МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Введение в нереляционные базы данных»

Тема: ИС (примитивного) прогнозирования пробок (Mongo)

Студенты гр. 7382 _____ Гаврилов А.В.

_____ Глазунов С.А.

_____ Еременко А.

Преподаватель _____ Заславский М.М.

Санкт-Петербург
2020

ЗАДАНИЕ
НА ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

Студенты Гаврилов А.В., Глазунов С.А., Еременко А.

Группа 7382

Тема работы: 27 ИС (примитивного) прогнозирования пробок (Mongo)

Исходные данные: Реализовать информационную систему прогнозирования пробок с визуализацией карт по данным из БД

Содержание пояснительной записки:
«Содержание», «Введение», «Качественные требования к решению», «Сценарий использования», «Модель данных», «Разработанное приложение», «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:
Не менее 10 страниц.

Дата выдачи задания: 17.09.2020

Дата сдачи реферата: 12.12.2020

Дата защиты реферата: 12.12.2020

Студенты _____ Гаврилов А.В.

_____ Глазунов С.А

_____ Еременко А.

Преподаватель _____ Заславский М.М.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы разработать сайт с возможностью интерактивного взаимодействия с картой, получением статистики и построение маршрута из точки А в точку В.

1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

1. Получение и обработка данных карты Санкт-Петербурга:

- Получение данных в формате .osm
- Парсинг данных
- Запись в mongodb (Создается бд, в ней 3 коллекции: nodes, ways, relattions. Данные записываются в соответствующую коллекцию.)

2. Создание web-сервера для обработки запросов с сайта.

3. Поиск путей и общих точек:

- Поиск нужных данных исходя из запроса
- С помощью A* найти конечное число путей до цели, например, меньше 5
- Поиск общих точек во всех путях

4. Реализация сайта с картой и интерактивным интерфейсом для выбора точек откуда/куда:

- При отправке от пользователя запроса на построение возможных путей, визуализировать полученный результат
- Реализовать интерактивный интерфейс
- Реализовать визуализацию пробок

5. Поиск библиотек/сервисов/датасетов для получения данных о пробках: (Если найден, то использовать полученные данные, иначе сгенерировать псевдорандомные значения для пробок в бальной системе от 0 до 5 и 1 раз в минуту обновлять данные).

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. Макет пользовательского интерфейса.

Главная страница, см. рис. 1.

