoteEndPoint.ToString());

            try
            {
                switch (path)
                {
                    case "/status":
                        await HandleStatusReq(request,
response);

                        break;
                    case "/test":
                        await HandleTestReq(request, response);
                        break;
                    case "/xml":
                        await HandleXmlReq(request, response);
                        break;
                    case "/osm_seq_search":
                        await HandleOsmSeqSearch(request,
response);

                        break;
                    case "/column_search":
                        await HandleColumnSearch(request,
response);

                        break;
                    case "/tags_search":
                        await HandleTagsSearch(request,
response);

                        break;
                }
            }
        }
    }
}
```

```

        case "/osm_addr_search":
            await osm_addr_search(request,
response);

            break;
        default:
            await HandleDefaultReq(request,
response);

            break;
    }
}
catch (Exception ex)
{
    await ResponseView.SendErrorResponse(response,
ex.Message);
}
}

public async Task HandleDefaultReq(HttpListenerRequest
request, HttpListenerResponse response)
{
    string exePath =
Assembly.GetExecutingAssembly().Location;
    string binPath = Path.GetDirectoryName(exePath);

    string projectRoot =
Directory.GetParent(binPath).Parent.Parent.FullName;

    string reqUrl = (request.Url.AbsolutePath == "/" ) ?
"/pages/startPage/start.html" : request.Url.AbsolutePath;
    string filePath = Path.Combine(projectRoot,
reqUrl.TrimStart('/'));

    try
    {
        using (Stream stream = new FileStream(filePath,
FileMode.Open))
        {
            byte[] buffer = new byte[1024 * 16];
            int nbytes;
            string mime;

            response.ContentType =
MimeType.mimeTypeMappings.TryGetValue(Path.GetExtension(filePath),
out mime) ? mime : "application/octet-stream";
            response.ContentLength64 = stream.Length;
            response.AddHeader("Date",
DateTime.Now.ToString("r"));
            response.AddHeader("Last-Modified",
File.GetLastWriteTime(filePath).ToString("r"));

            while ((nbytes = await
stream.ReadAsync(buffer, 0, buffer.Length)) > 0)
            {

```

```

                                await
response.OutputStream.WriteAsync(buffer, 0, nbytes);
                                }
                                }
                                }
                                catch (Exception ex)
                                {
                                    await ResponseView.SendErrorResponse(response,
ex.Message);
                                }

                                response.Close();
                                }

                                public async Task
HandleOsmSeqSearch(HttpListenerRequest request, HttpListenerResponse
response)
                                {
                                    try
                                    {
                                        using (var reader = new
StreamReader(request.InputStream, request.ContentEncoding))
                                        {
                                            var jsonString = await
reader.ReadToEndAsync();
                                            var searchParams =
JsonSerializer.Deserialize<SearchParams>(jsonString);

                                            var jsonResponse = await
_osmSearch.SeqAddrSearch(searchParams.City, searchParams.Street,
searchParams.Housenumber);

                                            await
ResponseView.SendJsonResponse(response, jsonResponse);
                                        }
                                    }
                                    catch (Exception ex)
                                    {
                                        await ResponseView.SendErrorResponse(response,
ex.Message);
                                    }
                                }

                                public async Task HandleAddrSearch(HttpListenerRequest
request, HttpListenerResponse response)
                                {
                                    try
                                    {
                                        using (var reader = new
StreamReader(request.InputStream, request.ContentEncoding))
                                        {
                                            var jsonString = await
reader.ReadToEndAsync();

```

```

        var searchParams =
JsonSerializer.Deserialize<SearchParams>(jsonString);

        var jsonResponse = await
_osmSearch.AddrSearch(searchParams.Input);

        await
ResponseView.SendJsonResponse(response, jsonResponse);
    }
}
catch (Exception ex)
{
    await ResponseView.SendErrorResponse(response,
ex.Message);
}
}

public async Task HandleStatusReq(HttpListenerRequest
request, HttpListenerResponse response)
{
    string message = "Server is running";
    await ResponseView.SendTextResponse(response,
message);
}
}
}

```

Файл ResponseView.cs

