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

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Введение в нереляционные базы данных»

Тема: Разработка веб-инструмента для обнаружения узких мест в транспортной сети

Студентка гр. 9304 _____ Селезнёва А.В.

Студент гр. 9304 _____ Тиняков С.А.

Студент гр. 9304 _____ Цаплин И.В.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2022

ЗАДАНИЕ

Студенты

Селезнёва А.В.

Тиняков С.А.

Цаплин И.В.

Группа 9304

Тема работы: Разработка веб-инструмента для обнаружения узких мест в транспортной сети

Исходные данные:

Необходимо разработать веб-инструмент, позволяющий визуализировать дорожную сеть на карте с расцветкой по уровню пробок, а также выполняющий анализ для заданных фрагментов сети на предмет наличия узких мест транспортного потока.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Качественные требования к решению»

«Сценарии использования»

«Модель данных»

«Разработанное приложение»

«Выводы»

«Приложения»

Предполагаемый объем пояснительной записки:

Не менее 30 страниц.

Дата выдачи задания: 01.09.2022

Дата сдачи реферата: 19.12.2022

Дата защиты реферата: 19.12.2022

Студентка гр. 9304	_____	Селезнёва А.В.
Студент гр. 9304	_____	Тиняков С.А.
Студент гр. 9304	_____	Цаплин И.В.
Преподаватель	_____	Заславский М.М.

АННОТАЦИЯ

Целью данного индивидуального задания является разработка веб-приложения с использованием MongoDB в качестве СУБД. В результате выполнения работы был разработан веб-инструмент, позволяющий визуализировать дорожную сеть на карте с расцветкой по уровню пробок, а также выполняющий анализ для заданных фрагментов сети на предмет наличия узких мест транспортного потока.

ANNOTATION

The goal of the individual homework assignment is to develop web application using MongoDB as the DBMS. As a result of the work, a web tool was developed for visualizing road network on the map with coloring according to the performance of traffic jams, as well as for performing analysis for given network fragments on existence of bottlenecks of the traffic flow.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Узким местом дорожной сети является локальное нарушение движения транспортных средств на улице, дороге или шоссе. В отличие от пробки, узкое место является результатом определенного физического состояния, часто конструкции дороги, неправильно синхронизированных светофоров или крутых поворотов.

Наличие узких мест приводит к заторам на дорогах, которые имеют ряд негативных последствий:

- Трата времени автомобилистов и пассажиров. Будучи непроизводственной деятельностью для большинства людей, пробки снижают экономическое здоровье региона;
- Расход топлива увеличивает загрязнение воздуха и выбросы углекислого газа из-за увеличения холостого хода, ускорения и торможения;
- Изнашивание транспортных средств в результате простоя в движении и частых разгонов и торможений приводит к более частым ремонтам и заменам;
- Большее количество ДТП, стресс, ухудшение здоровья автомобилистов и т.д.

Была поставлена задача разработать веб-инструмент, позволяющий визуализировать дорожную сеть на карте с расцветкой по уровню пробок, а также выполняющий анализ для заданных фрагментов сети на предмет наличия узких мест транспортного потока.

Предлагаемое решение: разработать веб-инструмент, которым будут пользоваться градостроительные проектировщики для симуляции загруженности дорог и выявления узких транспортных мест в городах.

КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

К решению выдвигаются следующие качественные требования:

- Задание фрагмента дорожной сети для симуляции;
- Анализ данных, полученных при симуляции;
- Сохранение и загрузка полученных данных.

СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

Макет UI.

Был разработан макет пользовательского интерфейса для разрабатываемого веб-инструмента. Разработанный макет представлен в приложении А.

Сценарии использования для задачи.

Сценарий использования – «Моделирование маршрутов из точки А в точку Б»:

Действующее лицо: Пользователь.

Предусловие: открыто данное веб-приложение в первоначальном виде.

Основной сценарий:

1. Нажать на точку на карте, которая будет служить отправным пунктом для маршрута – на карте на этом месте появилась отметка пин;
2. Нажать на точку на карте, которая будет служить пунктом прибытия для маршрута – на карте на этом месте появилась отметка пин и радиус поиска маршрутов;
3. В поле “Количество машин” ввести то количество машин, которое пользователь хочет пустить по дорогам маршрутов – введенное число появилось в указанном поле;
4. Выставить нужное значение радиуса поиска маршрутов на слайдере – при изменении значения на слайдере область на карте будет также изменяться;
5. Нажать на кнопку “Смоделировать” – на карте появятся возможные маршруты с отображением степени загруженности дорог, в таблице появятся данные о загруженности каждой дороги маршрутов в

порядке убывания загруженности, также появится информация о количестве маршрутов.

Альтернативный сценарий:

- Нажать на кнопку “Очистить точки на карте” – карта примет первоначальный вид без отметок.
- Нажать на чекбокс – галочка, стоящая по умолчанию, исчезнет, визуализация радиуса на карте пропадет.

Сценарий использования – “Фильтрация данных о дорогах”.

Действующее лицо: Пользователь.

Предусловие: Смоделирован маршрут по заданным пользователем точкам на карте, тумблер (toggle button) установлен в состояние “Дороги”.

Основной сценарий:

1. В поле “Адрес/имя” ввести адрес – введенная строка появилась в указанном поле;
2. В поле “min” ввести минимальное значение диапазона загруженности – введенное число появилось в указанном поле;
3. В поле “max” ввести максимальное значение диапазона загруженности – введенное число появилось в указанном поле;
4. В выпадающем меню “Тип дорог” выбрать нужный тип дороги;
5. Нажать на кнопку “Отфильтровать” – в таблице ниже появятся отфильтрованные значения в порядке убывания загруженности.

Альтернативный сценарий:

- Нажать на строку с дорогой в таблице — на карте выбранная дорога будет выделена окружностью.

Сценарий использования – “Фильтрация данных о маршрутах”:

Действующее лицо: Пользователь.

Предусловие: Смоделирован маршрут по заданным пользователем точкам на карте, тумблер (toggle button) установлен в состояние “Маршруты”.

Основной сценарий:

1. В поле “min” для длины маршрута ввести минимальное значение длины маршрута в км – введенное число появилось в указанном поле;
2. В поле “max” для длины маршрута ввести максимально значение длины маршрута в км – введенное число появилось в указанном поле;
3. В поле “min” для времени пути ввести минимальное значение времени в минутах - введенное число появилось в указанном поле;
4. В поле “max” для времени пути ввести максимальное значение времени в минутах - введенное число появилось в указанном поле;
5. Нажать на кнопку “Отфильтровать” – в таблице ниже появятся отфильтрованные значения в порядке убывания времени в пути.

Альтернативный сценарий:

- Нажать на строку с маршрутом в таблице — на карте останется только выбранный маршрут.

Сценарий использования – “Импорт данных”:

Действующее лицо: Пользователь.

Предусловие: открыто данное веб-приложение.

Основной сценарий:

1. Кликнуть на кнопку “Импорт”;
2. В появившемся диалоговом окне выбрать файл с данными в машиночитаемом формате JSON;
3. Кликнуть на кнопку “Импортировать”;
4. Дождаться окончания импорта данных.

a. При успешном импорте данных диалоговое окно закроется и отобразятся новые данные

b. При ошибке во время импортирования появится сообщение с ошибкой

Сценарий использования - “Экспорт данных”:

Действующее лицо: Пользователь.

Предусловие: Смоделирован маршрут по заданным пользователем точкам на карте.