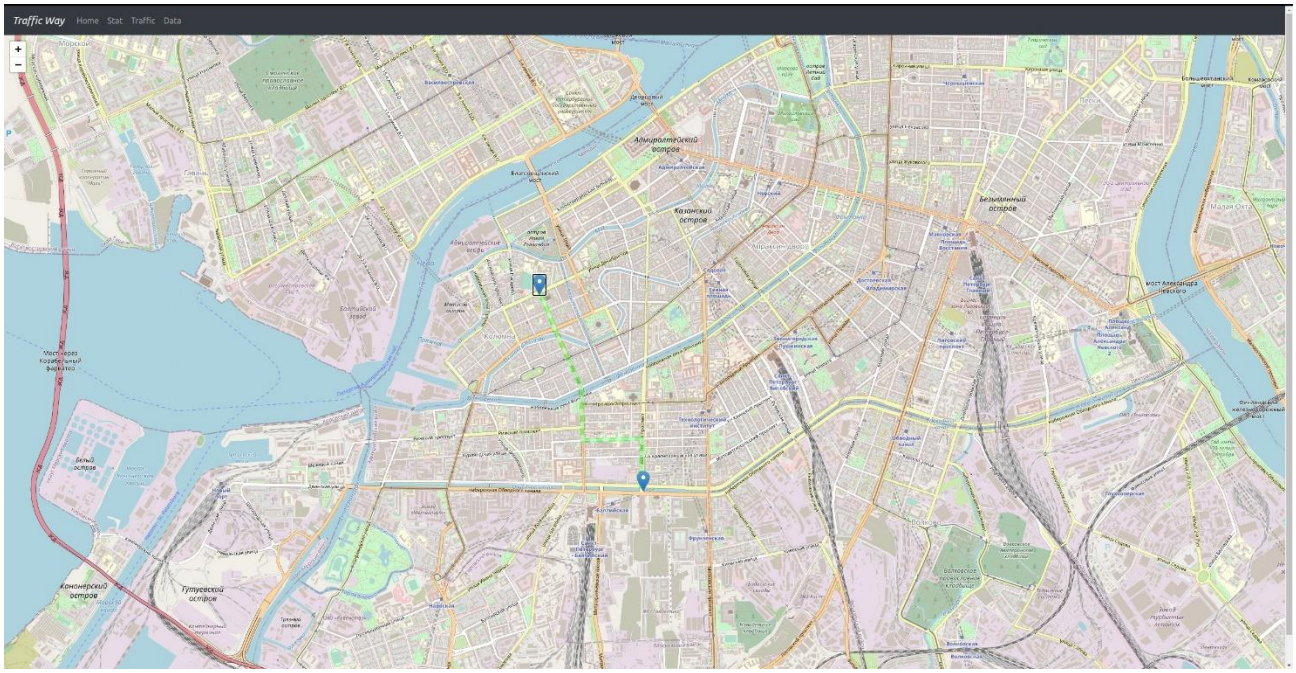

Рисунок 1 – Главная страница

Статистика пути, см. рис. 2.

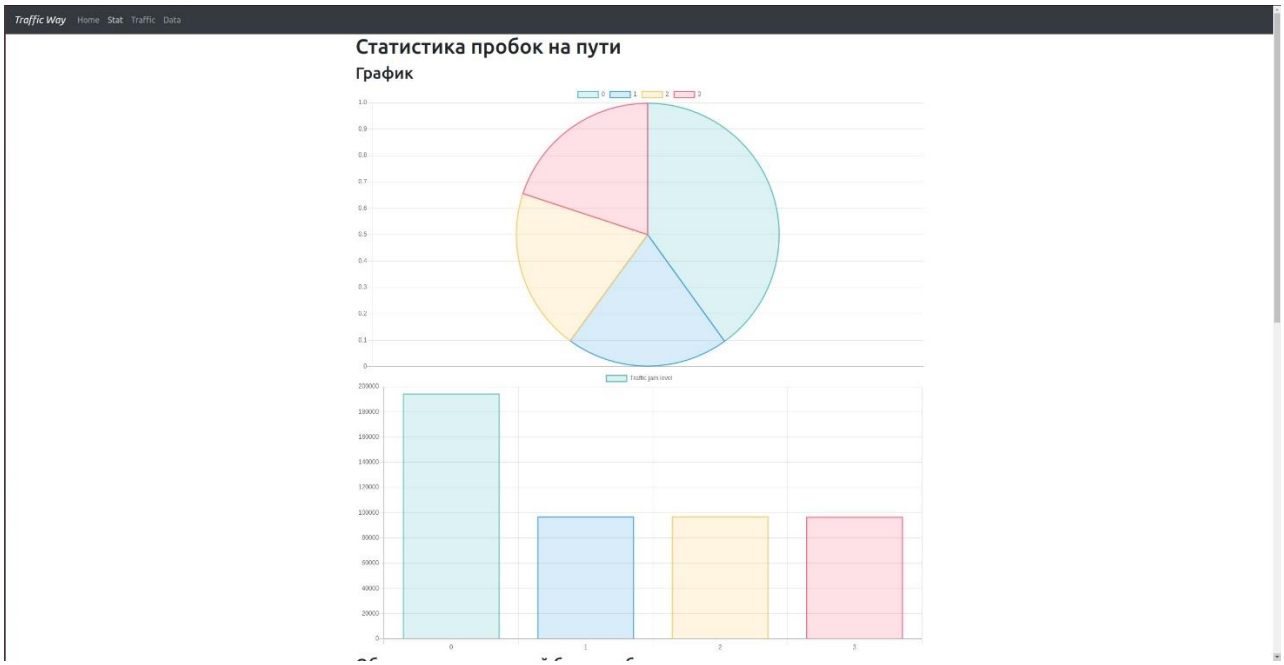

Рисунок 2 – Статистика пути

Общая статистика, см. рис. 3.

