**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра MO ЭВМ**

индиВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

**по дисциплине «Разработка программного обеспечения информационных систем»**

Тема: Построение маршрутов - Mongo

|  |  |  |
| --- | --- | --- |
| Студентка гр. 5303 |  | Клименко А.Ю. |
| Студентка гр. 5303 |  | Федорова М.Д. |
| Студент гр. 5303 |  | Шабашов Н.А. |
| Преподаватель |  | Заславский М.М. |

Санкт-Петербург

2018

**ЗАДАНИЕ**

**на индивидуальное домашнее задание**

|  |  |  |
| --- | --- | --- |
| Студенты Клименко А.Ю., Федорова М.Д., Шабашов Н.А. | | |
| Группа 5303 | | |
| Тема проекта: Построение маршрутов - Mongo | | |
| Исходные данные:  Проект должен быть разработан с использованием базы данных Mongo | | |
| Содержание пояснительной записки:  Содержание, Введение, Качественные требования к решению, Сценарии использования, Модель данных, Разработанное приложение, Заключение, Список использованных источников. | | |
| Предполагаемый объем пояснительной записки:  Не менее 15 страниц. | | |
| Дата выдачи задания: 13.09.2018 | | |
| Дата сдачи реферата: 27.12.2018 | | |
| Дата защиты реферата: 27.12.2018 | | |
| Студентка гр.5303 |  | Клименко А.Ю. |
| Студентка гр.5303 |  | Федорова М.Д. |
| Студент гр.5303 |  | Шабашов Н.А. |
| Преподаватель |  | Заславский М.М. |

**Аннотация**

В ходе выполнения данного индивидуального домашнего задания был реализован веб-сервис на основе СУБД MongoDB, с помощью которого выполняются следующие функции:

* построение маршрута из одной точки на карте в другую;
* просмотр информации о заданной точке на карте;
* экспорт данных.

**Summary**

In the course of this individual homework, a MongoDB-based web service was implemented, with which the following functions are performed:

• building a route from one point on the map to another;

• view information about a given point on the map;

• export data.**содержание**

Оглавление

[Введение 5](#_Toc533681664)

[1. Качественные требования к решению 6](#_Toc533681665)

[2. Сценарии использования 7](#_Toc533681666)

[2.1. Сценарии использования для задачи импорта, представления, анализа и экспорта данных 7](#_Toc533681667)

[2.2. Вывод 8](#_Toc533681668)

[3. Модель данных 9](#_Toc533681669)

[3.1. Описание структуры 9](#_Toc533681670)

[3.2. Нереляционная модель данных 10](#_Toc533681671)

[3.3. Аналог модели данных для SQL СУБД 11](#_Toc533681672)

[3.4. Запросы 13](#_Toc533681673)

[3.5. Выводы 14](#_Toc533681674)

[4. Разработанное приложение 15](#_Toc533681675)

[4.1. Краткое описание 15](#_Toc533681676)

[4.2. Использованные технологии 15](#_Toc533681677)

[4.3. Ссылки на Приложение 16](#_Toc533681678)

[Список использованных источников 18](#_Toc533681679)

[Приложение А. Документация по сборке и развертыванию приложения 19](#_Toc533681680)

[Приложение B. Инструкция для пользователя 20](#_Toc533681681)

[Приложение С. Снимки экрана приложения 21](#_Toc533681682)

# Введение

На данный момент в России насчитывается более 40,9 млн легковых автомобилей. В среднем на каждую тысячу жителей приходится 284 автомобиля. Это означает, что потребность просмотра маршрута из одной точки на карте в другую возрастает.

Целью проекта является разработка приложения, с помощью которого можно построить маршрут из одной точки на карте в другую.

В проекте разработано веб-приложение на основе СУБД MongoDB.

# 1. Качественные требования к решению

Необходимо разработать веб-приложение, позволяющее построить маршрут из одной точки на карте в другую.