Основной сценарий:

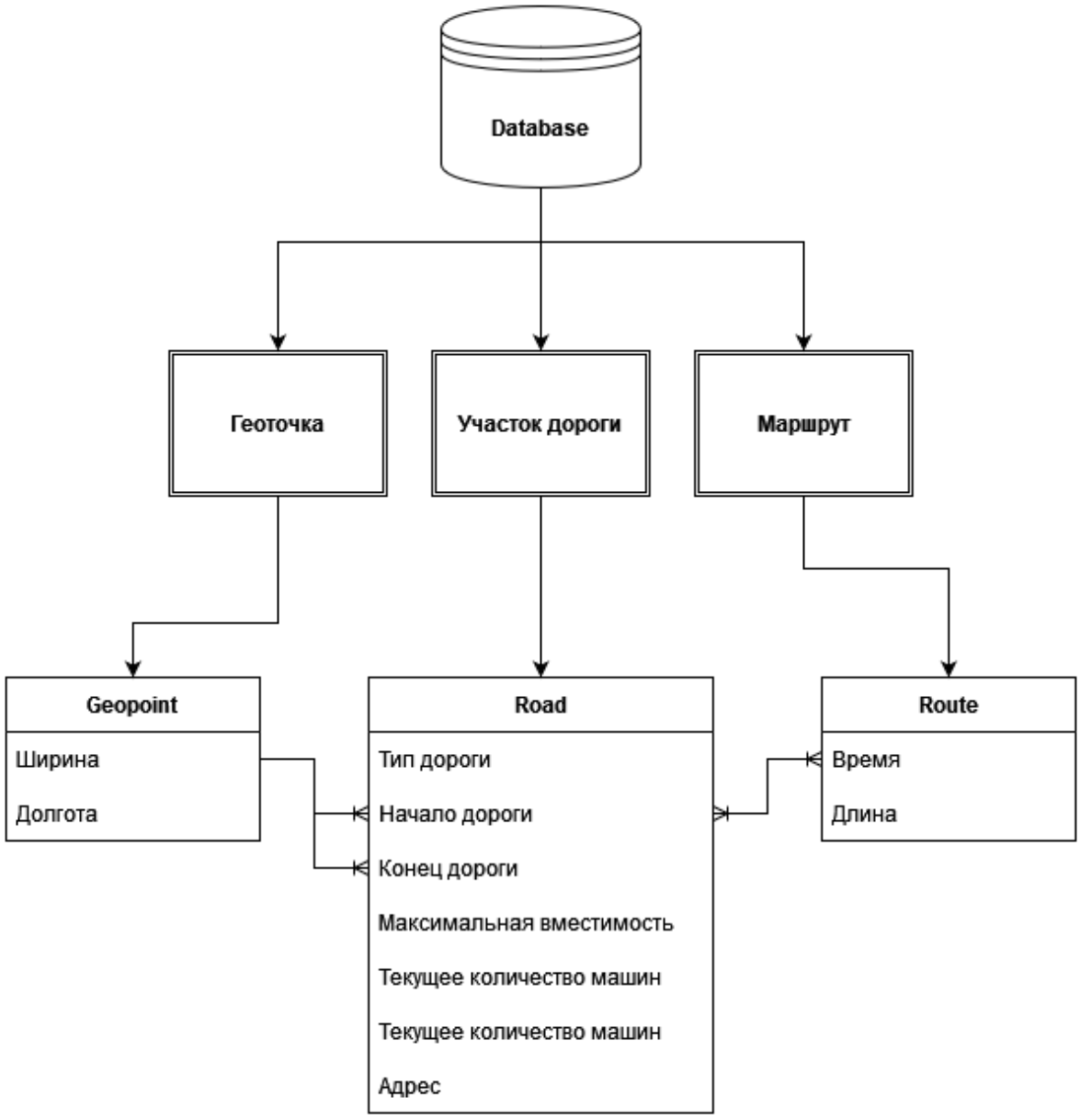
1. Кликнуть на кнопку “Экспорт”;
2. Выбрать место для сохранения файла с данными в машиночитаемом формате JSON.

Вывод.

Операции на запись используются только в сценариях “Моделирование маршрутов из точки А в точку Б” и “Импорт данных”. К тому же данные сценарии не используются в большом количестве, в отличии от оставшихся сценариев, которые используют только операции на чтение. Поэтому можно сделать вывод, что преобладают операции на чтение.

МОДЕЛЬ ДАННЫХ

Список сущностей.



Геоточка – обозначение точки на Земной плоскости:

Атрибуты:

- Широта;
- Долгота.

Участок дороги – основная сущность, которая отображает состояние дороги:

- Начало дороги – Геоточка начала дороги;

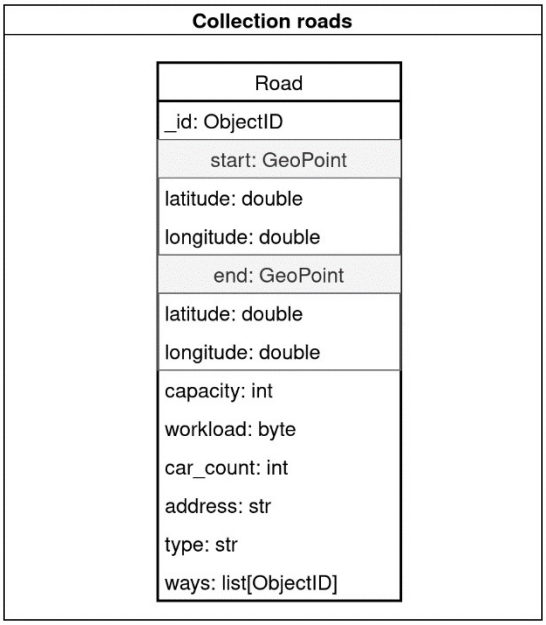
- Конец дороги – Геоточка конца дороги;
- Максимальная вместимость – количество машин, которое данный участок может вместить;
- Текущее количество машин – текущие количество машин на дороге;
- Адрес – текстовое представление данного участка дороги;
- Тип дороги – обозначение типа дороги, например, перекрёсток, проспект, шоссе.

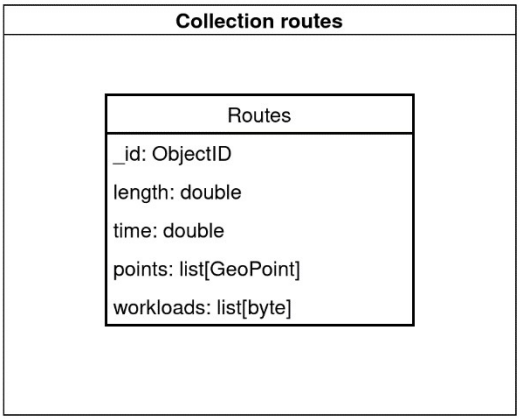
Маршрут – найденные маршруты, состоящие из участков дорог:

- Длина– длина маршрута в километрах;
- Время – время в минутах для проезда по данному маршруту.

Нереляционная модель данных.

Графическое представление.





Описание коллекций, типов данных и сущностей.

Данные будут храниться в двух коллекциях: одна для дорог, другая для маршрутов.