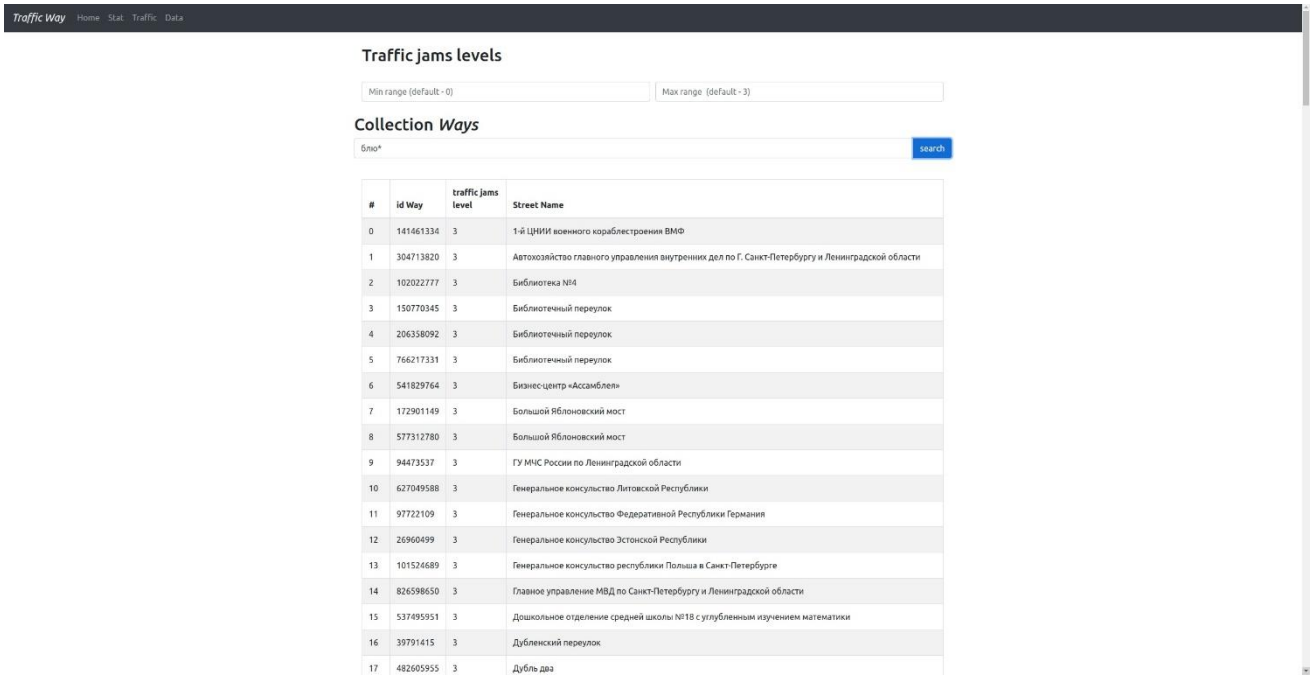

Рисунок 3 – Общая статистика

Импорт/экспорт данных, см. рис. 4.

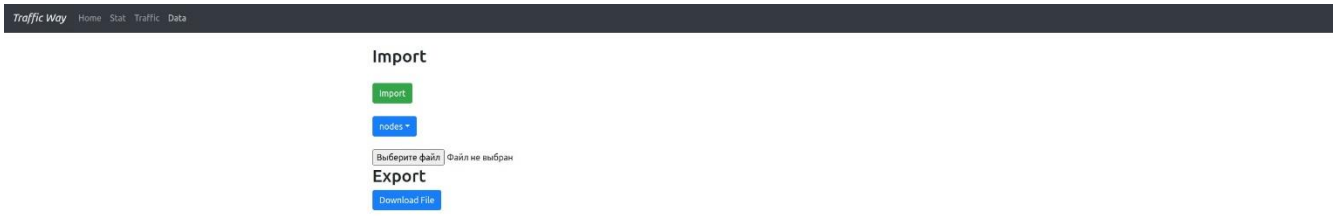

Рисунок 4 – Импорт/экспорт

2.2. Описание возможных сценариев использования.

Возможные сценарии использования включают в себя:

1. Просмотр карты.
2. Построение маршрута.
3. Просмотр данных о пробках.
4. Просмотр статистики (Диаграмма с распределением путей по уровню пробок).
5. Экспорт и импорт базы данных (Данные не переформатируются, а импортируются и экспортируются в формате, в котором они представлены в бд).

6. Просмотр страницы с информацией о данных, которые используются для прогнозирования пробок.

Рассмотрим каждый из них подробнее.

2.2.1. Просмотр карты.

Основной сценарий:

1. Пользователь заходит на сайт.

2. Пользователь может наблюдать карту Санкт-Петербурга.