Основные функции:

* Поиск и просмотр информации по названию/адресу;
* Информация по координате;
* Построение маршрутов;
* Экспорт данных.

# 2. Сценарии использования

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

Для экспорта данных необходимо отправить rest-запрос к одной из коллекций. Примеры запросов представлены ниже:

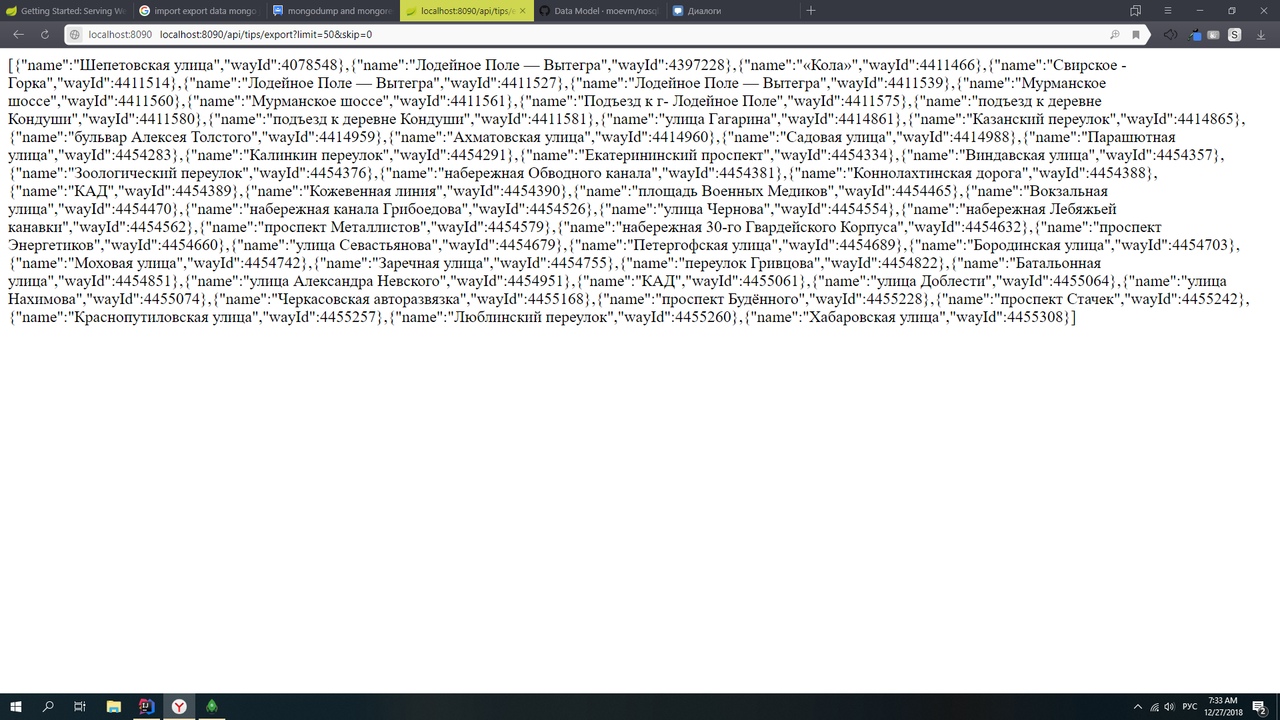
http://localhost:8090/api/way/export?limit=50&skip=0  
http://localhost:8090/api/tips/export?limit=50&skip=0  
http://localhost:8090/api/route/node/export?limit=50&skip=0  
http://localhost:8090/api/route/export?limit=50&skip=0  
<http://localhost:8090/api/node/export?limit=50&skip=0>

Здесь:

way – название коллекции;

export – операция;

limit=50 – ограничение на вывод.



