МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

КУРСОВОЙ ПРОЭКТ

по дисциплине «Разработка программного обеспечения информационных систем»

Тема: Эффективный поиск достопримечательностей по геокоординатам

Студент гр. 5303 _____ Ветров А.А.

Студент гр. 5303 _____ Круглик А.Д.

Студентка гр. 5303 _____ Смурова А.И.

Преподаватель _____ Заславский М.М.

Санкт-Петербург
2018

ЗАДАНИЕ
НА КУРСОВОЙ ПРОЕКТ

Студенты Ветров А.А., Круглик А.Д., Смурова А.И.

Группа 5303

Тема проекта: Эффективный поиск достопримечательностей по геокоординатам

Исходные данные:
Проект должен быть разработан с использованием базы данных Memcached

Содержание пояснительной записки:
Содержание, Введение, Качественные требования к решению, Сценарии использования, Модель данных, Разработанное прилоежение, Заключение, Список использованных источников.

Предполагаемый объем пояснительной записки:
Не менее 15 страниц.

Дата выдачи задания: 13.09.2018
Дата сдачи реферата: 20.12.2018
Дата защиты реферата: 20.12.2018

Студент	_____	Ветров А.А.
Студент	_____	Круглик А.Д.
Студентка	_____	Смурова А.И.
Преподаватель	_____	Заславский М.М.

АННОТАЦИЯ

В курсовом проекте реализован веб-сервис на основе СУБД Memcached, с помощью которого можно определить ближайшие достопримечательности для заданного маршрута.

SUMMARY

Web-service on the basis of Memcached DBMS by which it is possible to define the nearby landmarks for the set route is implemented in the course project.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время в Санкт-Петербурге насчитывается большое количество достопримечательностей. Даже жители города не всегда знают о том, что каждый день они проходят мимо какой-то достопримечательности и не замечают её.

Целью проекта является разработка приложения, с помощью которого можно определить ближайшие достопримечательности для заданного места или маршрута и получения краткой информации о ней.

В проекте разработано веб-приложение на основе СУБД Memcached.

1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Необходимо разработать веб-приложение, позволяющее узнать местонахождение ближайших достопримечательностей.

Основные функции:

- Поиск и просмотр информации про достопримечательность по названию;
- Поиск достопримечательностей около заданной точки;
- Поиск ближайших достопримечательностей по заданному маршруту;
- Импорт и экспорт данных.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. Сценарии использования для задачи импорта, представления, анализа и экспорта данных

Для импорта данных Пользователь должен нажать на кнопку «Добавить данные», выбрать файл формата .csv. В файле каждая строка – это новый объект, для объекта хранятся данные в следующем виде: number,name,name_object,date,author,address,district,protection_category,base,note.

Данные отображаются при поиске достопримечательности по названию и при поиске ближайших достопримечательностей.

Поиск достопримечательнойстей по названию

Основной поток:

1. Пользователь вводит название в поле «Введите название»
2. Пользователь нажимает кнопку «Найти».
3. После нажатия кнопки открывается страница с краткой информацией о найденном объекте. Пользователь видит название достопримечательности, наименование ансамбля, значимые даты для объекта, авторов объекта, а также карту, на которой отмечен данный объект.

Альтернативный поток:

1. Если в базе данных не оказалось введённой достопримечательности, то Пользователь видит сообщение о том, что такой достопримечательности не найдено.

Поиск ближайших достопримечательностей

Основной поток:

1. Пользователь может ввести одну или несколько точек (Альтернативный поток), рядом с которыми хочет найти достопримечательности, или использовать своё местоположение, чтобы найти достопримечательности рядом с собой.
2. После нажатия кнопки «Найти» (в первом случае) или «Найти по местоположению» (во втором случае) открывается страница со списком, в котором указаны найденные объекты, и карта с этими объектами и введенными точками для поиска.

3. При выборе объекта из списка открывается страница с достопримечательностью, на которой Пользователь видит название достопримечательности, наименование ансамбля, значимые даты для объекта, авторов объекта, самую близкую и самую дальнюю достопримечательность, а также карту, на которой отмечен данный объект.
4. Пользователь может нажать кнопку «Назад к списку», чтобы вернутся обратно к списку всех достопримечательностей, найденных по запросу.