```

using System.IO;
using System.Net;
using System.Text;
using System.Threading.Tasks;

namespace addrSearchCore
{
    public static class ResponseView
    {
        public static async Task
SendJsonResponse(HttpListenerResponse response, string jsonString)
        {
            byte[] buffer = Encoding.UTF8.GetBytes(jsonString);
            response.ContentLength64 = buffer.Length;
            response.ContentType = "application/json";
            using (Stream stream = response.OutputStream)
            {
                await stream.WriteAsync(buffer, 0,
buffer.Length);
            }
        }
        public static async Task
SendTextResponse(HttpListenerResponse response, string message)
        {

```

```

        byte[] buffer = Encoding.UTF8.GetBytes(message);
        response.ContentLength64 = buffer.Length;
        response.ContentType = "text/plain";
        using (Stream stream = response.OutputStream)
        {
            await stream.WriteAsync(buffer, 0,
buffer.Length);
        }
    }

    public static async Task
SendErrorResponse(HttpListenerResponse response, string
errorMessage)
    {
        response.StatusCode =
(int)HttpStatusCode.InternalServerError;
        await SendTextResponse(response, errorMessage);
    }
}

```

Файл Server.cs

```

using System;
using System.Net;
using System.Threading.Tasks;

namespace addrSearchCore
{
    internal class Server
    {
        private HttpListener _listener;

        public Server(string[] prefixes)
        {
            _listener = new HttpListener();
            foreach (var prefix in prefixes)
            {
                _listener.Prefixes.Add(prefix);
            }
        }

        public void Start()
        {
            try
            {
                _listener.Start();
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex);
            }
            Console.WriteLine("Server started");
            OsmSearch osm = new OsmSearch();
        }
    }
}

```

```

        osm.addrSearch();

        Task.Run(async () => await Listen());
    }

    public void Stop()
    {
        _listener.Stop();
        _listener.Close();
        Console.WriteLine("Server stopped");
    }

    private async Task Listen()
    {
        while (_listener.IsListening)
        {
            try
            {
                var context = await
                _listener.GetContextAsync();
                Task.Run(() => ProcessReq(context));
            }
            catch (HttpListenerException) { }
        }
    }