1. «Построение маршрутов»
   1. Пользователь в поле поиска вводит необходимый адрес/название места и нажимает на кнопку «Найти»;
   2. Система отмечает на карте найденное место, показывая о нем краткую информацию.
   3. Пользователь нажимает на кнопку «Проложить маршрут»;
   4. Система выводит окно «Откуда – куда», где пользователь может ввести место отправления, а место назначения – это уже отмеченное место на карте
   5. Пользователь вводит адрес отправления и нажимает кнопку «Построить»
   6. Система отмечает адрес отправления, строит маршрут и выводит его на карту, отображая информацию о расстоянии пути и времени его прохождения.
2. «Информация по координате»
3. Пользователь в поле поиска вводит необходимые координаты и нажимает на кнопку «Найти» или отмечает искомое место на карте, кликая на него;
4. Система отмечает на карте найденное место, показывая о нем краткую информацию (координаты).
5. «Поиск по названию/адресу»
6. Пользователь в поле поиска вводит необходимый адрес/название места и нажимает на кнопку «Найти»;
7. Система отмечает на карте найденное место, показывая о нем краткую информацию.

## 2.2. Вывод

Для решения преобладают операции чтения, так как для Пользователя реализован поиск объектов на карте по адресу/координате и отображение маршрутов.

# 3. Модель данных

Модель данных для нереляционной базы данных.

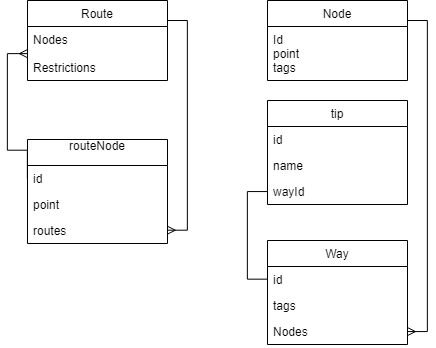


Рисунок 1 – Модель данных нереляционной БД

## 3.1. Описание структуры

**Коллекция Node:**

Id – идентификатор точки (int64);

tags – поля точки (массив пар string).

**Коллекция Route:**

Id – идентификатор маршрута (int64)

Restrictions – ограничения на маршруте (массив int)

Tags – поля маршрута (массив пар string)

Nodes – массив точек

**Коллекция routeNodes:**

id – идентификатор (int64)

point – координаты точки (2\*double)

routes – массив точек(string+int)

**Коллекция tip:**

id – идентификатор (int64)

name – адрес (string)

wayId – идентификатор пути (int64)

**Коллекция Way**:

Id – идентификатор пути(int64)

Tags – поля маршрута (массив пар string)

Nodes – массив точек

## 3.2. Нереляционная модель данных

Расчет памяти для MongoDB:

Средний размер node: 104байта

Средний размер Route: 481 байт

Средний размер Relation: 200 байт

Средний размер routeNode: 156 байт

Средний размер tip: 116 байт

Средний размер Way: 500 байт

Размер node: 1.2Гб

Размер Route: 0.1Гб

Размер Relation: 12Мб

Размер routeNode: 0.25Гб

Размер tip: 8Мб

Размер Way: 0.7Гб

Общий размер 2.3Гб

## 3.3. Аналог модели данных для SQL СУБД

На рисунке 1 представлена модель данных для реляционной базы данных.

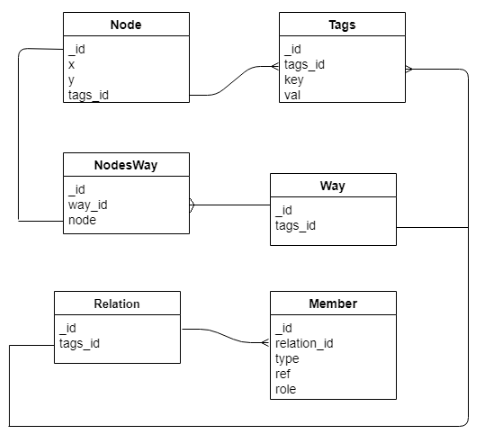


Рисунок 2 – Модель данных реляционной БД

**Описание назначений коллекций, типов данных и сущностей**

6 коллекций:

1-ая Way - хранит данные о путях.