3. Пользователь перемещает область видимости карты с помощью мыши

4. Пользователь масштабирует карту с помощью кнопок "+" и "-" или колесика мыши.

Запросы к бд не требуются.

2.2.2. Построение пути из точки A в точку B.

Дополнительный сценарий:

1. Пользователь заходит на сайт, видит карту СПб и 2 указателя на карте, которые можно перемещать для построения маршрута.

2. Пользователь перемещает точку A в нужную ему точку на карте.

3. Пользователь перемещает точку B в нужную ему точку на карте.

4. Строится маршрут из точки A в точку B.

Для построения маршрута используются следующие запросы из Use cases:

1. 1 – (2 раза для каждого пути).

2. 3 – (в крайнем случае алгоритма A* - n раз).

3. 2– (до 6 запросов на итерацию A* (так как перекрестков, где пересекаются более 3х дорог нет)=> в крайнем случае 6n раз).

2.2.3. Просмотр данных о пробках.

Дополнительный сценарий:

1. Пользователь заходит на сайт и переходит на вкладку "Traffic".

2. Получает о данные о загруженности дорог в виде балла.

3. Пользователь может осуществлять поиск по улицам и фильтрацию по уровню пробок.

2.2.4. Просмотр статистики базы данных.

Дополнительный сценарий:

1. Пользователь заходит на сайт и нажимает на кнопку статистики.

2. Пользователь получает статистику в виде диаграммы с информацией о том, какое количество путей соответствует каждому баллу пробок (0-3 балла).

2.2.5. Экспорт/импорт данных.

Дополнительный сценарий:

1. Пользователь заходит на сайт и нажимает на кнопку "Экспорт/импорт".

2. При импорт открывается окно загрузки данных, загружаемый файл - json файл коллекции.

3. При экспорте скачивается файл с данными бд на текущий момент времени (по файлу с коллекции).

2.2.6. Просмотр страницы с информацией о данных, которые используются для прогнозирования пробок.

Дополнительный сценарий:

1. Пользователь заходит на сайт и нажимает на кнопку "Статистика".

2. Строит нужный маршрут.

3. В виде таблицы выводятся данные, которые использовались для создания прогноза о пробках на текущем построенном пути (Возможно выводится средняя скорость, длина пути).

3. МОДЕЛЬ ДАННЫХ

3.1.Нереляционная модель данных.

3.1.1. Графическое представление.

Графическое представление нереляционной модели базы данных изображено на рис 5.

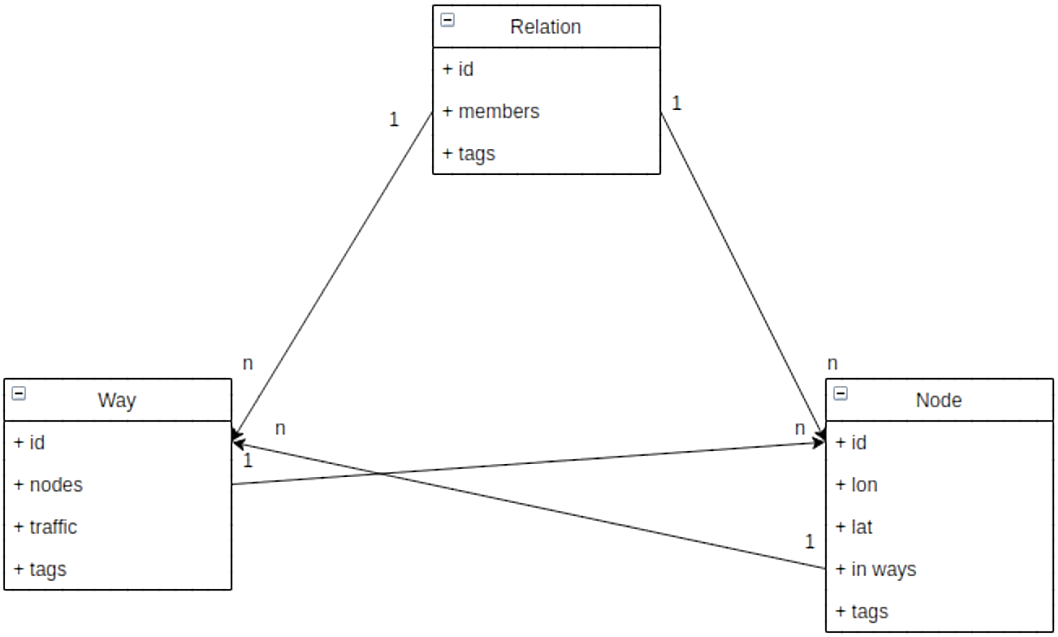

Рисунок 5 – Схема нереляционной базы данных

3.1.2. Описание назначений коллекций, типов данных и сущностей.