    private async Task ProcessReq(HttpListenerContext
context)
    {
        var requestController = new RequestController();
        await
requestController.HandleRequest(context.Request, context.Response);
    }
}

```

Приложение В

(Обязательное)

Сервис адресного географического поиска

Презентация

					ПГУ 1.090301.18.001 ПЗ	Лист
						87
Изм.	Лист	№ докум.	Подпись	Дата		

Сервис адресного географического поиска

Выполнил студент группы 20ВВП1
Новиков И.М.
Научный руководитель
к.т.н. доцент кафедры ВТ Слепцов Н.В.

Пенза, 2024

1

Цель работы

Разработать сервис адресного географического поиска

Объект работы

Сервис, дающий возможность:

- Выполнять адресный поиск
- Выполнять поиск по OSM тегам
- Выполнять раздельный поиск
- Визуализировать геоинформационные данные на актуальной карте мира

2

Актуальность темы

Основная проблема подобных сервисов – неполноценный доступ к используемому инструменту, что сильно ограничивает пользователей и разработчиков, желающих внедрить адресный поиск в свои проекты.

Это и подчеркивает необходимость разработки нового, открытого и бесплатного сервиса для адресного географического поиска.

3

Аналоги



Nominatim

4

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.18.001 ПЗ

Лист

89

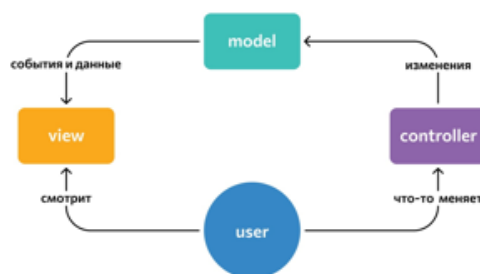
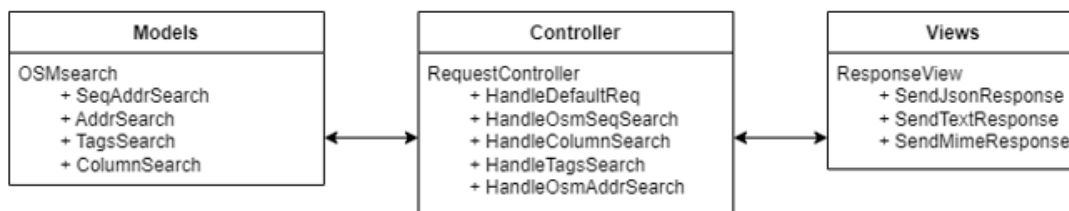
Стек технологий



5

Архитектура приложения

MVC



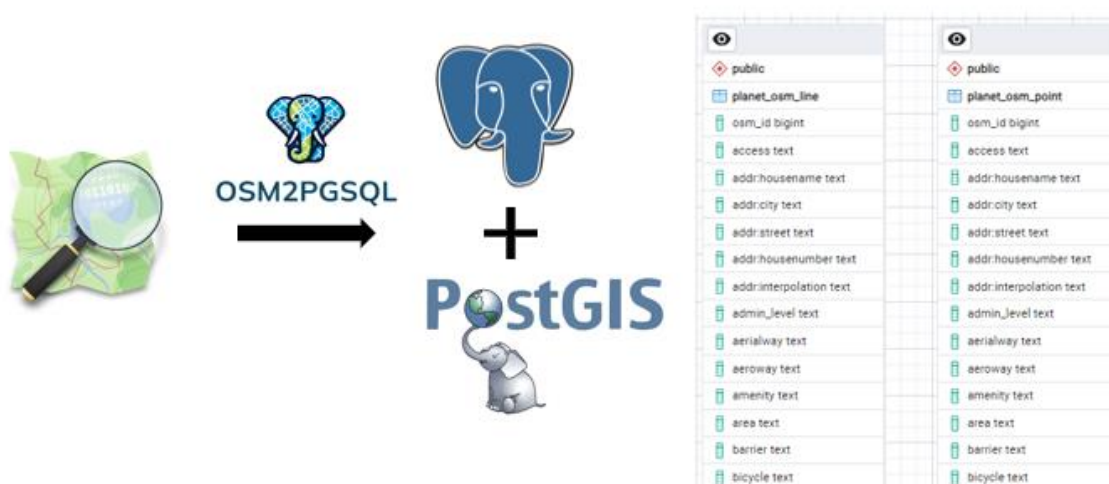
6

Варианты использования



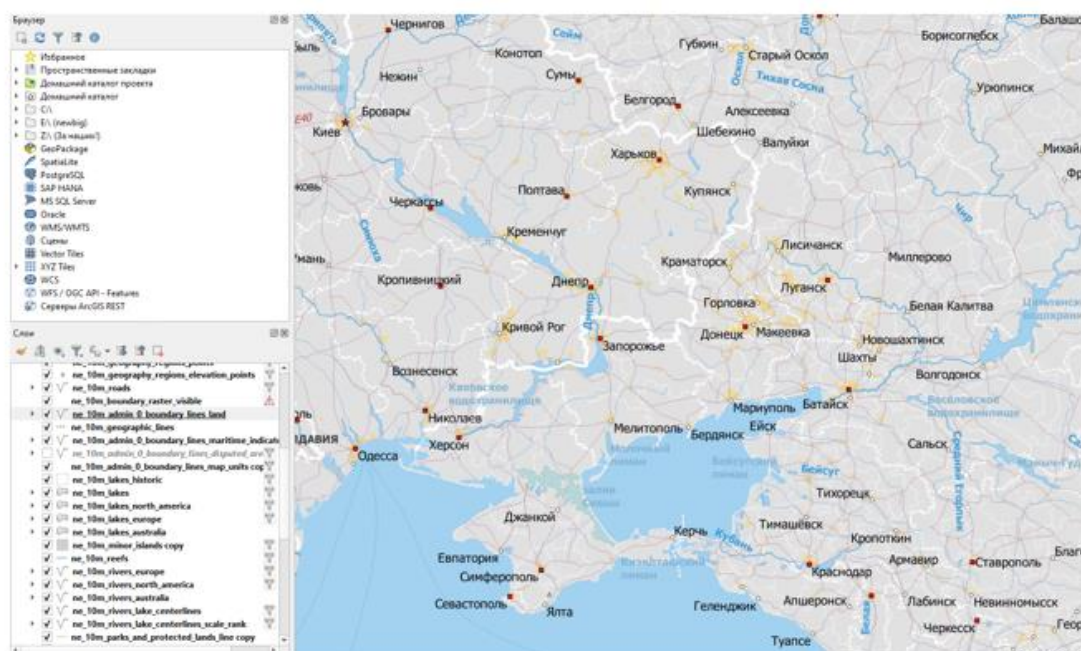
7

База данных



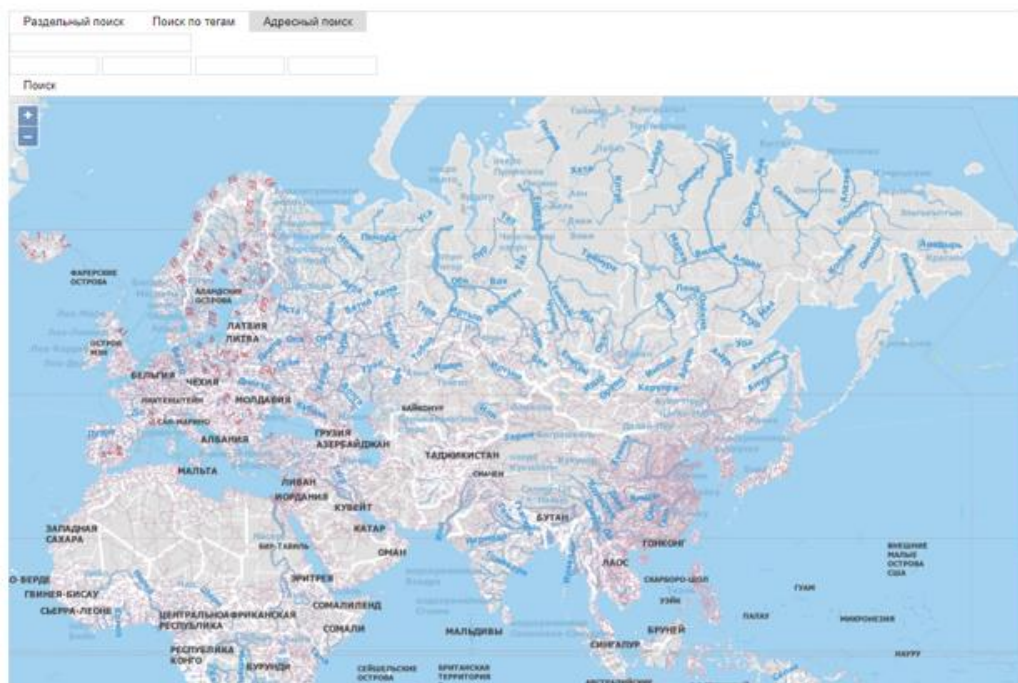
8