В коллекции roads, в которой будет храниться одна сущность – Road (дорога). Road будет иметь следующие атрибуты:

- _id: ObjectID – необходимо для любого объекта в MongoDB;
- start: GeoPoint, end: GeoPoint – сущности geopoint, которые служат для определения догори в плоскости Земли;
- capacity: int – максимальная вместимость машин;
- car_count: int – текущее количество машин;
- workload: byte – загруженность по 10-и балльной шкале. Данный атрибут необходим для быстрого поиска в MongoDB;
- address: str – адрес;
- type: str – тип дороги;
- ways: list[ObjectID] – лист с _id дорог, до которых можно добраться из данной дороги.

Сущность GeoPoint отдельно не хранится, она всегда является вложенным документов в Road. Атрибуты GeoPoint:

- latitude: double – широта;
- longitude: double – долгота.

В коллекции routes, в которой будет храниться одна сущность – Route (маршрут). Route будет иметь следующие атрибуты:

- _id: ObjectID – необходимо для любого объекта в MongoDB;
- length: double – длина пути в километрах;
- time: double – время для проезда по данному маршруту с учётом загруженности дорог;
- points: list[GeoPoint] – точки в пространстве, через которые проходит данный маршрут;
- workloads: list[byte] – загруженность дорог в маршруте, т.е. загруженность участка между двумя точками. Значение по индексу 0 соответствует загруженности между точкой под индексом 0 и 1, значение под индексом 1 – для точек под индексами 1 и 2, и т.д.

Оценка объёма информации.

Объём сущности GeoPoint:

$$2 \cdot sizeof(double) = 2 \cdot 8 = 16 байта.$$

Объём сущности Road:

$$sizeof(ObjectID) + 2 \cdot sizeof(GeoPoint) + 2 \cdot sizeof ¿$$

$$¿ 12 + 2 \cdot 16 + 2 \cdot 4 + 1 + S + T + W \cdot sizeof(ObjectID) = 53 + S + T + 12 \cdot W байт,$$

где S – длина строки, T – длина строки для обозначения типа дороги, W – количество дорог, до которых можно добраться из текущей.

Возьмём средние значения:

- S – средняя длина названий улиц ~25 символов;
- T – средняя длина ~8 символов;
- W – среднее значение путей ~2 ед., т.е. в среднем один поворот (проехать прямо, повернуть налево/направо).

Получаем средний объём:

$$53 + S + T + 12 \cdot W = 53 + 25 + 8 + 12 \cdot 3 = 122 байт$$

Объём сущности Route:

$$sizeof(ObjectID) ë + 2 \cdot sizeof(double) + (1 + sizeof(GeoPoint)) \cdot R = 12 + 2 \cdot 8 + (1 + 16) \cdot R = 28 + 17 \cdot R байт$$

Возьмём среднее значение R: в худшем случае поиск маршрутов будет осуществляться по всей карте, тогда длину маршрута можно взять как

$$2 \cdot \sqrt{N}$$

Получаем средний объём:

$$28 + 17 \cdot 2 \cdot \sqrt{N} = 28 + 34 \cdot \sqrt{N},$$

где N – количество дорог в коллекции Roads

Итоговый объём информации:

$$122 \cdot N + (28 + 34 \cdot \sqrt{N}) \cdot M = 122 \cdot N + (28 + 34 \cdot \sqrt{N}) \cdot 1.7 \cdot N = (170 + 58 \cdot \sqrt{N}) \cdot N байт,$$

где N – количество дорог.

Преобразование M в 1.7 $\sqrt{N}$ описано в приложении Б.

Избыточность модели.

Каждая дорога соединена с хотя бы одной другой дорогой, тогда получается, что они имеют одну общую точку GeoPoint. Таким образом каждая дорога содержит избыточные данные о своём положении объёмом в размер сущности GeoPoint. Также избыточными данными является информация о загруженности (workload), потому что её можно получить из значений capacity и cars_count. Таким образом избыточный объём равен

$$sizeof(GeoPoint)+sizeof(byte)=16+1=17\ байт.$$

Каждый маршрут содержит в себе часть информации о дорогах, которая дублируется из сущности Road. Таким образом атрибуты points и workloads являются полностью избыточными, их можно было бы заменить на ObjectID дорог (Roads). 17 - 12 = 5 байт избыточности данных.

Общая избыточность модели составляет:

$$17\cdot N+5\cdot 2\cdot\sqrt{(N)}\cdot M=17\cdot N+10\cdot\sqrt{N}\cdot M=(1+\sqrt{N})\cdot 17\cdot N\ байт,$$

где N – количество дорог.

Преобразование M в $1.7\cdot N$ описано в приложении Б

Итоговая избыточность:

$$((170+58\cdot\sqrt{N})\cdot N)/((153+41\cdot\sqrt{N})\cdot N)=1.42$$

Так как $N\cdot\sqrt{N}$ растёт быстрее, чем N, то коэффициенты 58 и 41 являются главными, поэтому отношение между ними будет являться коэффициентом избыточности.

Рост модели.

Модель растёт с полиномиальной скоростью, так как переменная N присутствует два раза со степенями 1 и 3/2.

Запросы к модели.

Сценарий “Построение модели”.

Запрос на получение дорог в заданной области:

```
db.roads.find({
     "start.latitude": {
     $gt: lower_lat,
     $lt: upper_lat
     },
     "start.longitude": {
     $gt: lower_lng,
     $lt: upper_lng
     },
     "end.latitude": {
     $gt: lower_lat,
     $lt: upper_lat
     },
     "end.longitude": {
     $gt: lower_lng,
     $lt: upper_lng
     },
     $where: function() {
     return (((this.start.latitude - center_latitude) ** 2) /
(latitude_radius ** 2) + ((this.start.longitude -
center_longitude) ** 2) / (longitude_radius ** 2) <= 1
     && ((this.end.latitude - center_latitude) ** 2) /
(latitude_radius ** 2) + ((this.end.longitude -
center_longitude) ** 2) / (longitude_radius ** 2) <= 1)
     }
})
```

Запрос на обновление состояния загруженности определённой дороги:

```
db.roads.updateOne({
     "_id": ObjectId(...)
},
{
     $set: {
     "car_count": cars_on_road
     }
})
```

Запрос на добавление нового маршрута:

```
db.routes.insertMany([
     {
     "length": route_len1,
     "time": route_time1,
     "points": [
          {
                "latitude": lat1,
               "longitude": lng1
          },
```

```
            {
                  "latitude": lat2,
                  "longitude": lng2
            },
            {
                  "latitude": lat3,
                  "longitude": lng3
            }
      ],
      "workloads": [
            workload1,
            workload2
      ]
      },
      {
      "length": route_len2,
      "time": route_time2,
      "points": [
            {
                  "latitude": lat4,
                  "longitude": lng4
            },
            {
                  "latitude": lat5,
                  "longitude": lng5
            },
            {
                  "latitude": lat6,
                  "longitude": lng6
            }
      ],
         "workloads": [
            workload3,
            workload4
      ]
      }
])
```

Количество запросов:

- 1 запрос на получение дорог;
- 1 запрос на создание M маршрутов;
- R запросов на обновление измененных дорог;
- Общее количество: 2 + R запросов.

Использованные коллекции:

- roads;
- routes.

Сценарий “Фильтрация данных о дорогах”.

Запрос:

```
db.roads.find({
     "workload": {
     $lte: max_workload,
     $gte: min_worklaod
     },
     "address": {
     $regex: address_part
     },
     "type": road_type
})
```

Количество запросов: 1 запрос.

Использованные коллекции: roads.

Сценарий “Фильтрация маршрутов”.