Альтернативный поток:

1. Пользователь вводит адрес в поле «Введите адрес», нажимает Enter и данная точка отражается под полем. Если Пользователь хочет, то он может удалить какую-то введенную точку.
2. После ввода всех точек Пользователь нажимает на кнопку «Найти» (Основной поток, п.2)

Для экспорта данных необходимо нажать на кнопку «Получить данные». При этом скачается файл формата .csv.

2.2. Вывод

Для решения преобладают операции чтения, так как для Пользователя реализован поиск достопримечательностей, то есть поиск и вывод данных, а не их добавление.

3. МОДЕЛЬ ДАННЫХ

3.1. Описание структуры

Информация о каждой достопримечательности хранится по отдельному ключу.

Ключ - координаты достопримечательности.

Значение java object - словарь с ключами:

- "Наименование ансамбля" : String;
- "Наименование объекта" : String;
- "Датировка" : String;
- "Авторы" : String;
- "Адрес" : String;
- "Район города" : String;
- "Основание" : String;
- "Примечания" : String.

3.2. Нереляционная модель данных

Расчет памяти для memcached:

(Ключ типа String (24 байта) + значение(java object все поля которого - строки) String(~24 байта) * numOfCol(8)) * numOfObjects(~8000).

Итого ~ 1 728 000 байт

Формула:

("Наименование ансамбля" + "Наименование объекта" + "Датировка" + "Авторы" + "Адрес" + "Район города" + "Основание" + "Примечания") * N = 216 * N

Примечание: при необходимости оптимизации memcached для определенных запросов количество требуемой памяти будет увеличиваться.

3.3. Аналог модели данных для SQL СУБД

На рисунке 1 представлена модель данных для реляционной базы данных.

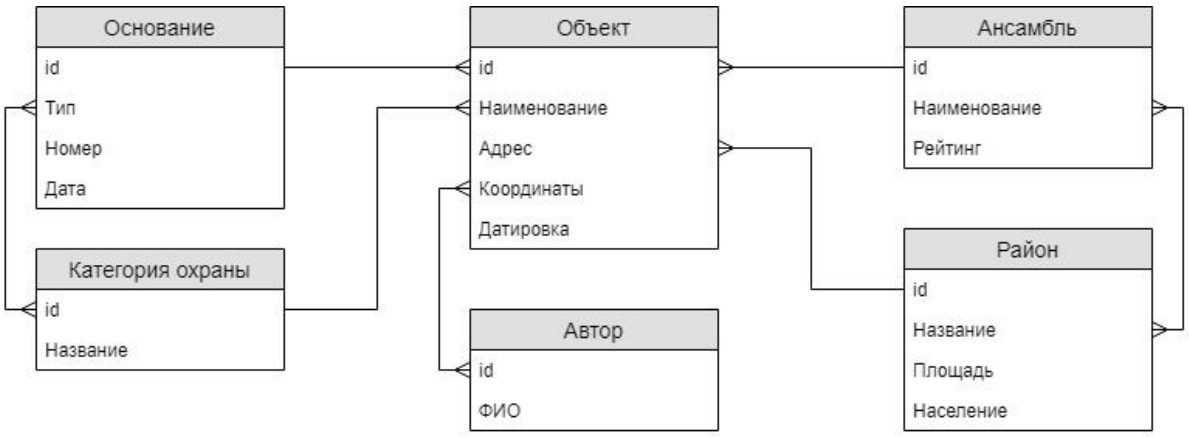

Рисунок 1 – Модель данных

Используемые атрибуты:

- "Наименование ансамбля" : String;
- "Наименование объекта" : String;
- "Название категории охраны" : String;
- "Датировка" : String;
- "Дата основания" : Date;
- "Авторы" : String;
- "Адрес" : String;
- "ФИО" : String;
- "Район города" : String;
- "Основание" : String;
- "Примечания" : String;
- "Широта" : Float;
- "Долгота" : Float;
- "id" : Integer.

