

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

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ
по дисциплине «Разработка программного обеспечения информационных
систем»
Тема: Сопоставление разнородных наборов открытых (гео) данных -
самые людные улицы

Студентка гр. 5382	_____	Васильева А.Е.
Студентка гр. 5382	_____	Коппель Т.С.
Студентка гр. 5382	_____	Петрова Т.А.
Преподаватель	_____	Заславский М.М.

Санкт-Петербург
2018

ЗАДАНИЕ НА ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

Студенты Васильева А.Е., Коппель Т.С., Петрова Т.А.

Группа 5382

Тема проекта: Сопоставление разнородных наборов открытых (гео) данных -
самые людные улицы

Исходные данные:

Проект должен быть разработан с использованием базы данных Neo4j

Содержание пояснительной записки:

Содержание, Введение, Качественные требования к решению, Сценарии использования, Модель данных, Разработанное приложение, Заключение, Список использованных источников, Приложение А “Документация по сборке и развертыванию приложения”, Приложение В “Инструкция для пользователя”, Приложение С “Снимки экрана приложения”.

Предполагаемый объем пояснительной записки:

Не менее 20 страниц.

Дата выдачи задания: 13.09.2018

Дата сдачи реферата: 26.12.2018

Дата защиты реферата: 26.12.2018

Студентка гр. 5382	_____	Васильева А.Е.
Студентка гр. 5382	_____	Коппель Т.С.
Студентка гр. 5382	_____	Петрова Т.А.
Преподаватель	_____	Заславский М.М.

АННОТАЦИЯ

В курсовом проекте реализовано веб-приложение на основе СУБД Neo4j для сопоставления разнородных наборов открытых (гео) данных - самых людных улиц. Приложение позволяет определять людность улиц, отрисовывая их на карте разными цветами.

SUMMARY

The course project implemented a web application based on the Neo4j database to compare heterogeneous sets of open (geo) data - the most crowded streets. The application allows you to determine the density of streets, drawing them on the map in different colors.

СОДЕРЖАНИЕ

Введение	5
1. Качественные требования к решению	6
2. Сценарии использования	7
2.1. Макеты UI	7
2.2. Сценарии использования	8
2.3. Вывод.....	9
3. Модель данных	10
3.1. Нереляционная модель данных.....	10
3.2. Модель данных для SQL СУБД	13
3.3 Сравнение моделей.....	16
4. Разработанное приложение	17
4.1. Краткое описание	17
4.2. Используемые технологии	17
4.3. Ссылки на приложение	17
Заключение.....	18
Список использованных источников.....	19
Приложение А. Документация по сборке и развертыванию приложения.....	20
Приложение В. Инструкция для пользователя.....	21
Приложение С. Снимки экрана приложения	22

ВВЕДЕНИЕ

Для прогулки или знакомства с городом важно выбрать подходящий район. Например, вы хотите выбрать популярное место, с отметкой которого постоянно выкладывают фотографии, или, наоборот, тихое и спокойное.

Целью проекта является разработка приложения, с помощью которого можно определять людность улиц. Людность зависит от числа публикаций в Instagram, а результатом является карта, на которой отрисовываются улицы разными цветами. По цвету улицы можно понять о ее людности: красный – очень людная, желтая – средне людная, зеленая – нелюдная.

В проекте разработано веб-приложение на основе СУБД Neo4j с использованием данных OpenStreetMap и Instagram.

1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Необходимо разработать веб-приложение, позволяющее определять людность улиц.

Основные функции:

- Определение людности улицы;
- Отрисовка улицы в соответствии с ее людностью
- Импорт и экспорт данных.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. Макеты UI

Главная страница (рис.1).

Главная страница: людность улиц


Выберите улицу:

Улица	
1	Первая
2	Вторая

Рисунок 1 “Макет главной страницы”

Страница с картой (рис.2).