Описание и назначение коллекций приведено в табл. 1.

Таблица 1 – Коллекции

Кол-лекция	**Описание назначений**			
	Описание	**Название**	**Тип данных**	**Описание**
Node	Содержит информацию о точках на карте	`_id`	`ObjectID`	id точки
		`lon`	`Numbers`	долгота точки
		`lat`	`Numbers`	ширина точки
		`In_ways`	`Array`	пути в которых есть это нода

Way	Содержит информацию о дорогах на карте	`_id`	`ObjectID`	id дороги
		`nodes`	`Array`	список id точек
		`traffic`	`Numbers`	состояние дороги(пробка)
Relation	Содержит информацию о связях дорог и точек	`_id`	`ObjectID`	id отношения
		`members`	`Array`	id участников(дорога /точка) отношения

3.1.3. Оценка удельного объема информации, хранимой в модели.

В графическом представлении NoSQL и SQL версии нашей модели совпадают.

Type	Size
Objectid	12 bytes
Numbers	8 bytes
String	4 bytes

Размер одного `Node = 12 + 2*8 + ways_with_nodes * 8`

Размер одного `Way = 12 + 8 + length_nodes_in_way * 12 + length_tags_in_way * legnth_string_of_tag`

Размер одного `Relation = 12 + 8 * length_nodes_and_ways_in_relation + length_tags_in_relation * legnth_string_of_tag`

Так как количество тегов сильно разнится от объекта к объекту, то нельзя посчитать среднее для него. Также можно сказать и про количество id, которые хранятся в объектах. Поэтому выше описана конечная версия оценки размера

бд. Объем нашей БД не может быть статичным, но можно посчитать, от чего он будет зависеть. На основе этого общий объем БД составляет:

`N * (12 + 2*8 + ways_with_nodes * 8) + M * (12 + 8 + length_nodes_in_way * 12 + length_tags_in_way * legnth_string_of_tag) + L * (12 + 8 * length_nodes_and_ways_in_relation + length_tags_in_relation * legnth_string_of_tag), N, M, L >= 0.`

N - кол-во Node.

M - кол-во Way. (M ~ N/3)

L - кол-во Relation. (L ~ N/100)

`Total = N * (12B + 2*8B + ways_with_nodes(~4) * 8B + 4B * 10) + N/3 * (12B + 8B + length_nodes_in_way(~2) * 12B + length_tags_in_way(~5) * legnth_string_of_tag(~10) * 4B) + N/100 * (12B + 8B * length_nodes_and_ways_in_relation (~4) + length_tags_in_relation (~5) * legnth_string_of_tag(~10) * 4B) = N * 100B + N/3 * 244B + N/100 * 244B = N * 183B`

Избыточность данных в in_ways в Node. Это сделано для ускорения поиска путей, в которых лежит данная точка. То есть избыточная часть составляет – `N * (4 * 8)B = 32B * N`. Чистый объем - 151B.

3.1.4. Запросы к модели.

1. Поиск всех узлов, с координатами в определенном квадрате со стороной FIND_RANGE, которые являются частью какого-то пути.

```
dbnodes.find({
        'lon': {
            '$gt': y - FIND_RANGE,
            '$lt': y + FIND_RANGE
        },
        'lat': {
            '$gt': x - FIND_RANGE,
            '$lt': x + FIND_RANGE
        },
        'in_ways': {
            '$ne': []
        }
    })
```

2. Поиск пути по id.

```
db.ways.find_one({'_id': way_id})
```

3. Поиск узла по id.

```
db.nodes.find_one({'_id': node_id})
```

4. Поиск всех объектов - relations, в которые включен конкретный объект.

```
db.relations.find({'members.ref': {'$all': [id]}})
```

5. Поиск всех путей, которые проходят через данный узел.

```
db.ways.find({'nodes': {'$all': [id]}})
```

6. Изменить список путей, частью которых является данный узел.

```
dbnodes.update_one(
            {'_id': node_id},
            {'$set':
                 {'in_ways': n_ways}
             })
```

7. Сохранение узлов, путей, отношений в коллекции.

```
self.db.nodes.save({'_id': id,
                    'lon': lon,
                    'lat': lat,
                    'in_ways': []})

self.db.ways.save({'_id': id, 'nodes': nodes,'avg_speed': 0, 'tags': tags})

self.db.relations.save({'_id': id, 'members': members 'tags': tags})
```

3.2.Аналог модели данных для SQL СУБД.