* id () - идентификатор пути;
* tags id - id тегa пути

2-ая Node - хранит точки

* id - идентификатор точки;
* x - координата х
* y - координата у
* tags id - id тегa пути

3-я Relation - хранит отношения

* id - идентификатор точки;
* tags id - id тегa пути

4-ая Member - хранит данные об участниках

* id - идентификатор участника;
* relation id - идентификатор отношения
* type - хранит тип
* ref - хранит ссылку на id участника отношения
* role - роль участника отношения

5-ая NodesWay

* id - идентификатор
* way id - идентификатор пути;
* node - точки;

6-ая Tags

* id - идентификатор
* tags id - идентификатор тега
* key - ключ тега
* val - значение тега

**Оценка удельного объема информации, хранимой в модели**

Среднее количество тегов = 100;

Среднее количество nodes = 100

Среднее количество relation = 1000000

Среднее количество member = 10000000

Size (nodes) = 152

Size (way) = 88

Size (NodesWay) = 36

Size (Way) = 24

Size (Relarion) = 24

Size (Member) = 100

Size = (10088 + 12 + 100(88 + 88100) + 100100) \* n

## 3.4. Запросы

Примеры запросов представлены в таблице 1.

Таблица 1 – Примеры запросов

|  |  |  |
| --- | --- | --- |
| Формулировка запроса | MongoDB | SQL |
| Поиск места по названию | db.getCollection("Tip").find({ "name" : { "$regex" : "r" , "$options" : "i"}}) | SELECT \* From Tip t WHERE [t.name](https://vk.com/away.php?utf=1&to=http%3A%2F%2Ft.name) == REGEXP\_LIKE('.\*r.\*'); |
| Поиск дорог | db.getCollection("Way").find({ "tags.highway" :"road"}}) | SELECT \* FROM Way w WHERE w.type == "restriction"; |
| Информация по координатам | db.getCollection('tip').find({ "wayId" : 23744795}) | SELECT \* FROM tip t WHERE t.wayId == 23744795; |
| Ограничения на дорогах | db.getCollection("Relation").find({ "tags.type" :"restriction"}}) | SELECT \* FROM Relation r WHERE r.type == "restriction"; |
| Ближайшую к координате | db.getCollection("Node").find({ "point" : { "$near" : { "x" : 59.93794923711271 , "y" : 30.360471515615167}}}) | SELECT \* FROM Node n WHERE n.x > 55.7 AND n.x < 55.8 AND n.y > 55.7 AND n.y < 55.8; |

## 

## 3.5. Выводы

Для рассматриваемой задачи наиболее подходящей базой данных будет нереляционная БД, т.к. сложность запросов будет меньше. Но у нереляционной базы данных тоже есть свои недостатки – объем затрачиваемой памяти больше.

# 4. Разработанное приложение

## 4.1. Краткое описание

Разработанное приложение осуществляет построение маршрутов. Приложение состоит из главной станицы, на которой отображаются следующие формы:

1. Форма поиска по адресу/названию;
2. Форма построения маршрутов с информацией откуда и куда будет строится маршрут;
3. Форма «popUp» с информацией о данной метке на карте.