Самые людные улицы



Топ-10 людных улиц этого района:

1. Первая улица
...
10. Десятая улица

Рисунок 2 “Макет страницы с картой”

2.2. Сценарии использования

Для задачи импорта данных:

1. Пользователь находится на главной странице
2. Пользователь нажимает на кнопку «Выберите файл»
3. Пользователь в появившемся окне выбирает файл, содержащий данные для БД, в формате .graphml*
4. Пользователь нажимает на кнопку «Открыть»
5. Пользователь нажимает на кнопку «Импорт»
6. Пользователь видит главную страницу сайта с обновленными данными

*Файл должен иметь следующую структуру:

- Для Node:

```
<node id="n1335" labels=":Node"><data key="labels":Node</data><data key="lon">30.285741806030273</data><data key="id_node">1240250230</data><data key="lat">59.9754524230957</data></node>
```

- Для Way:

```
<node id="n1450" labels=":Way"><data key="labels":Way</data><data key="id_way">4406425</data><data key="name_street">улица Мира</data></node>
```

Для задачи анализа данных:

Сценарий №1

1. Пользователь находится на главной странице
2. Пользователь вводит название улицы в строку для ввода
3. Пользователь нажимает кнопку «Найти»
4. Пользователь видит вторую страницу с картой, на которой выбранная улица отрисована определенным цветом

Сценарий №2

1. Пользователь находится на главной странице
2. Пользователь выбирает улицу из списка и нажимает на ее название
3. Пользователь видит вторую страницу с картой, на которой выбранная улица отрисована определенным цветом

Сценарий №3

1. Пользователь находится на главной странице
2. Пользователь нажимает на кнопку «Все улицы»
3. Пользователь видит вторую страницу с картой, на которой отрисованы все улицы из БД

Сценарий №4

1. Пользователь находится на главной странице

2. Пользователь сортирует строки в столбцах таблицы, нажав на название нужного столбца
3. Пользователь видит таблицу с упорядоченными строками по названию улицы или по количеству публикаций

Сценарий №5

4. Пользователь находится на второй странице с картой
5. Пользователь нажимает на кнопку «Вернуться на первую страницу»
6. Пользователь видит главную страницу

Для задачи экспорта данных:

1. Пользователь находится на главной странице
2. Пользователь нажимает на кнопку «Экспорт»
3. Пользователь видит скаченный файл в формате .graphml в папке загрузок

2.3. 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. Нереляционная модель данных