Запрос:

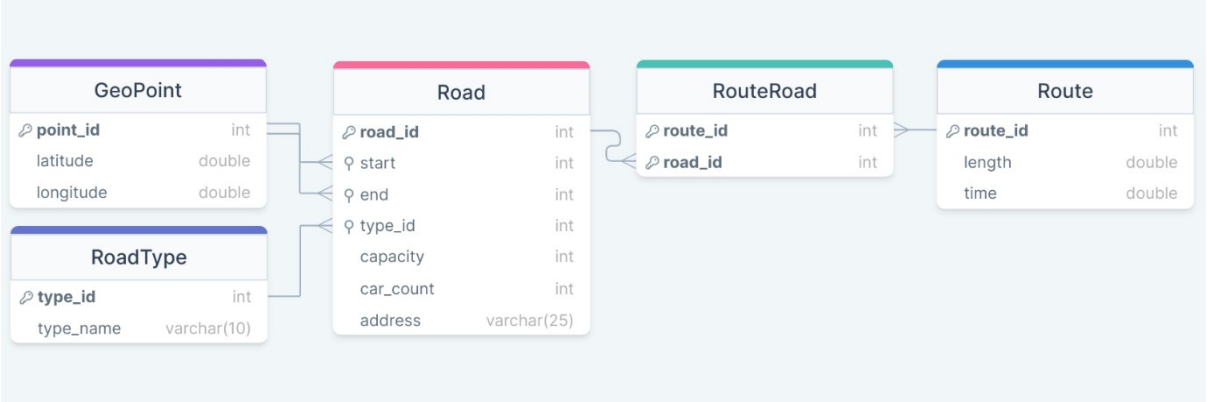
```
db.routes.find({
     "length": {
     $lt: max_len,
     $gt: min_len
     },
     "time": {
     $lt: max_route_time,
     $gt: min_route_tim
     }
})
```

Количество запросов: 1 запрос.

Использованные коллекции: routes.

Реляционная модель данных.

Графическое представление.



Описание коллекций, типов данных и сущностей:

- Таблица GeoPoint — хранит информацию о сущности *Геоточка*:
  - point_id: int — уникальный идентификатор точки;
  - latitude: double — широта;
  - longitude: double — долгота.

- Таблица Road — хранит информацию о сущности *Участок дороги*:
  - road_id: int — уникальный идентификатор участка дороги;
  - start: int — идентификатор точки, которая является началом участка дороги;
  - end: int — идентификатор точки, которая является концом участка дороги;
  - type_id: int — идентификатор типа дороги;
  - capacity: int — максимальная пропускная способность участка дороги;
  - car_count: int — количество машин на данном участке дороги;
  - address: varchar(25) — адрес.

- Таблица RoadType — хранит возможные типы дорог:

- type_id: int — уникальный идентификатор типа;
- type_name: varchar(10) — название типа.

- Таблица Route — хранит информацию о сущности *Маршрут*:
- route_id: int — уникальный идентификатор маршрута;
- length: int — длина маршрута в километрах;
- time: double — время в минутах, требуемое для проезда по данному маршруту с учётом загруженности дорог.

- Таблица RouteRoad — хранит информацию о принадлежности участка дороги к определённому маршруту:
- route_id — идентификатор маршрута;
- road_id — идентификатор участка дороги.

Оценка объёма информации.

Объём одной записи в таблице GeoPoint:

sizeof ¿.

Объём одной записи в таблице Road:

sizeof ¿.

Объём одной записи в таблице RoadType:

sizeof ¿.

Объём одной записи в таблице Route:

sizeof ¿.

Объём одной записи в таблице RouteRoad:

sizeof ¿,

N — количество участков дорог, 2N — количество геоточек.

Возьмём среднюю длину маршрута из пункта о нереляционной модели данных $2 \cdot \sqrt{N}$.

Возьмём количество маршрутов $M = 1.7 \cdot N$, расчёт примерной зависимости представлен в Приложении Б. $2 \cdot \sqrt{N} \cdot M$ — количество записей в RouteRoad.

Возьмём количество типов дорог равное 3 (прим. улица, проспект, перекрёсток).

Итоговый объём информации:

$$N \cdot sizeof(Road) + 2 \cdot N \cdot sizeof(GeoPoint) + 2 \cdot \sqrt{(N)} \cdot M \cdot sizeof(RouteRoad) + M \cdot sizeof(Route) + 3 \cdot sizeo$$

.

Избыточность модели

Модель не несёт в себе избыточности.

Рост модели.

Так же, как и в случае нереляционной модели, модель растёт с полиномиальной скоростью.

Запросы к модели.

Сценарий “Построение модели”:

Необходимо получить две точки, ближайшие к указанным на карте.

```
SELECT TOP 1 * FROM GeoPoint
ORDER BY SQRT( POWER((latitude - input_latitude), 2) +
POWER((longitude - input_longitude), 2) ) DESC;
```

Данный запрос необходимо выполнить два раза для получения двух точек для анализа маршрутов между ними.

Необходимо получить все участки дорог в заданной области. Входные данные – центр эллипса и длина его осей. Формула для проверки того, находится ли точка внутри эллипса:

$$\frac{(x-h)^2}{r_x^2} + \frac{(y-k)^2}{r_y^2} \le 1,$$

где (h, k) – координаты центра, (r_x, r_y) – длины полуосей.

```
SELECT * FROM Road
INNER JOIN GeoPoint as start_point ON Road.start_id =
start_point.point_id
INNER JOIN GeoPoint as end_point ON Road.end_id =
end_point.point_id
WHERE ((POWER(start_point.latitude - input_latitude, 2) /
POWER(input_latitude, 2) + POWER(start_point.longitude -
input_longitude, 2) / POWER(input_longitude_axis, 2)) <= 1)
AND  WHERE ((POWER(end_point.latitude - input_latitude,
2) / POWER(input_latitude, 2) + POWER(end_point.longitude -
input_longitude, 2) / POWER(input_longitude_axis, 2)) <=
1);
```

Необходимо обновить данные о загруженности участков дорог.

```
UPDATE Road
SET car_count=input_car_count
WHERE road_id=input_road_id;
```

Данный запрос необходимо выполнить R раз, где R – число измененных участков дорог

Необходимо добавить полученные маршруты в таблицу Routes и полученные участки дорог из маршрутов в таблицу RouteRoad.

```
START TRANSACTION
DECLARE route_id int
INSERT INTO Route
VALUES (length, time, workload)
SET route_id = LAST_INSERT_ID()
INSERT INTO RouteRoad
VALUES         (route_id, road_id_1), … (route_id,
road_id_S)
COMMIT;
```

Данный запрос на добавление маршрута и добавление всех его участков дорог выполняется M раз, где M - количество маршрутов.

Количество запросов: $3+M+R$.

Количество коллекций: 2.

Сценарий “Фильтрация информации об участках дорог”.

Необходимо выбрать участки дорог, соответствующие заданным параметрам.

```
SELECT * FROM Road
INNER JOIN RoadType
```

```
     ON Road.type_id = RoadType.id
WHERE Road.car_count / Road.capacity * 10 >= input_min
AND Road.car_count / Road.capacity * 10 <= input_max
AND Road.address = input_address
AND RoadType.type_name = input_type
Количество запросов: 1
Количество коллекций: 2
SELECT * FROM Road
INNER JOIN RoadType
     ON Road.type_id = RoadType.id
WHERE Road.car_count / Road.capacity * 10 >= input_min
AND Road.car_count / Road.capacity * 10 <= input_max
AND Road.address = input_address
AND RoadType.type_name = input_type
```

Количество запросов: 1.