Расчет для sql database:

(String + id (Int)) * numOfAuth + (String * ObjNumOfFields + id (Int)) * numOfObj + (String * AnsNumOfFields + id (Int)) * numOfAns + (String * DistNumOfFields + id (Int)) * numOfDist + (String * BasisNumOfFields + id (Int)) * numOfBasis (String * SecCatNumOfFields + id (Int)) * numOfSecCat ~ (24 + 4) * 6000 + (24 * 4 + 4) * 8000 + (24 * 1 + 4) * 500 + (24 * 3 + 4) * 18 + (24 * 3 + 4) * 8000 + (24 + 4) * 100 ~ 168 000 + 800 000 + 14 000 + 1 368 + 608 000 + 2 800 ~ 1 600 000

Итого ~ 1 600 000 (байт)

Так как рассматривали ER диаграмму без учета дополнительной памяти для связи таблиц, поэтому можно утверждать, что в этом случае объемы затрачиваемой памяти будут приблизительно одинаковыми.

Формула

("ФИО" + id) * M + ("Наименование" + "Адрес" + "Высота" + "Ширина" + "Долгота" + "Датировка" + id) * P + ("Наименование ансамбля" + id) * Q + ("Название района" + id) * T + ("Тип" + "Номер" + "Дата основания" + id) * S + ("Название категории охраны" + id) * W = ("ФИО" + id) * 2N/3 + ("Наименование" + "Адрес" + "Высота" + "Ширина" + "Долгота" + "Датировка" + id) * N + ("Наименование ансамбля" + id) * N/16 + ("Название района" + id) * N/500 + ("Тип" + "Номер" + "Дата основания" + id) * N + ("Название категории охраны" + id) * N/80 = (("ФИО" + id) * 2/3 + ("Наименование" + "Адрес" + "Высота" + "Ширина" + "Долгота" + "Датировка" + id) + ("Наименование ансамбля" + id) / 16 + ("Название района" + id) / 500 + ("Тип" + "Номер" + "Дата основания" + id) + ("Название категории охраны" + id) / 80) * N = 245 * N

3.4. Запросы

Запросы SELECT

Select по координатам в memcached за O(1), а в реляционных БД за O(n). Для других select-запросов, когда в реляционных БД придется выполнять множественные join`ы, что является крайне дорогой операцией, memcached можно дополнить таким образом, чтобы запросы выполнялись за константное время (в худшем случае за линейное).

Примеры запросов представлены в таблице 1.

Таблица 1 – Примеры запросов

Формулировка запроса	memcached	SQL
Вся информация по названию	get "Координата"\r	SELECT * FROM "Объект" JOIN "Автор" JOIN "Ансамбль" JOIN "Район" JOIN "Основание" JOIN "Категория охраны" WHERE "Объект"."Наименование" = "Наименование"
	Придется выполнить N запросов	Придется выполнить один запрос, но, стоит помнить, что JOIN - крайне дорогая операция
Вся информация по координатам	get "Координата"\r	SELECT * FROM "Объект" JOIN "Автор" JOIN "Ансамбль" JOIN "Район" JOIN "Основание" JOIN "Категория охраны" WHERE "Объект"."Широта" = "Широта" AND "Объект"."Долгота" = "Долгота"
	Придется выполнить один запрос	Придется выполнить один запрос, но, стоит помнить, что JOIN - крайне дорогая операция
Вся информация для всех достопримечательностей	stats cachedump n m\r	SELECT * FROM "Объект" JOIN "Автор" JOIN "Ансамбль" JOIN "Район" JOIN "Основание" JOIN "Категория охраны"
	Придется выполнить количество запросов равное количеству slab	Придется выполнить один запрос, но, стоит помнить, что JOIN - крайне дорогая операция
Ближайшую к координате	Невозможно	SELECT * FROM "Объект" JOIN "Автор" JOIN "Ансамбль" JOIN "Район" JOIN "Основание" JOIN "Категория охраны" WHERE ACOS(SIN("Объект"."Широта") * SIN("Широта") + COS("Объект"."Широта")*COS("Широта")*COS("Объект"."Долгота" - "Долгота")) ORDER BY ASCLIMIT 1
		Придется выполнить один запрос, но, стоит помнить, что JOIN - крайне дорогая операция

Запросы CREATE

Асимптотически одинаково O(1).

Запросы UPDATE

Ситуация аналогична select-запросам.