Графическое представление

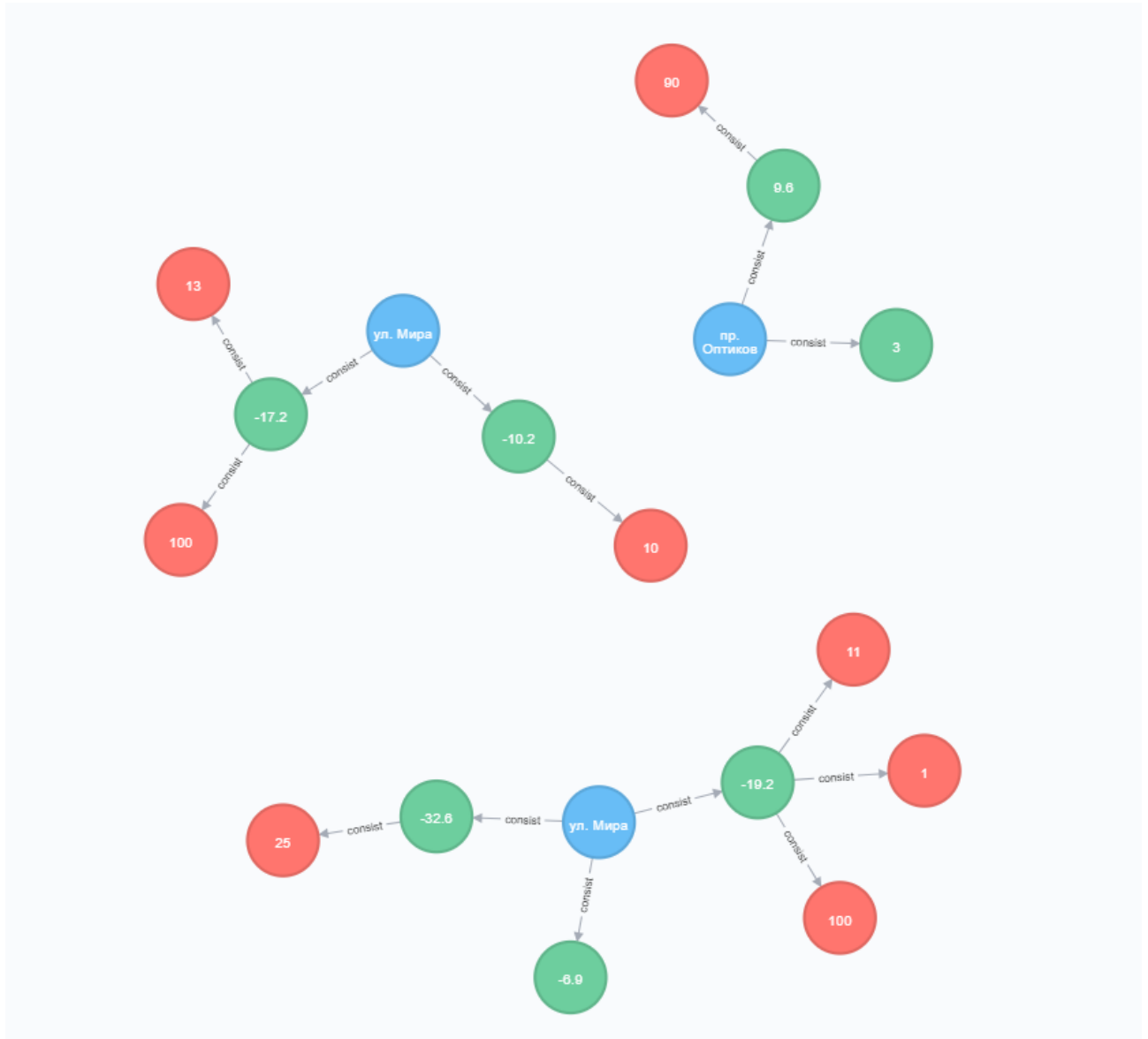


Рисунок 3 “Графическое представление нереляционной модели данных”

Описание назначений коллекций, типов данных и сущностей

Модель данных включает в себя сущности трех типов - Way, Node, Id_location. Улица состоит из одного или нескольких путей (Way) - голубого цвета в графе. Путь состоит в свою очередь из нескольких точек (Node), в количестве от 2 до 2000 - зеленого цвета в графе. Эти данные берутся из OpenstreetMap. Точка содержит нужные координаты для поиска соответствующих этим координатам location_id в Инстаграме, а по location_id

можно узнать о публикациях и их количестве. Таким образом, Node имеет связи с Id_location, которые красного цвета в графе.

Источник данных: Instagram, OpenStreetMap.

Структура данных: данные представлены в виде направленного графа, вершинами которого являются сущности типа Way, от которых идут связи к вершинам типа Node, а от них - к вершинам типа Id_location.

Описание типов используемых данных для сущности Way:

- "id_way", int - id пути
- "name_street", String - название улицы

Описание типов используемых данных для сущности Node:

- "id_node", int - id точки
- "lat", int - широта
- "lon", int - долгота

Описание типов используемых данных для сущности Id_location:

- "id_location", int - id локации
- "count_pub", int - количество публикаций для данного места

Оценка удельного объема информации, хранимой в модели (сколько потребуется памяти, чтобы сохранить объекты, как объем зависит от количества объектов)

Оценка удельного объема информации, хранимой в Way:

- "id_way", int - 4 байта
- "name_street", String - средняя длина названия улицы 10 символов, ~10 байт

Оценка удельного объема информации, в сущности Node:

- "id_node", int - 4 байта
- "lat", float - 4 байта
- "lon", float - 4 байта

Описание типов используемых данных для сущности Id_location:

- "id_location", int - 4 байта
- "count_pub", int - 4 байта

Итог: $((4+10)*M + (4+4+4)*L*M + (4+4)*K*L*M)*N$, где N - количество улиц, M - среднее количество путей, из которых состоит улица, L - среднее

количество точек, из которых состоит путь, K - среднее количество публикаций для `location_id` из Инстаграма * среднее количество `location_id` для одной точки. Пусть $M = 12$, $L = 8$ (данные взяли из openStreetMap и подсчитали среднее значение, исходя из случайной выборки 20 улиц и 20 отрезков). БД будет занимать: $(14*12 + 12*8*12 + 8*K*12*8)*N = (1320 + 768*K)*N$ байт.