Количество коллекций: 2.

Сценарий “Фильтрация маршрутов”.

Необходимо выбрать маршруты, соответствующие заданным параметрам.

```
SELECT * FROM Route
WHERE length  >= input_min_len
AND length <= input_max_len
AND time  >= input_min_time
AND time <= input_max_time
```

Количество запросов: 1.

Количество коллекций: 1.

Сравнение моделей.

Удельный объем информации.

N - количество дорог.

NoSQL:

- Объем информации: $58 \cdot N^{\frac{3}{2}} + 170 \cdot N$ байт;
- Избыточность: 1.42.

SQL:

- Объем информации: $27.2 \cdot N^{\frac{3}{2}} + 117.9 \cdot N + 42$ байт;

- Избыточность: отсутствует.

Запросы по отдельным сценариям использования:

NoSQL:

Сценарий “Построение модели”:

- Количество операций: 2 + R, R – число обновлённых дорог;
- Количество коллекций: 2.

Сценарий “Фильтрация данных о дорогах”:

- Количество операций: 1;
- Количество коллекций: 1.

Сценарий “Фильтрация данных о маршрутах”:

- Количество операций: 1;
- Количество коллекций: 1.

SQL:

Сценарий “Построение модели”:

- Количество операций: 3 + M + R, R - число обновлённых дорог, M - число добавленных маршрутов;
- Количество коллекций: 3.

Сценарий “Фильтрация данных о дорогах”:

- Количество операций: 1;
- Количество коллекций: 2.

Сценарий “Фильтрация данных о маршрутах”:

- Количество операций: 1;
- Количество коллекций: 1.

SQL модель базы данных превосходит NoSQL модель по удельному объёму информации за счёт отсутствия дублирования информации. Но в SQL модели требуется больше запросов и используется больше таблиц для реализации сценариев использования. Система предназначена для помощи

в принятии решений при проектировании городов, карта которых содержит большое количество участков дорог. Следовательно, приоритетным является возможность системы хранить большое количество данных, скорость выполнения запросов второстепенна. Таким образом, для реализации системы более подходящей является SQL модель базы данных.

Примеры хранения данных в MongoDB.

```
# db.roads.find()
[
  {
     _id: ObjectId("635a95381cc85f50234036f4"),
     start: { latitude: 59.978279, longitude: 30.3271163 },
     end: { latitude: 59.9813885, longitude: 30.3214304 },
     capacity: 10,
     car_count: 3,
     workload: 3,
     address: 'Улица Попова',
     type: 'улица',
     ways: [ ObjectId("635a95381cc85f50234036f5") ]
  },
  {
     _id: ObjectId("635a95381cc85f50234036f5"),
     start: { latitude: 59.9813885, longitude: 30.3214304 },
     end: { latitude: 59.9726873, longitude: 30.3227888 },
     capacity: 10,
     car_count: 8,
     workload: 8,
     address: 'Улица Попова',
     type: 'проспект',
     ways: [ ObjectId("635a95391cc85f50234036f6") ]
  },
  {
     _id: ObjectId("635a95391cc85f50234036f6"),
     start: { latitude: 59.9726873, longitude: 30.3227888 },
     end: { latitude: 59.978279, longitude: 30.3271163 },
     capacity: 5,
     car_count: 1,
     workload: 2,
     address: 'Улица Попова',
     type: 'перекрёсток',
     ways: [
      ObjectId("635a95381cc85f50234036f4"),
   ObjectId("635a95391cc85f50234036f7")
     ]
```

```
  },
  {
     _id: ObjectId("635a95391cc85f50234036f7"),
     start: { latitude: 59.9726873, longitude: 30.3227888 },
     end: { latitude: 59.9732595, longitude: 30.3164684 },
     capacity: 1,
     car_count: 1,
     workload: 10,
     address: 'Инструментальная улица',
     type: 'улица',
     ways: [ ObjectId("635a95391cc85f50234036f6") ]
  }
]
# db.routes.find()
[
  {
     _id: ObjectId("635a96fb1cc85f50234036f8"),
     length: 21.68,
     time: 17.81,
     points: [
     { latitude: 59.978279, longitude: 30.3271163 },
      { latitude: 59.9813885, longitude: 30.3214304 },
     { latitude: 59.9726873, longitude: 30.3227888 }
     ],
     workloads: [ 8, 5 ]
  },
  {
     _id: ObjectId("635a96fb1cc85f50234036f9"),
     length: 2.6,
     time: 10,
     points: [
     { latitude: 59.9726873, longitude: 30.3227888 },
     { latitude: 59.9732595, longitude: 30.3164684 },
     { latitude: 59.9726873, longitude: 30.3227888 }
     ],
     workloads: [ 2, 10 ]
  }
]
```

РАЗРАЗБОТАННОЕ ПРИЛОЖЕНИЕ

Разработанное приложение состоит из одной страницы, на которой отображаются карта, настройки для симуляции и фильтрации, таблица с полученными данными.

В приложении реализованы следующие возможности:

- Построение маршрута по двум заданным пользователем точкам;
- Настройка фрагмента дорожной сети для симуляции;
- Фильтрация полученных данных в результате симуляции;
- Просмотр конкретного маршрута на карте.

Снимки экранов приложения представлены в приложении Д.

Использованные технологии:

- OSM;
- Mongo;
- Python;
- JavaScript;
- CSS, HTML, React;
- Docker и Docker-compose.

Документация по сборке и развертыванию приложения представлена в приложении В.

Инструкция для пользователя представлена в приложении Г.

Снимки экрана приложения представлены в приложении Д.