## 4.2. Использованные технологии

При написании приложения использовались следующие технологии:

* Java 8 – языки программирования, использованные в написании сервера;
* Spring Data – инструмент для доступа к базе данных MongoDB;
* Spring Boot – фреймвор для ускоренного написания приложений с Rest API;
* MongoDB – документированная база данных, взятая за основу и для изучения в данной работе;
* Maven – это инструмент для сборки Java проекта;
* Spring framework – фреймворк, созданный для облегчения написания веб-приложений. В данном приложении используются модуль Boot для упрощенного развертывания системы, модуль Data для создания упрощенных GET-запросов к базе данных и модель MVC для настройки веб-приложения.
* Apache commons – библиотека для открытия файла с картами;
* Junit - иблиотека для модульного тестирования программного обеспечения на языке Java;
* Google code gson – библиотека, позволяющая конвертировать объекты JSON в Java-объекты и наоборот;
* Lombok – это библиотека, с помощью которой можно сократить количество шаблонного кода, который нужно писать на Java;
* Leaflet – библиотека с открытым исходным кодом, написанная на JavaScript, предназначенная для отображения карт на веб-сайтах;
* HTML – стандартизированный язык разметки документов;
* CSS – формальный язык описания внешнего вида документа, написанного с использованием языка разметки;
* Java Script – языки программирования, использованные в написании UI слоя приложения
* JQuery – библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML.

## 4.3. Ссылки на Приложение

Исходный код приложения и инструкция по установке находятся по ссылке:

https://github.com/moevm/nosql2018-mongo\_routingЗаключение

В ходе выполнения данного индивидуального домашнего задания было разработано приложение с использованием базы данных MongoDB, с помощью которого выполняются следующие функции:

* построение маршрута из одной точки на карте в другую;
* просмотр информации о заданной точке на карте;
* экспорт данных.

Было проведено тестирование разработанного приложения, которое прошло успешно. Приложение работает корректно.

При создании приложения были изучены особенности работы нереляционных баз данных.

# Список использованных источников

1. Java Platform Standard Edition 8 Documentation URL: https://docs.oracle.com/javase/8/docs/ (дата обращения: 19.12.2018).
2. Welcome to the MongoDBDB Docs. URL: https://docs.MongoDBdb.com/?\_ga=2.140482107.947430166.1545497958-550656450.1544206702дата обращения: 19.12.2018).
3. Добро пожаловать на OpenStreetMap! URL: https://www.openstreetmap.org/#map=16/54.3561/18.6303 (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 MongoDB.
4. Запустить базу данных MongoDB.
5. Запустить jar-файл.
6. Перейти в браузере по адресу: http://localhost:8090.

# Приложение B. Инструкция для пользователя

При входе на сайт отображается географическая карта.

В левом верхнем углу располагается поиск по карте. Пользователь может ввести название или улицу и, выбрав из предложенных вариантов, посмотреть информацию по конкретному месту.

Просмотреть информацию о месте можно нажав на него. В появившемся окне будут показаны координаты данного места.

Построить маршрут пользователь может, выбрав пункт отправления и пункт назначения.

# Приложение С. Снимки экрана приложения

На рисунках 2-5 изображены снимки экрана приложения.