Запросы к модели, с помощью которых реализуются сценарии использования

- Запрос на создание экземпляра сущности Way
CREATE (a:Way {id:11, name:"ул. Мира"})
- Запрос на создание экземпляра сущности Node
CREATE (d:Nod {id:111, lat:124.6, lon:-17.2})
- Запрос на создание экземпляра сущности Id_location
CREATE (n:Instagram_location {id:124342, count_public:100})
- Запрос на создание связи между сущностями Way и Node
CREATE (a)-[:consist]->(d)
- Запрос на создание связи между сущностями Node и Id_location
CREATE (d)-[:has]->(n)
- Запрос на вывод количества публикаций по названию улицы
MATCH (n:Way {name:"ул. Мира"})-[:consist]->(m:Nod), (m:Nod)-[:consist]->(r:Instagram_location) RETURN sum(r.count_public)
- Запрос на сортировку улиц по людности
MATCH (n:Way)-[:consist]->(m:Nod), (m:Nod)-[:consist]->(r:Instagram_location) RETURN n.name, sum(r.count_public) AS countAll ORDER BY countAll DESC

Потребуется $N*M$ запросов для создания экземпляров сущности Way, $N*M*L$ - для создания экземпляров сущности Node и $N*M*L*K$ запросов для создания экземпляров сущности Instagram_location. Также потребуется создать связи между сущностями: $N*M*L + N*M*L*K$, где N - количество улиц, M - среднее количество путей, из которых состоит улица, L - среднее количество точек, из которых состоит путь, K - среднее количество публикаций для `location_id` из Инстаграма * среднее количество `location_id` для одной точки. Пусть $M = 12$, $L = 8$ (данные взяли из openStreetMap и подсчитали среднее

значение, исходя из случайной выборки 20 улиц и 20 отрезков). Будет 1 раз использоваться запрос сортировки для создания рейтинга улиц и S запросов для вывода количества публикаций, где S - количество запросов пользователя. Итог: $204*N + 192*N*K + S + 1$.

3.2. Модель данных для SQL СУБД

Графическое представление

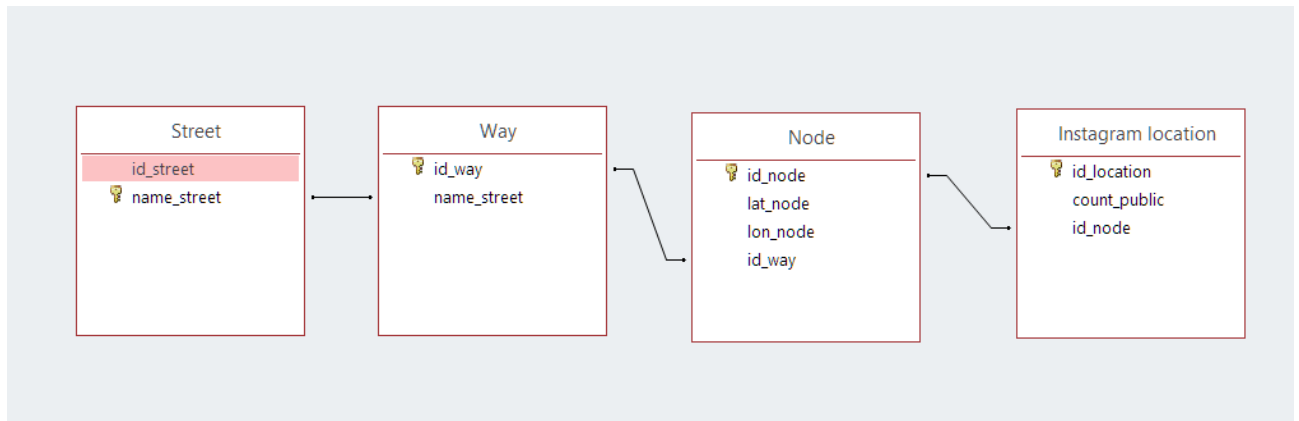


Рисунок 4 “Графическое представление реляционной модели данных”