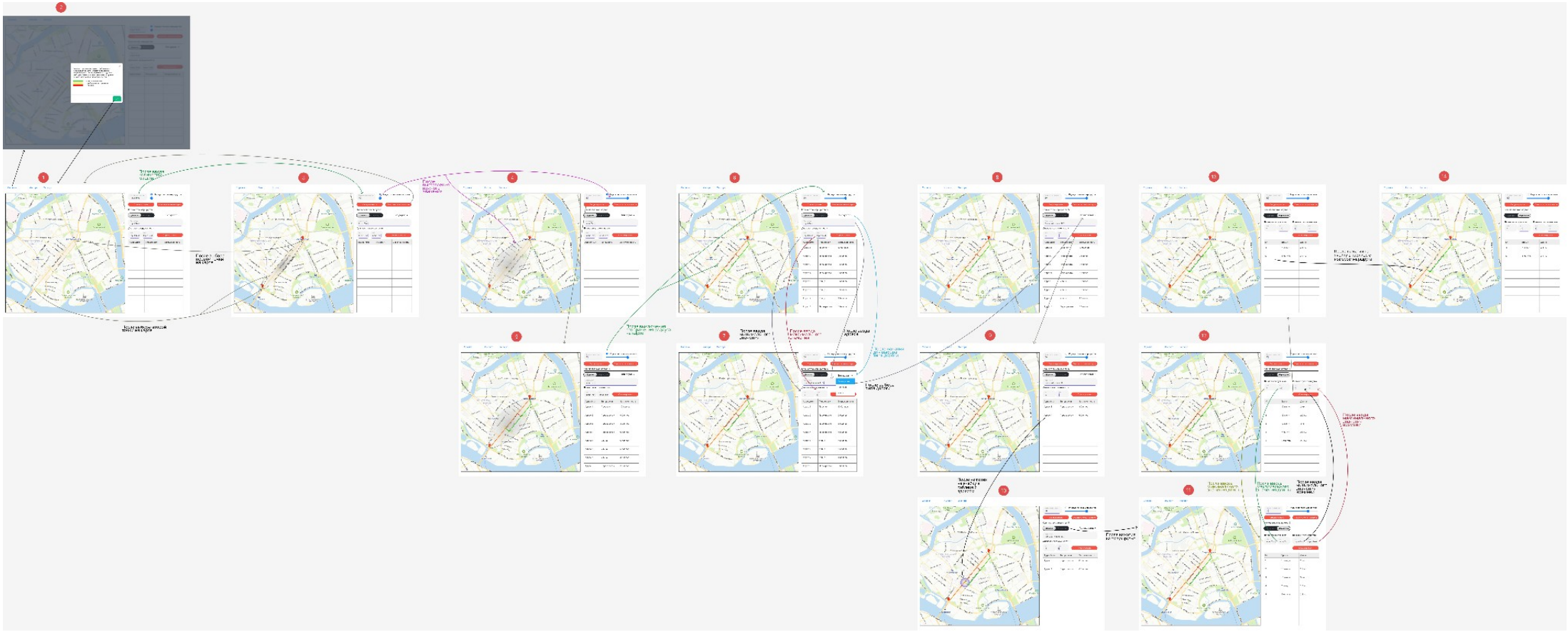
ВЫВОДЫ

Было разработано приложение для симуляции загруженности фрагмента дорожной сети. Для взаимодействия с инструментом был реализован веб-интерфейс. Данные для карты загружаются из сервиса Open Street Maps (OSM). Также была реализована возможность импорта и экспорта данных в машиночитаемом формате JSON.

Разработанное приложение содержит следующие недостатки: приложение не учитывает физическое состояние дорог и крутость поворотов, а также не учитывает режимы работы светофоров.

Дальнейшим решением может быть реализация приложения для мобильных устройств.

ПРИЛОЖЕНИЕ А



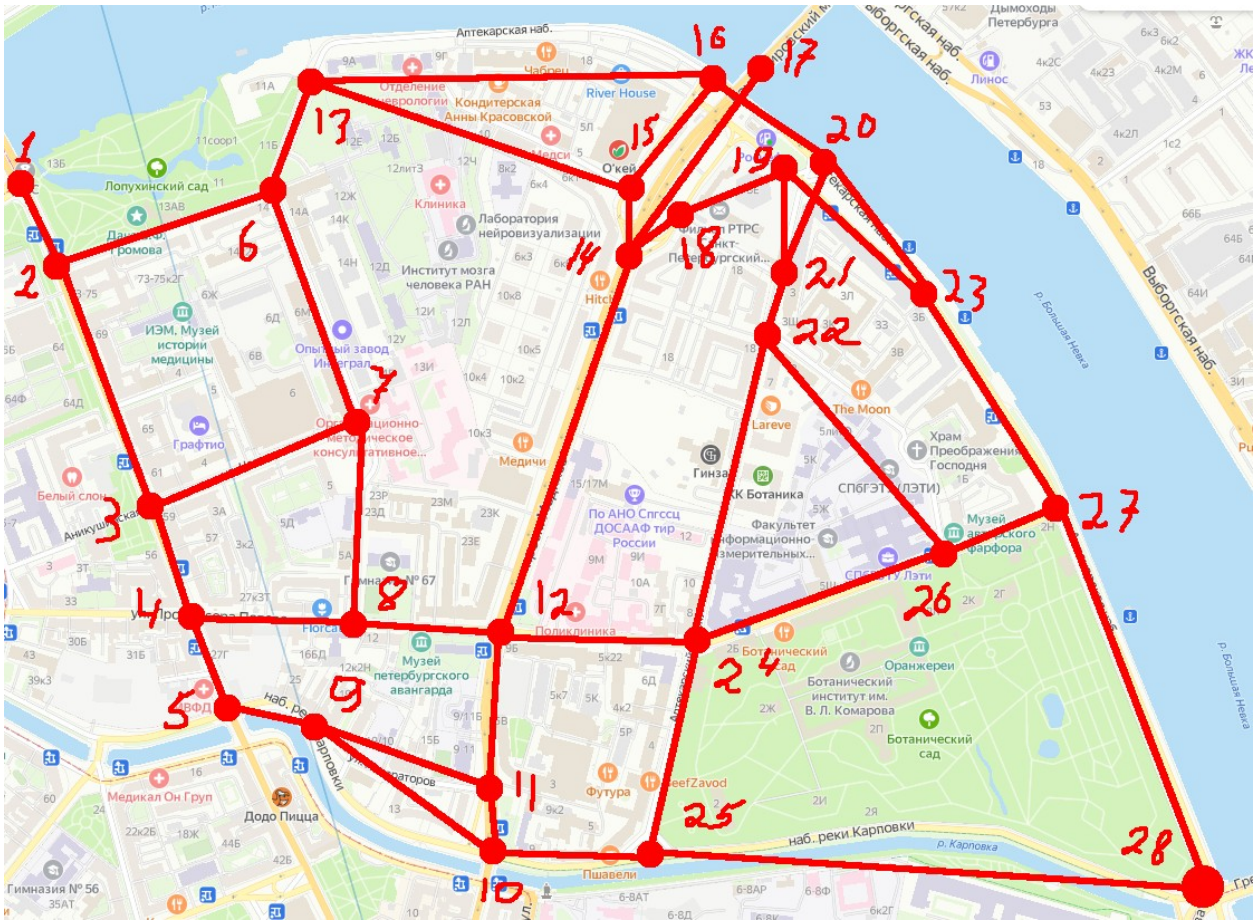
ПРИЛОЖЕНИЕ Б

Расчёт зависимости количества маршрутов от количества дорог.

Для оценки зависимости количества маршрутов (M) от количества дорог (N) возьмём дорожный граф около ЛЭТИ. Упростим модель:

- Все дороги прямые (в реальности они извилистые и поэтому дорога без разветвлений разбивается на несколько);
- Все дороги двунаправленные.

Получаем следующий граф:



Поиск маршрутов будет осуществляться с некоторым ограничением: если есть в текущей точке есть путь до уже пройденной точки, то текущая точка не рассматривается. Например, путь 1-2-3-7-6 нельзя взять, потому что есть путь из 2 в 6, т.е. такой маршрут образует **явный** крюк, что водители в реальной жизни очень редко делают. А вот путь 1-2-3-4-8-12-14-13 будет

учитываться, так как это **неявный** крюк, хоть и есть более короткий путь 2-6-13.

На карте точками отмечены места пересечения дорог, а не сами дороги, потому что так проще искать маршруты, а на результат это не повлияет, так как маршрут и так состоит из точек. Как и ранее, предположим, что будут задействованы все дороги (т.е. радиус покрывает всю область карты), а маршруты ищутся из одного угла карты в другой угол: из узла 1 в узел 28. Для подсчёта количества путей использовалась программа count_ways.py. Результат: 66 путей. На рассмотренном графе 39 дорог (дорога, связывающая точку 17 не учитывается, потому что это тупик в данном графе).

Получаем, $что \frac{M}{N} = \frac{66}{39} = 1.7 = ¿ M = 1.7 \cdot N$.