Запросы DELETE

Ситуация аналогична update-запросам, за исключением того, что в реляционных БД каскадное удаление, что является дорогой операцией, но позволяет сохранить целостность данных.

3.5. Выводы

Для рассматриваемой задачи NoSQL модель данных предпочтительнее SQL модели в плане сложности запросов к БД и скорости их выполнения, причем затрачиваемое количество памяти будет примерно одинаково, но, стоит также отметить, что некоторые запросы в memcached выполнить невозможно и придется производить обработку данных вручную.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Краткое описание

Разработанное приложение осуществляет эффективный поиск достопримечательностей по геоданным по названию и по геокоординатам. Приложение состоит из страниц:

1. Главная страница. На данной странице Пользователь может задать данные для поиска.
2. Страница с отображением найденных достопримечательностей. Пользователь на карте видит отмеченные достопримечательности. Может выбрать одну из них и узнать более подробную информацию о ней.
3. Страница с отображением достопримечательности. Пользователь видит карту, в центре которой достопримечательность и её название. Также на данной странице отображается самая ближняя и самая дальная достопримечательность относительно неё.
4. Страница с поиском достопримечательностей по местоположению Пользователя. Пользователь должен подтвердить найденное местоположение, а затем запустить поиск достопримечательностей рядом.

4.2. Использованные технологии

При написании приложения использовались следующие технологии:

- Java 8;
- Spring Core;
- Spring Boot;
- Memcached;
- Maven;
- Google Maps API;
- Open Map API;
- Lombok;
- Thymeleaf;
- HTML;
- CSS;
- Java Script.

4.3. Ссылки на Приложение

Исходный код приложения и инструкция по установке находятся по ссылке:

https://github.com/moevm/nosql2018-memcached_routing

ЗАКЛЮЧЕНИЕ

Разработано приложение для эффективного поиска достопримечательностей Санкт-Петербурга с использованием базы данных Memcached. Приложение работает корректно, но имеется узкое место: получение координат по адресу с использованием открытого бесплатного Open Map API. Решить данную проблему можно, преобретя платный ключ разработчика для Google Maps API.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Java Platform Standard Edition 8 Documentation URL: https://docs.oracle.com/javase/8/docs/ (дата обращения: 19.12.2018).
2. Core Technologies // Spring Framework Documentation URL: https://docs.spring.io/spring/docs/current/spring-framework-reference/core.html#spring-core (дата обращения: 19.12.2018).
3. Spring Boot Reference Guide URL: https://docs.spring.io/spring-boot/docs/current/reference/html/ (дата обращения: 19.12.2018).

ПРИЛОЖЕНИЕ А. ДОКУМЕНТАЦИЯ ПО СБОРКЕ И РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ

Инструкция по сборке и запуску:

1. Скачать проект из репозитория.
2. Собрать из него файл с расширением .jar. Это можно сделать с помощью технологии Maven или средствами IDE.
3. Скачать базу данных Memcached.
4. Запустить базу данных Memcached.
5. Запустить jar-файл.
6. Перейти в браузере по адресу: http://localhost:8080.

ПРИЛОЖЕНИЕ В. ИНСТРУКЦИЯ ДЛЯ ПОЛЬЗОВАТЕЛЯ

Главный экран

При входе на сайт отображается главная страница, на которой пользователь должен добавить данные для поиска, нажав на кнопку «Добавить данные» и выбрав файл в открывшемся окне.

Для поиска Пользователь должен ввести или название достопримечательности, или геокоординаты для поиска и нажать кнопку «Поиск», также Пользователь может нажать на кнопку «Поиск по местоположению».

Страница с отображением найденных достопримечательностей

При поиске по геокоординатам Пользователь попадает на страницу с картой, на которой отображены, найденные достопримечательности.

Пользователь может выбрать один из маркеров и, нажав на кнопку «Узнать информацию», открыть страницу с отображением одной достопримечательности.

Страница с отображением достопримечательности

При поиске по названию достопримечательности или при выборе конкретной достопримечательности из нескольких Пользователь попадает на страницу, на которой отображено название достопримечательности, самая близкая и самая дальняя достопримечательность, а также карта, которая отцентрированная по геоданым достопримечательности.

Страница с поиском достопримечательностей по местоположению

Пользователь должен подтвердить найденное местоположение с помощью кнопки «Найти достопримечательности», а затем запустить поиск достопримечательностей.