Описание назначений коллекций, типов данных и сущностей

Реляционная модель БД состоит из таблиц:

- Street
- Way
- Node

Instagram location

Таблица Street состоит из атрибутов:

- id_street - id улицы
- name_street (ключ) - название улицы

Таблица Way состоит из атрибутов:

- id_way (ключ) - id пути
- name_street - название улицы

Таблица Node состоит из атрибутов:

- id_node (ключ) - id точки
- lat_node - широта
- lon_node - долгота
- id_way - id пути

Таблица Instagram location состоит из атрибутов:

- id_location (ключ) - id окации
- count_public - количество публикаций
- id_node - id точки

Оценка удельного объема информации, хранимой в модели (сколько потребуется памяти, чтобы сохранить объекты, как объем зависит от количества объектов)

Таблица Street:

- id_street, длинное целое, 4 байта
- name_street, средняя длина названия улицы 10 символов, ~10 байт

Таблица Way:

- id_way, длинное целое, 4 байта
- name_street, средняя длина названия улицы 10 символов, ~10 байт

Таблица Node состоит из атрибутов:

- id_node, длинное целое, 4 байта
- lat_node, двойное с плавающей точкой, 8 байт
- lon_node, двойное с плавающей точкой, 8 байт

id_way, длинное целое, 4 байта

Таблица Instagram location состоит из атрибутов:

- id_location, длинное целое, 4 байта
- count_public, длинное целое, 4 байта
- id_node, длинное целое, 4 байта

Итог: $(4+10)*N + (4+10)*M*N + (4+8+8+4)*L*M*N + (4+4+4)*K*L*M*N$,
где N - количество улиц, M - среднее количество путей, из которых состоит улица, L - среднее количество точек, из которых состоит путей, K - среднее количество публикаций для location_id из Инстаграма * среднее количество location_id для одной точки. Пусть M = 12, L = 8 (данные взяли из openStreetMap и подсчитали среднее значение, исходя из случайной выборки 20 улиц и 20 отрезков). БД будет занимать: $(14 + 14*12 + 24*8*12 + 12*K*8*12)*N = (2472 + 1152*K)*N$ байт.

Запросы к модели, с помощью которых реализуются сценарии использования

- Запрос на добавление объекта сущности Street:
`INSERT INTO Street VALUES('id_street', 'name_street');`
- Запрос на добавление объекта сущности Way:
`INSERT INTO Way VALUES('id_way', 'name_street');`
- Запрос на добавление объекта сущности Node:
`INSERT INTO Node VALUES('id_node', 'lat_node', 'lon_node', 'id_way');`
- Запрос на добавление объекта сущности Instagram location:
`INSERT INTO Instagram location VALUES('id_location', 'name_location', 'lat_location', 'lon_node', 'count_public', 'id_node');`
- Запрос для вывода количества публикаций для конкретной улицы:
`SELECT Street.name_street, Sum([Instagram location].count_public) AS SumOfcount_public FROM ((Street INNER JOIN Way ON Street.name_street = Way.name_street) INNER JOIN Node ON Way.id_way = Node.id_way) INNER JOIN [Instagram location] ON Node.id_node = [Instagram location].id_node GROUP BY Street.name_street;`
- Запрос на сортировку улиц по количеству публикаций:
`SELECT Street.name_street, Sum([Instagram location].count_public) AS SumOfcount_public FROM ((Street INNER JOIN Way ON Street.name_street = Way.name_street) INNER JOIN Node ON Way.id_way = Node.id_way) INNER JOIN [Instagram location] ON Node.id_node = [Instagram location].id_node GROUP BY Street.name_street ORDER BY Sum([Instagram location].count_public) DESC;`