ПРИЛОЖЕНИЕ В

Документация по сборке и развертыванию приложения.

Собрать приложение командой *docker-compose build;*

Запустить приложение командой *docker-compose up.*

ПРИЛОЖЕНИЕ Г

Инструкция для пользователя.

При помощи кликов указать точки на карте, которые будут служить пунктами убытия и прибытия для маршрута. В поле “Количество машин” ввести то количество машин, которое пользователь хочет пустить по дорогам маршрутов. Выставить нужное значение радиуса поиска маршрутов на слайдере. Далее нажать на кнопку “Смоделировать”. После этого на карте появятся возможные маршруты с отображением степени загруженности дорог, в таблице появятся данные о загруженности каждой дороги маршрутов, также появится информация о количестве маршрутов. Если переключиться на “Маршруты”, то в таблице будет информация о маршрутах: номер маршрута, время его прохождения и его длина.

Чтобы отфильтровать полученные данные, нужно в поля ввести данные: в поле “Адрес/имя” ввести адрес дороги; в поле “min” ввести минимальное значение диапазона загруженности; в поле “max” ввести максимальное значение диапазона загруженности; в выпадающем меню “Тип дорог” выбрать нужный тип дороги. После этого нажать на кнопку “Отфильтровать”. В таблице ниже появятся отфильтрованные значения.

Для фильтрации маршрутов необходимо: в поле “min” для длины маршрута ввести минимальное значение длины маршрута в км; в поле “max” для длины маршрута ввести максимально значение длины маршрута в км; в поле “min” для времени пути ввести минимальное значение времени в минутах; в поле “max” для времени пути ввести максимальное значение времени в минутах. После этого нажать на кнопку “Отфильтровать”. В таблице ниже появятся отфильтрованные значения в порядке убывания времени в пути.

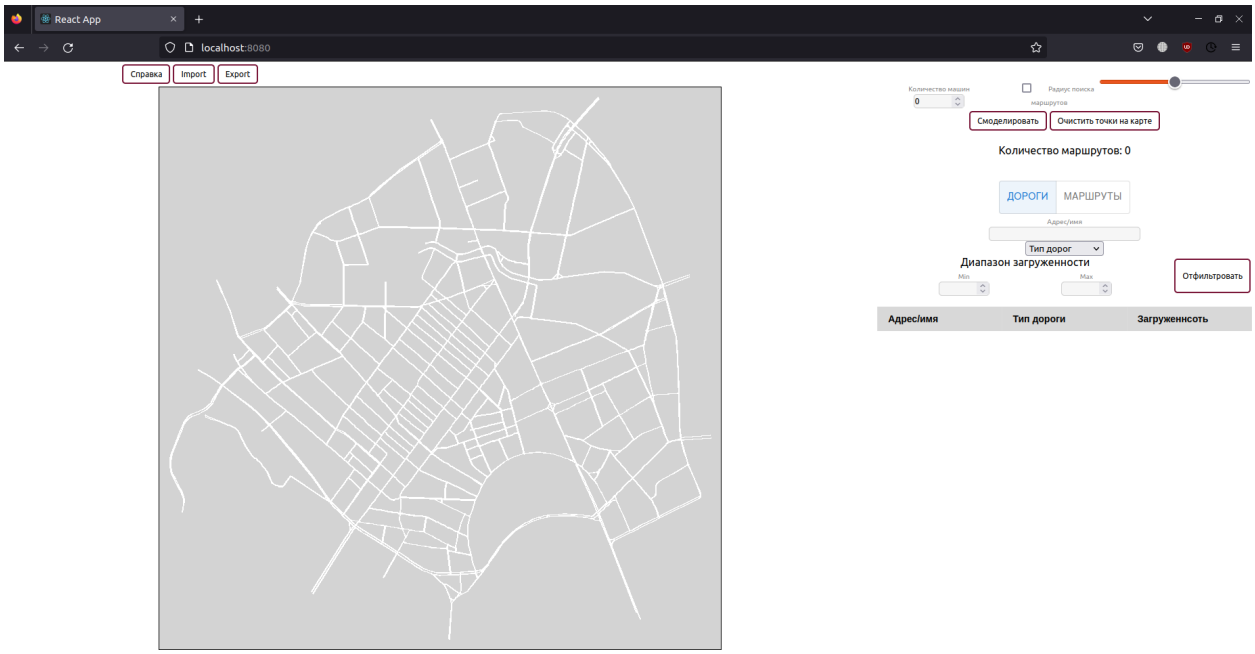
Для импорта необходимо нажать на кнопку “Импорт” и в появившемся диалоговом окне выбрать выбрать файл с данными в машиночитаемом формате JSON. Затем на кнопку “Импортировать” и дождаться окончания

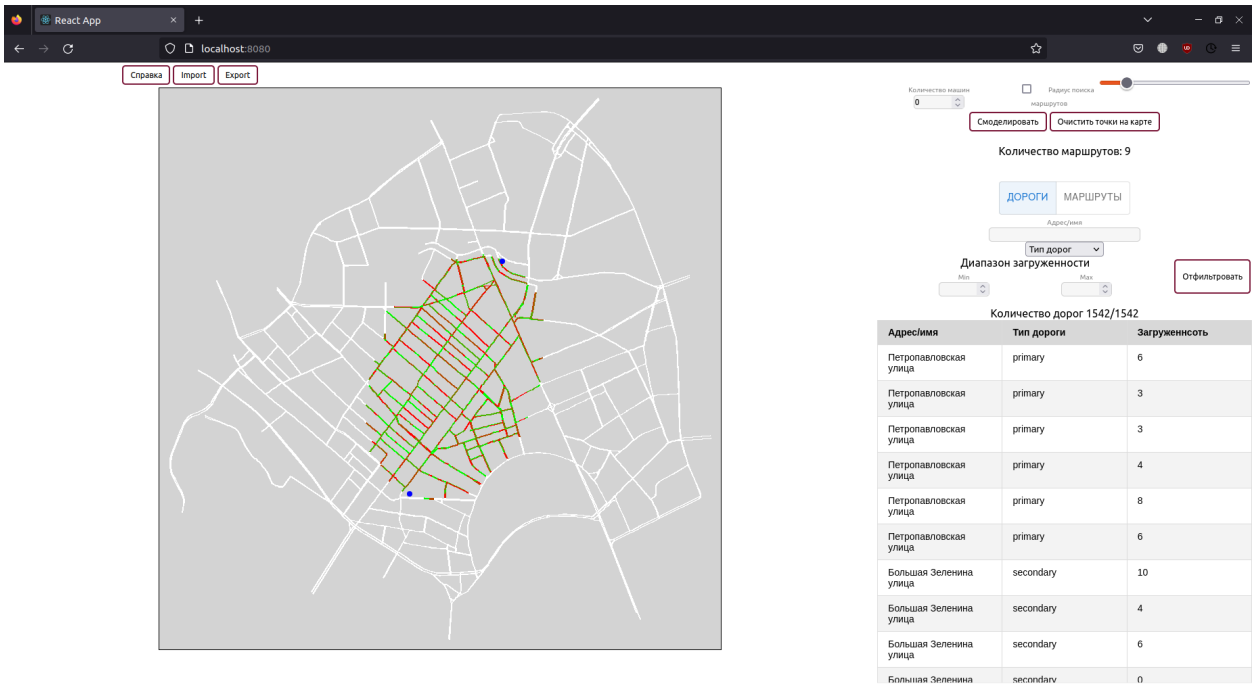
импорта данных. При успешном импорте данных диалоговое окно закроется и отобразятся новые данные. При ошибке во время импортирования появится сообщение с ошибкой.