Навигация по страницам приложения осуществляется с помощью кнопок меню в шапке страницы. Также Пользователь может получить загруженные данные с помощью кнопки «Получить данные».

ПРИЛОЖЕНИЕ С. СНИМКИ ЭКРАНА ПРИЛОЖЕНИЯ

На рисунках 2-5 изображены снимки экрана приложения.

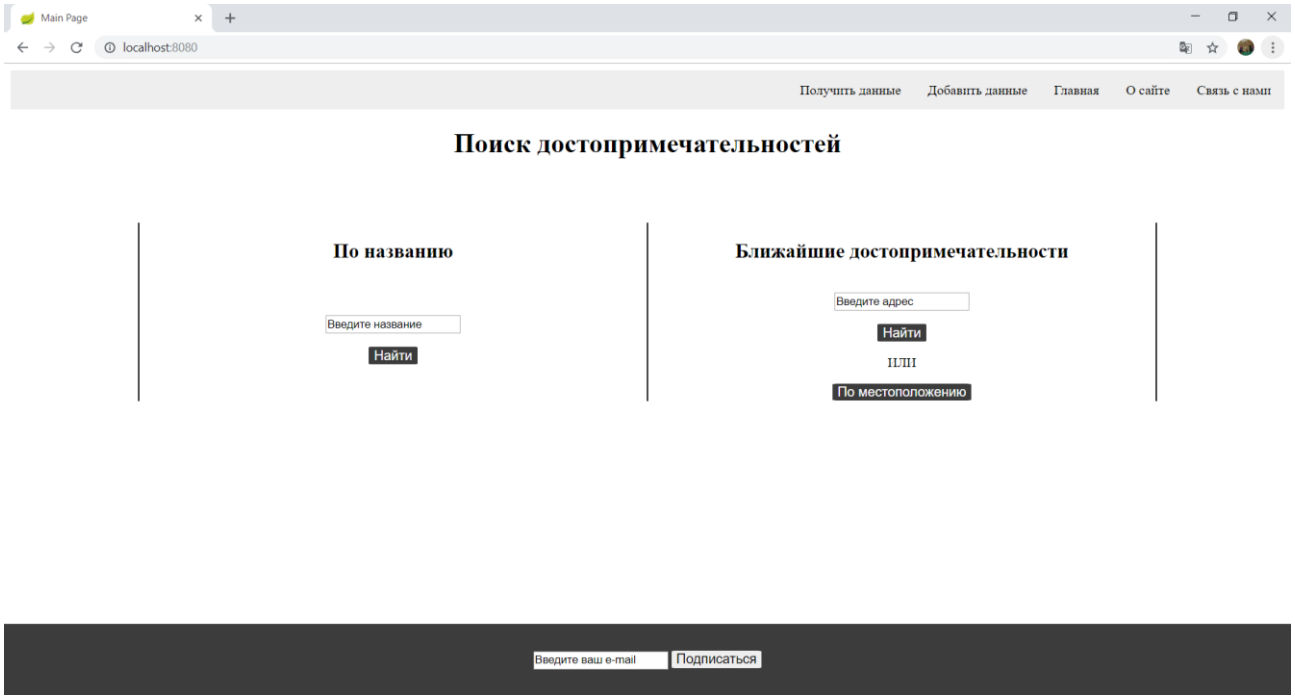

Рисунок 2 – Главная страница

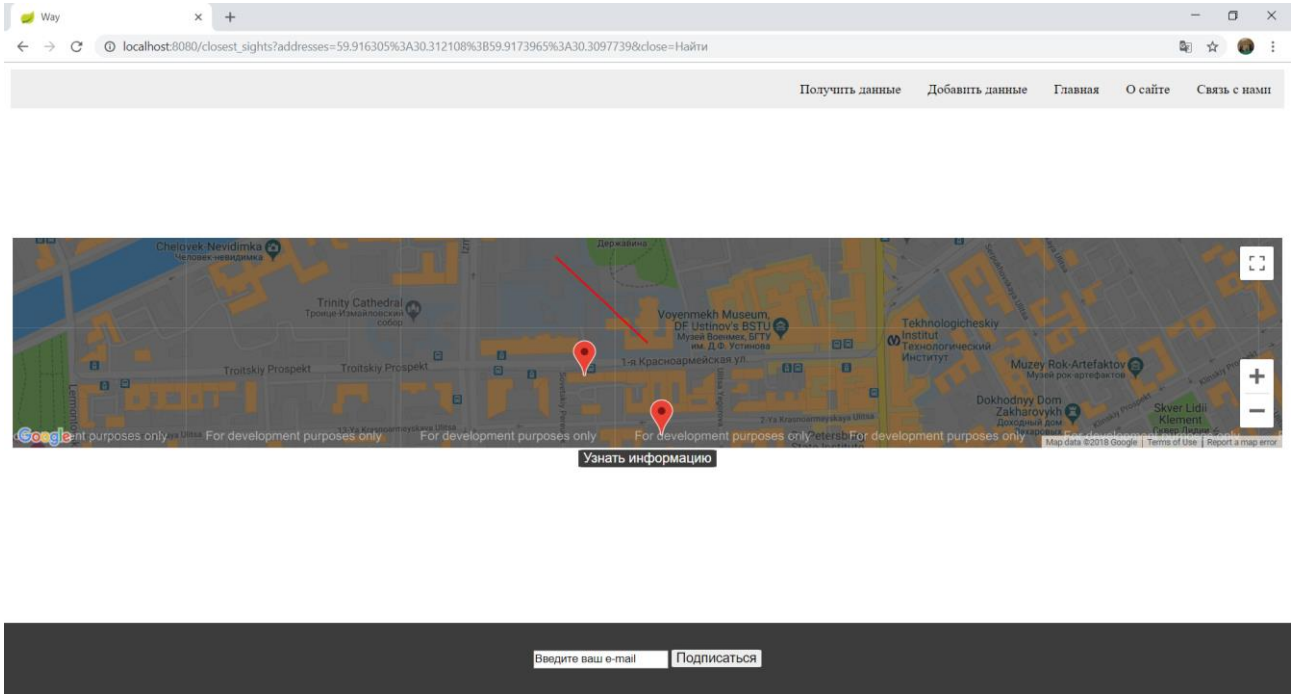

Рисунок 3 – Страница с отображением найденных достопримечательностей