3.2.1. Графическое представление.

Вышеописанную модель данных для библиотечных карточек также можно представить в виде реляционной модели, графическое представление см. рис 6.

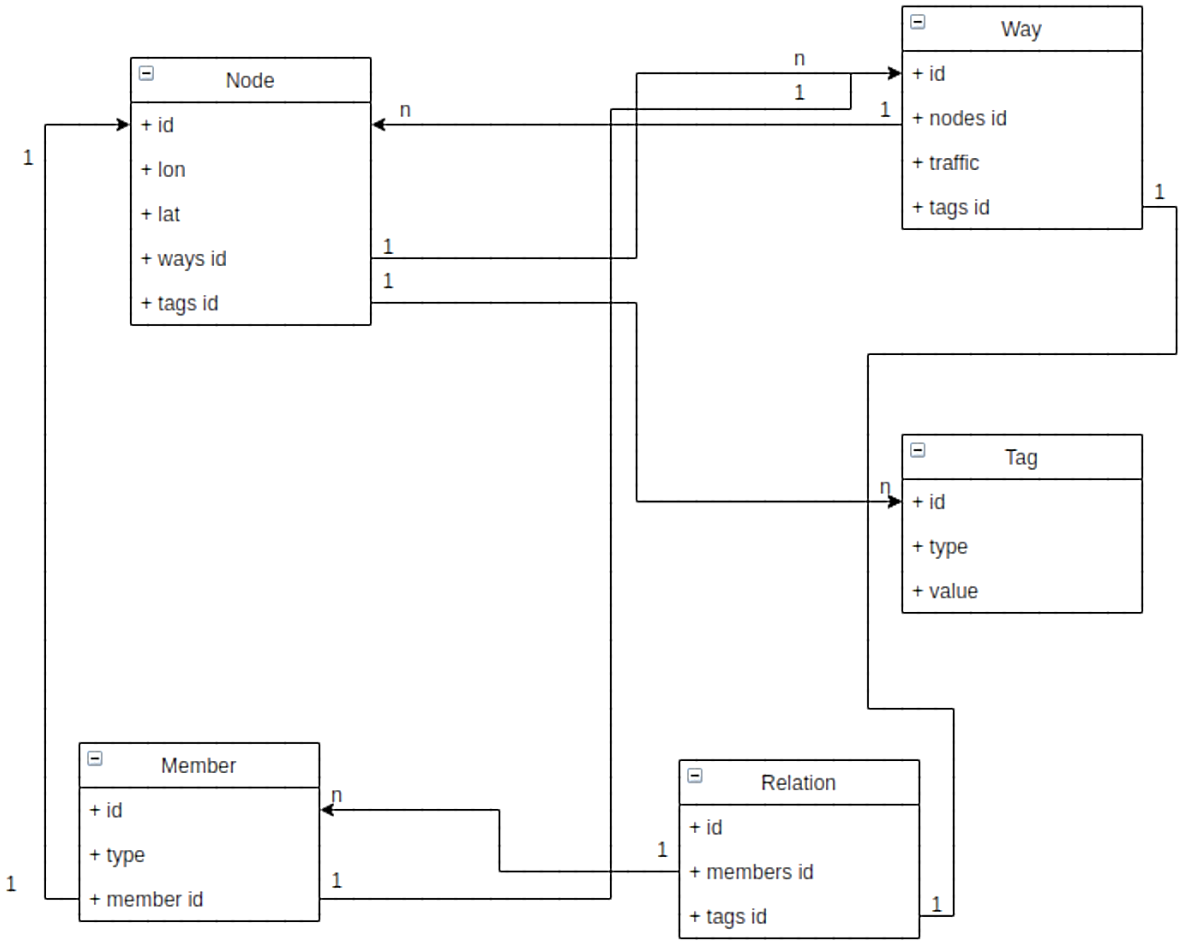

Рисунок 6 – Схема реляционной базы данных

3.2.2. Оценка удельного объема информации, хранимой в модели.