Потребуется N запросов для заполнения таблицы Street. $N * M$ запросов для заполнения таблицы Way, $N * M * L$ - для заполнения таблицы Node и $N * M * L * K$ заполнения таблицы Instagram_location, где N - количество улиц, M - среднее количество путей, из которых состоит улица, L - среднее количество точек, из которых состоит путей, K - среднее количество публикаций для location_id из Инстаграма * среднее количество location_id для одной точки. Пусть $M = 12$, $L = 8$ (данные взяли из openStreetMap и подсчитали среднее значение, исходя из случайной выборки 20 улиц и 20 отрезков). Будет 1 раз использоваться запрос сортировки для создания рейтинга улиц и S запросов для вывода количества публикаций, где S - количество запросов пользователя. Итог: $N + N * M + N * M * L$

$$+ N * M * L * K + S + 1 = N + N * 12 + N * 12 * 8 + N * 12 * 8 * K + S + 1 = 108 * N + 96 * N * K + S + 1.$$

3.3 Сравнение моделей

Удельный объем информации - где данные занимают больший объем при прочих равных

Нереляционная модель данных будет занимать $(1320 + 768 * K) * N$ байт. А SQL-модель - $(2472 + 1152 * K) * N$ байт. Т.е. нереляционной модели данных потребуется меньше памяти.

Запросы по отдельным юзкейсам

Запросы и для вывода информации о количестве публикаций, и для сортировки улиц для составления рейтинга самых людных улиц более простые и компактные.

Количество запросов для совершения юзкейсов в зависимости от числа объектов в БД и прочих параметров

В нереляционной модели данных потребуется $204 * N + 192 * N * K + S + 1$ запросов, в SQL-модели: $108 * N + 96 * N * K + S + 1$.

Вывод, что лучше

SQL-модель данных лучше по количеству запросов, но проигрывает нереляционной модели по простоте этих запросов и требуемому размеру памяти.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Краткое описание

Разработанное приложение определяет людность для улицы или набора улиц и отрисовывает улицу на карте соответствующим цветом. Приложение состоит из страниц:

1. Главная страница. На данной странице пользователь может задать улицу для отрисовки, а также загрузить/выгрузить данные БД.
2. Страница с отображением заданной улицы или всех улиц, информация о которых находится в БД. Пользователь на карте видит отрисованную определенным цветом улицу.

4.2. Используемые технологии

При написании приложения использовались следующие технологии:

- Java 8;
- Neo4j;
- Maven;
- OpenStreetMap API;
- Instagram API;
- Leafletjs
- HTML;
- CSS;
- Java Script.

4.3. Ссылки на приложение

Исходный код приложения и инструкция по установке находятся по ссылке:

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

ЗАКЛЮЧЕНИЕ

Разработано приложение для определения людности улиц Петроградского района Санкт-Петербурга с использованием базы данных Neo4j на основе данных OpenStreetMap и Instagram.

В последующем планируется добавить данные всего Петербурга и области.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Java Platform Standard Edition 8 Documentation: <https://docs.oracle.com/javase/8/docs/> (дата обращения: 19.12.2018).
2. The Neo4j Cypher Manual v3.5: <https://neo4j.com/docs/cypher-manual/current/> (дата обращения: 19.12.2018).
3. Instagram API: <https://www.instagram.com/developer/> (дата обращения: 19.12.2018).
4. OpenStreetMap API: <https://wiki.openstreetmap.org/wiki/API> (дата обращения: 19.12.2018).
5. Leafletjs API: <https://leafletjs.com/reference-1.3.4.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 Neo4j.
4. Запустить базу данных Neo4j.
5. Запустить jar-файл.
6. Перейти в браузере по адресу: <http://localhost:8080/Servlet>.

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 «Выбрать файл» и, выбрав файл в открывшемся окне, нажать кнопку «Импорт».

Для отрисовки улицы пользователь должен ввести название улицы и нажать кнопку «Найти», также пользователь может выбрать название улицы из таблицы или нажать на кнопку «Все улицы» для отрисовки всех улиц.

Для более удобного поиска улицы в таблице пользователь может отсортировать значения строк таблицы по названию улицы, либо по количеству публикаций, нажав на название соответствующего столбца.

Пользователь может получить данные из БД, нажав на кнопку «Экспорт» на главной странице.