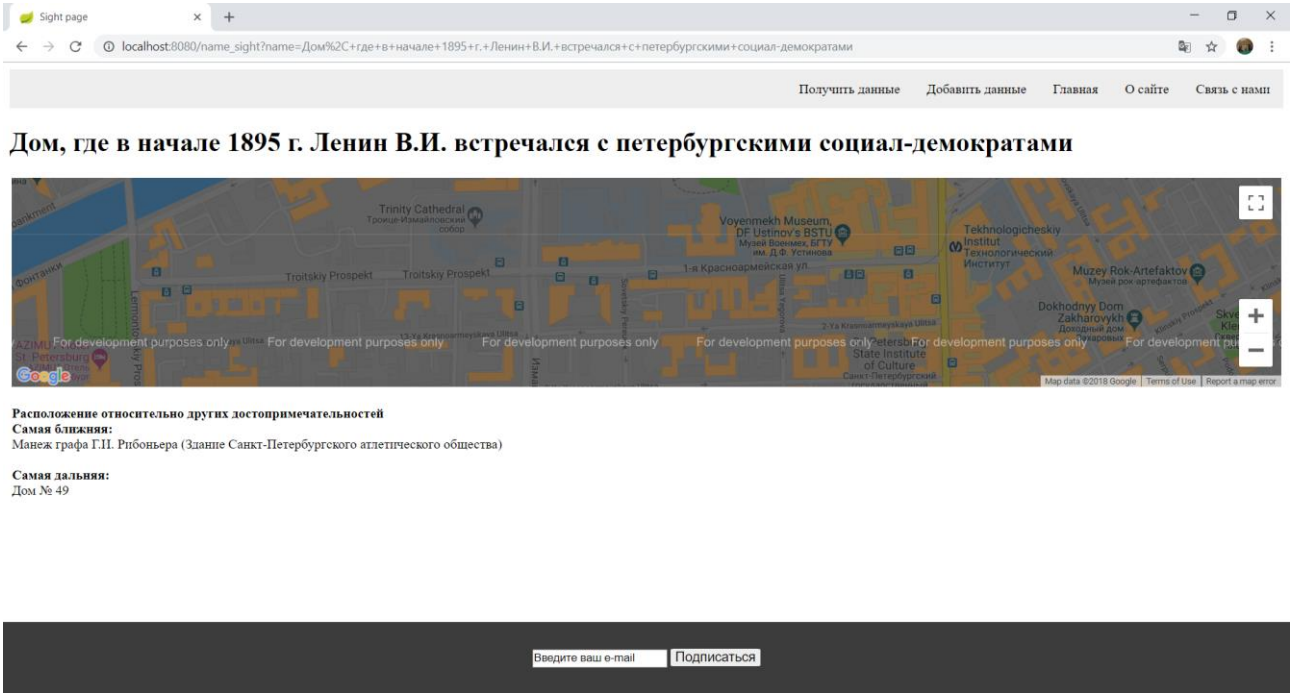

Рисунок 4 – Страница с отображением достопримечательности

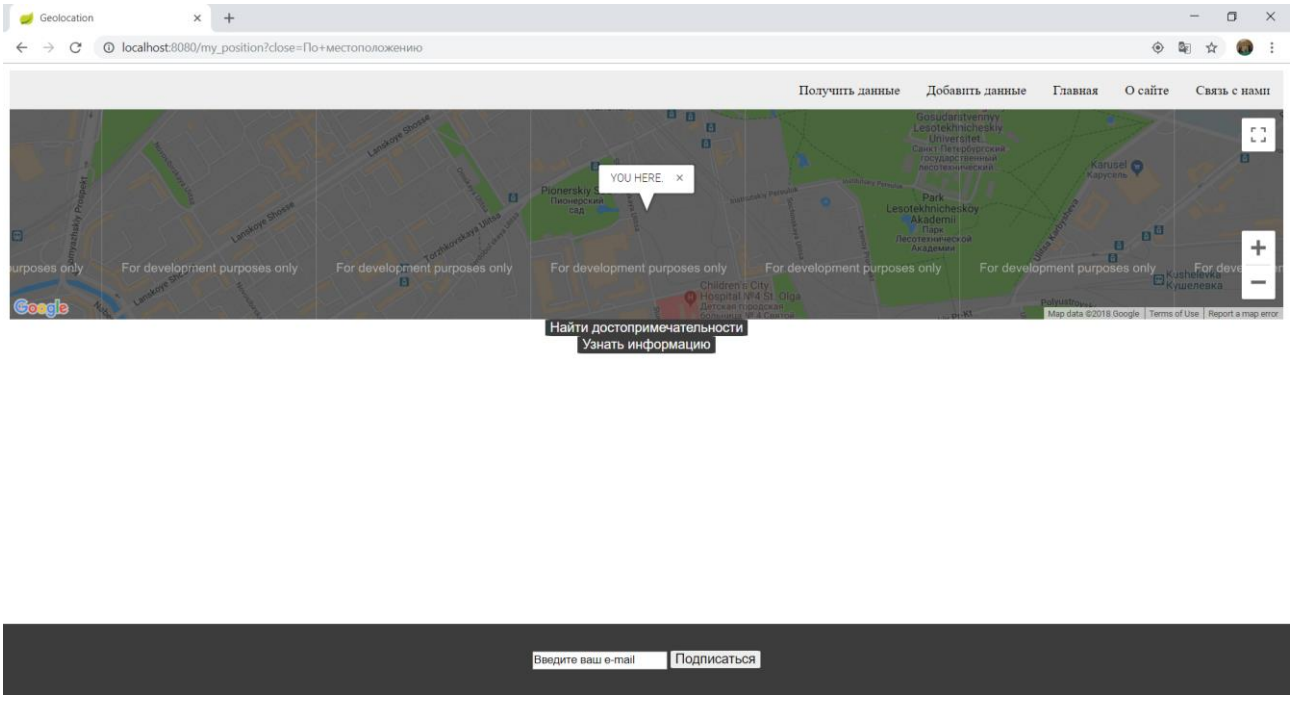

Рисунок 5 – Страница с поиском достопримечательностей по местоположению