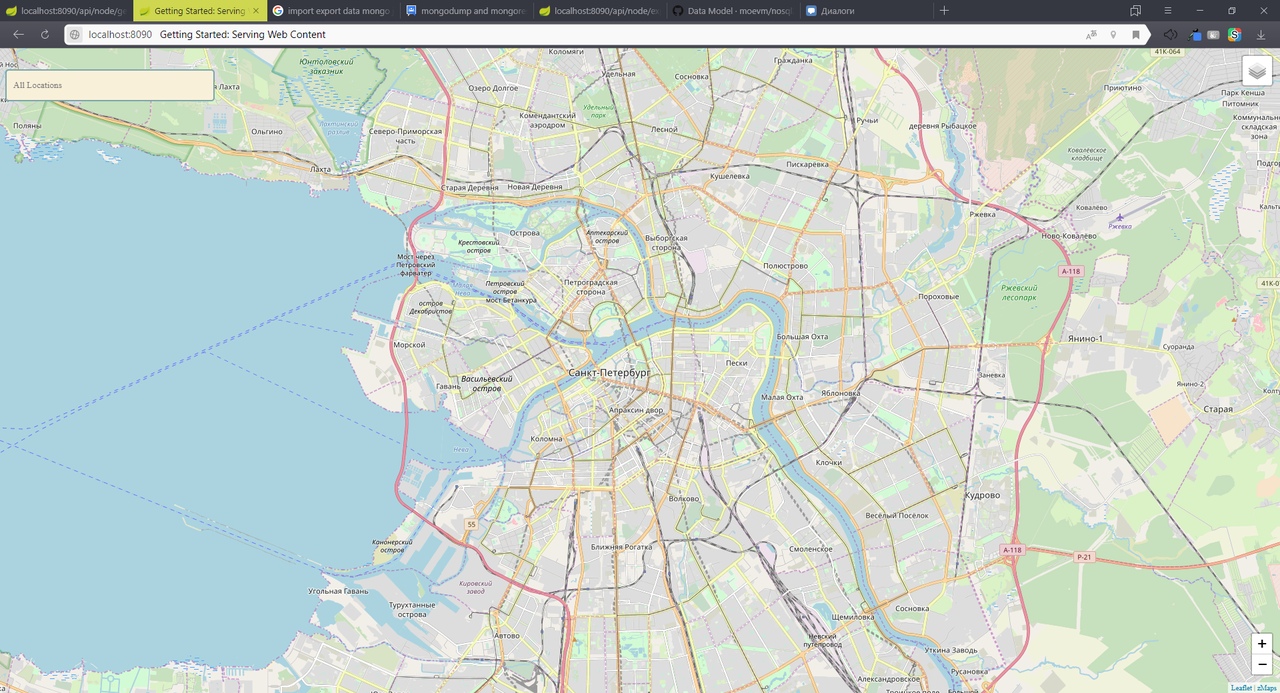


Рисунок – Главная страница сайта

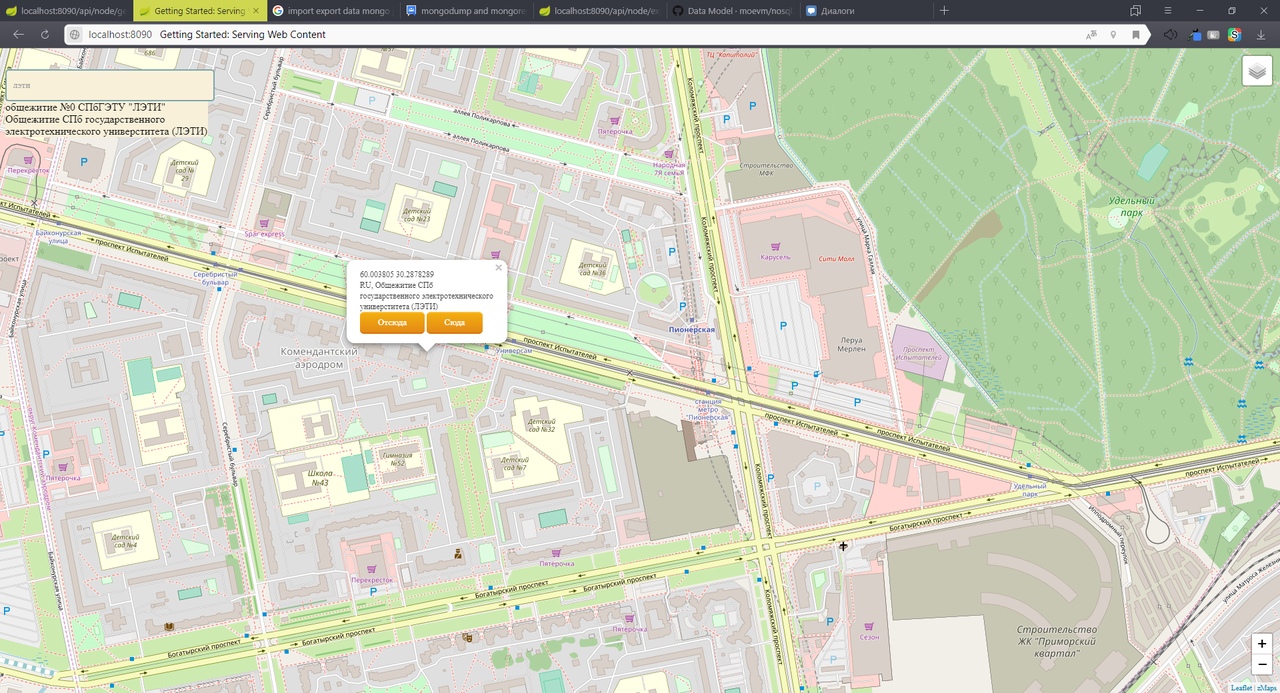