Страница с картой

После выбора улицы пользователь попадает на страницу с картой, на которой отрисована выбранная улица, либо все улицы, информация о которых находится в БД. По цвету улицы можно понять о людности улицы: красный – очень людная, желтая – средне людная, зеленая – нелюдная.

По кнопке «Вернуться на первую страницу» пользователь может попасть на главную страницу.

На рисунках 4-6 изображены снимки экрана приложения.

Все улицы на карте сразу: [Все улицы](#)

Введите название улицы или выберите его из таблицы:

Название улицы	Количество публикаций
Каменноостровский проспект	226
Южная дорога	157
Большой проспект П.С.	147
Малый проспект П.С.	128
Петровский проспект	105
2-я Берёзовая аллея	98
Большая Пушкарская улица	85
улица Профессора Попова	84
Петроградская набережная	77
проспект Меликов	71
Пионерская улица	70
Песочная набережная	68
Спортивная улица	67
Северная дорога	64
Аптекарская набережная	64
Левантовский проспект	62
улица Блохина	61
Морской проспект	60
улица Чапаева	60

Работа с БД

Для экспорта нажмите:

Для импорта нажмите:

Рисунок 4 “Главная страница”

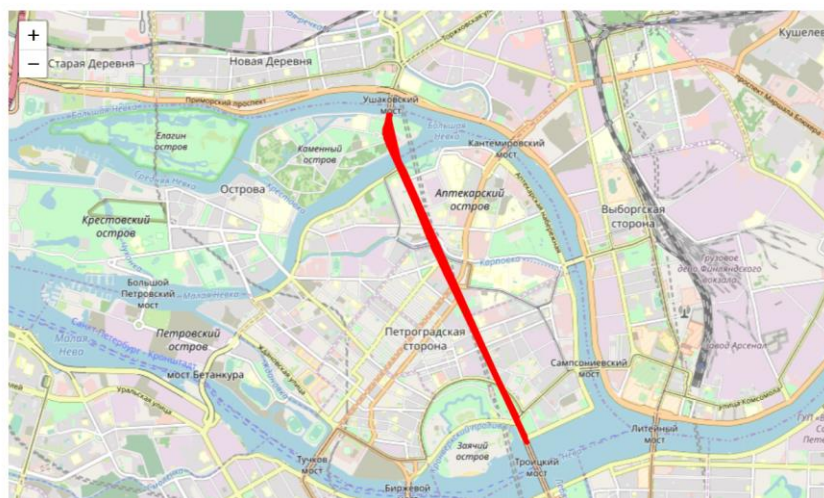
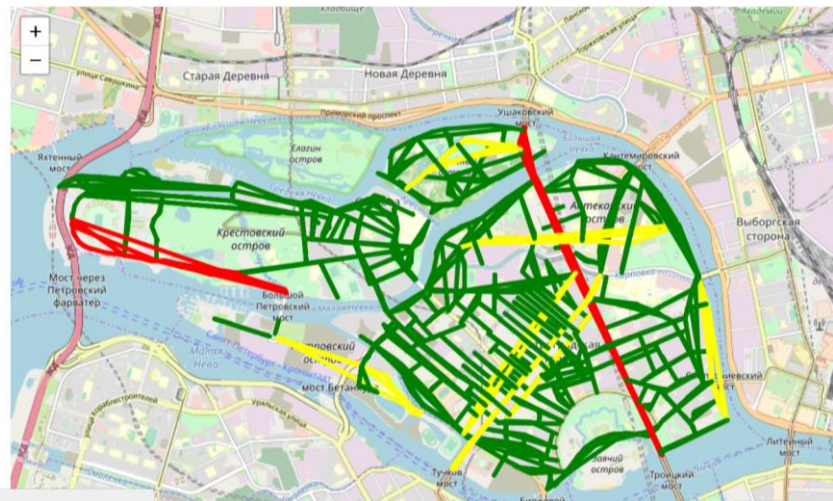
[Вернуться на первую страницу](#)

Рисунок 5 “Страница с картой при выборе одной улицы”

Crowdest Map

[Вернуться на первую страницу](#)



Идет обработка запроса

Рисунок 6 “Страница с картой при выборе всех улиц”