Объем реляционной базы данных будет отличаться лишь в тегах. Именно: `-N/4 * 200B - N/100 * 200B + 4B * (N + N/4 + N/100) = -40B *N. Total V = 143B * N`

3.2.3. Запросы к модели.

1. Поиск всех узлов, с координатами в определенном квадрате со стороной FIND_RANGE, которые являются частью какого-то пути.

```
SELECT * FROM nodes
WHERE (nodes.lon > y - FIND_RANGE) AND (nodes.lon < y + FIND_RANGE)
AND (nodes.lat > x - FIND_RANGE) AND (nodes.lat < x + FIND_RANGE) AND (EXISTS(nodes_in_ways))
```

2. Поиск пути по id.

```
SELECT * FROM ways
WHERE ways._id = way_id
```

3. Поиск узла по id.

```
SELECT * FROM nodes
WHERE nodes._id = node_id
```

4. Поиск всех объектов - relations, в которые включен конкретный объект.

```
SELECT * FROM relations
INNER JOIN members WHERE id in members.ref
```

5. Поиск всех путей, которые проходят через данный узел.

```
SELECT * FROM ways
WHERE id in ways.nodes
```

6. Изменить список путей, частью которых является данный узел.

```
UPDATE nodes
SET in_ways = n_ways
WHERE _id = node_id
```

7. Сохранение узлов, путей, отношений в коллекции.

```
INSERT INTO nodes
VALUES (id, lon, lat, [])


INSERT INTO ways
VALUES (id, nodes, 0, tags)


INSERT INTO nodes
VALUES (id, members, tags)
```

3.3. Вывод - что лучше, SQL или NoSQL модель.

- SQL показывает себя лучше, чем NoSQL по объему.
- В SQL реализации модели данных пришлось бы создавать дополнительные таблицы для связей, что увеличивает суммарное количество создаваемых таблиц.
- Количество запросов, необходимых для выполнения юзкейсов в SQL модели больше.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Краткое описание.

Результатом проекта должен быть сайт с возможностью интерактивного взаимодействия с картой, получением статистики и построение маршрута из точки А в точку В. Загружаться в базу данных будет только карта Санкт-Петербурга. Поиск путей будет реализован самостоятельно без использования библиотеки, которая будет визуализировать карту по данным из БД. Пробки будут визуализироваться отдельным слоем (данные генерируются рандомно либо берутся из стороннего сервиса).

4.2. Схема экранов приложения.

На сайте имеется 4 страницы и на каждую из них можно попасть только одним способом - нажатием на соответствующую кнопку сверху в навигационной панели, см. рис. 7.

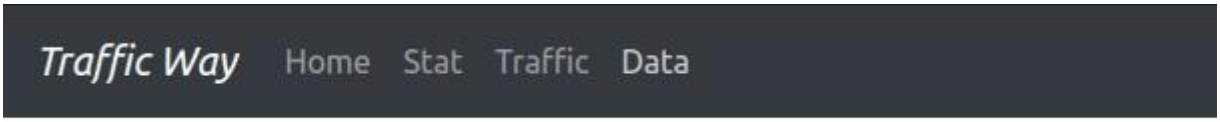

Рисунок 7 – Навигационная панель

4.3. Использованные технологии.

В качестве языка програмиирования были выбраны python (backend) и javascript (frontend), база данных: pympngo, frontend библиотека визуализации карты: Leaflet.

4.4. Ссылка на Приложение.

https://github.com/moevm/nosql2h20-traffic-mongo

4.5. Документация по сборке и развертыванию приложения

Инструкция по сборке и запуску приложения предоставлена в Readme: https://github.com/moevm/nosql2h20-traffic-mongo

ВЫВОДЫ

В ходе выполнения задания был реализован сайт с возможностью интерактивного взаимодействия с картой, получением статистики и построение маршрута из точки А в точку В. Имеется возможность загружать в базу данных карту Санкт-Петербурга. Поиск путей реализован самостоятельно без использования библиотеки, которая будет визуализировать карту по данным из БД. Пробки визуализируются отдельным слоем (данные генерируются рандомно либо берутся из стороннего сервиса). Таким образом, цель, поставленная перед началом работы, достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация MongoDB: https://docs.mongodb.com/manual/
2. Документация React:https: https://reactjs.org/docs/getting-started.html
3. Документация Docker https://dker.ru/docs/