Составление карты



9

Общий вид карты, отображаемым фронтом сервис



10

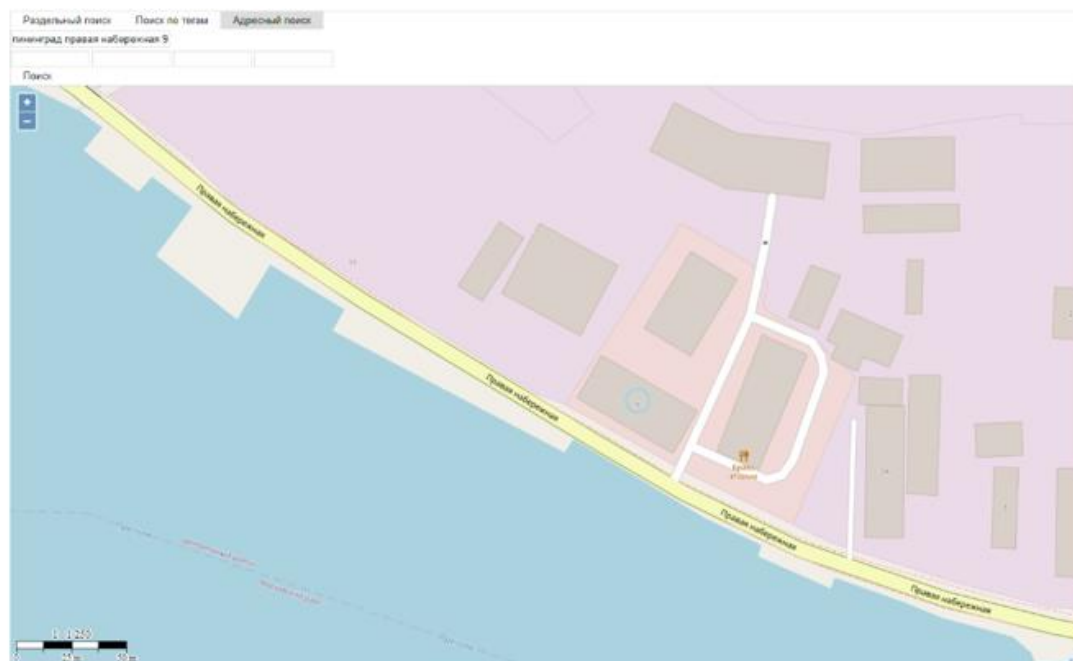
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.18.001 ПЗ

Лист

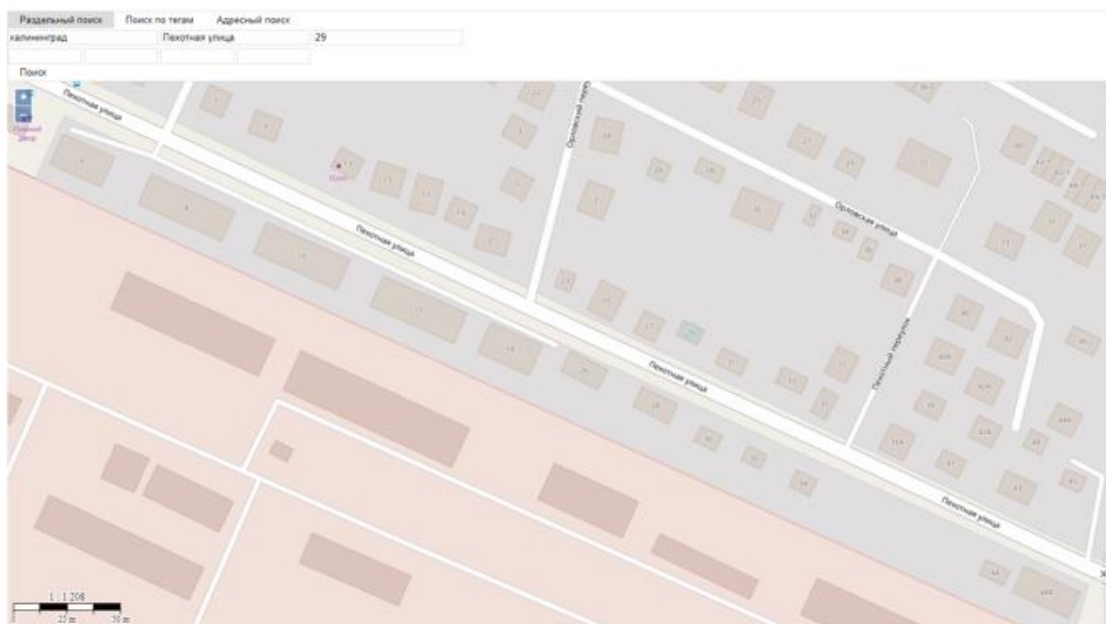
92

Адресный поиск



11

Раздельный поиск



12

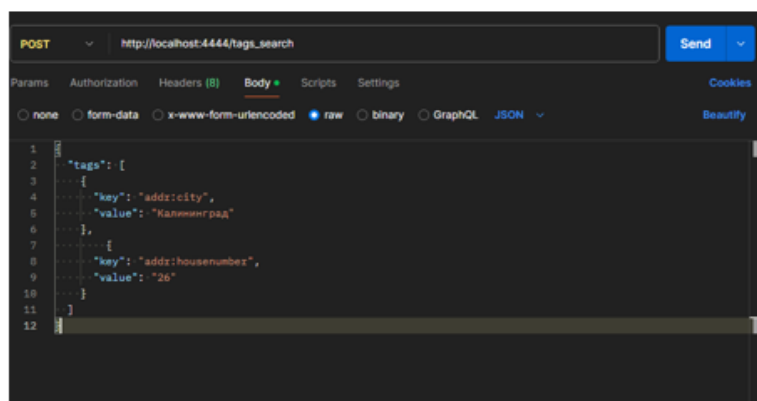
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.18.001 ПЗ

Лист

93

Поиск по тегам



13

Результат работы

Результатом работы является сервис адресного географического поиска, который позволяет

Сервис, дающий возможность:

- Выполнять поиск местоположения географических объектов
- Выполнять поиск объектов по OSM тегам
- Выполнять отдельный адресный поиск

14

Спасибо за внимание!

15

					ПГУ 1.090301.18.001 ПЗ	Лист
						95
Изм.	Лист	№ докум.	Подпись	Дата		