Рисунок – Поиск по названию

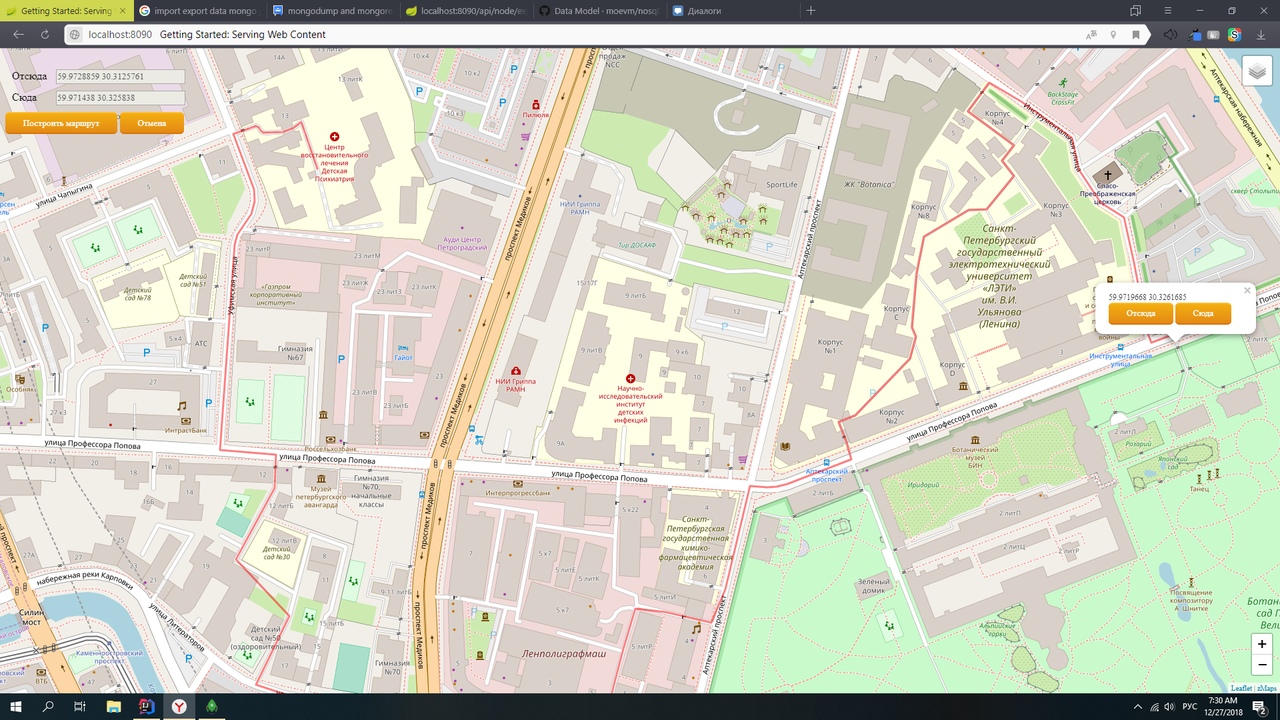


Рисунок – Построение маршрута