Для экспорта необходимо нажать на кнопку “Экспорт” и выбрать место для сохранения файла с данными в машиночитаемом формате JSON.

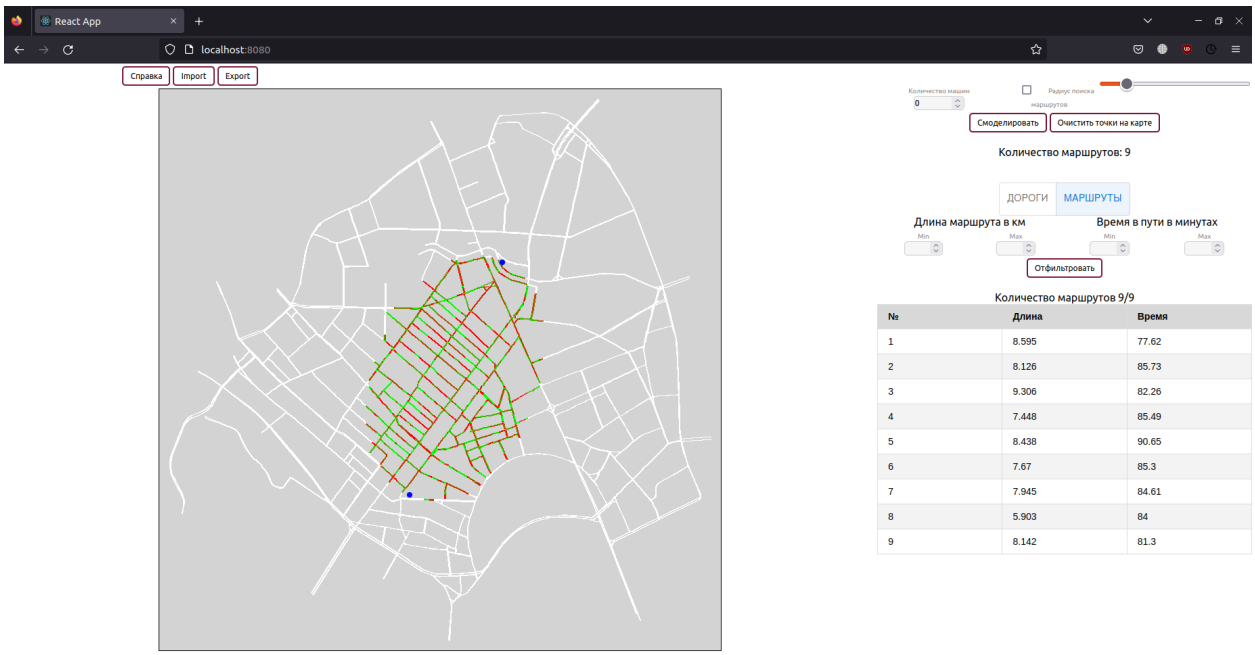
ПРИЛОЖЕНИЕ Д

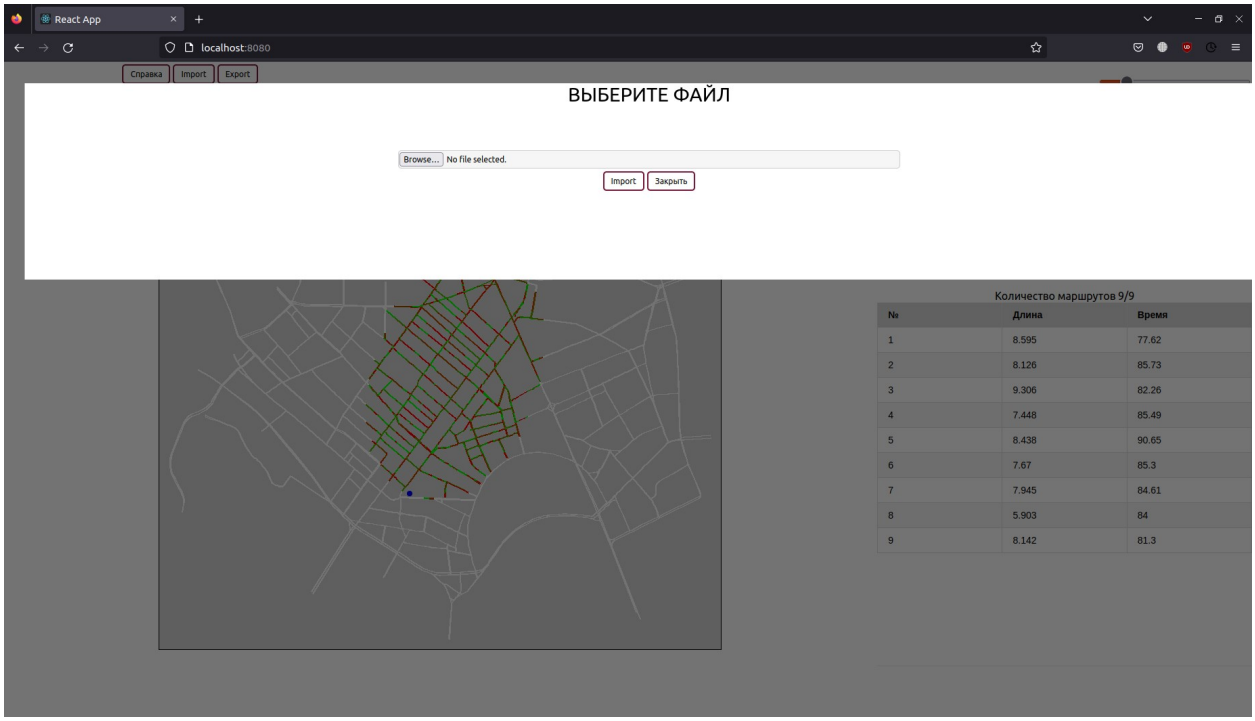
Снимки экрана приложения





Адрес/имя	Тип дороги	Загруженнсоть
Петропавловская улица	primary	6
Петропавловская улица	primary	3
Петропавловская улица	primary	3
Петропавловская улица	primary	4
Петропавловская улица	primary	8
Петропавловская улица	primary	6
Большая Зеленина улица	secondary	10
Большая Зеленина улица	secondary	4
Большая Зеленина улица	secondary	6
Большая Зеленина	secondary	0


React App
localhost:8080
Справка
Import
Export
Количество машин
0
Радиус поиска маршрутов
Смоделировать
Очистить точки на карте
Количество маршрутов: 9
ДОРОГИ
МАРШРУТЫ
Длина маршрута в км
Время в пути в минутах
Min
Max
Отфильтровать
Количество маршрутов 9/9
№ Длина Время
1 8.595 77.62
2 8.126 85.73
3 9.306 82.26
4 7.448 85.49
5 8.438 90.65
6 7.67 85.3
7 7.945 84.61
8 5.903 84
9 8.142 81.3


React App
localhost:8080
Справка
Import
Export
ВЫБЕРИТЕ ФАЙЛ
Browse...
No file selected.
Import
Закрыть
Количество маршрутов 9/9
№ Длина Время
1 8.595 77.62
2 8.126 85.73
3 9.306 82.26
4 7.448 85.49
5 8.438 90.65
6 7.67 85.3
7 7.945 84.61
8 5.903 84
9 8.142 81.3