

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

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Введение в нереляционные базы данных»

Тема: Составление маршрутов интересных пеших прогулок по СПБ

Студенты гр. 0382

Андрющенко К.С.

Злобин А.С.

Ильин Д.А

Преподаватель

Заславский М.М.

Санкт-Петербург

2024

ЗАДАНИЕ

Студенты

Андрющенко К.С.

Злобин А.С.

Ильин Д.А.

Группа 0382

Тема задания: Составление маршрутов интересных пеших прогулок по СПБ

Исходные данные:

Задача - сделать сервис для составления пеших маршрутов по Санкт-Петербургу. Пользователи - администраторы, пользователи. Необходимые (но не достаточные фичи) - личные страницы, страницы маршрутов, страница генерации маршрута (задаем настройки - интересы, протяженность - учитываем время года, ширину тротуаров).

Содержание пояснительной записки:

«Содержание»

«Введение»

«Сценарий использования»

«Модель данных»

«Разработанное приложение»

«Заключение»

«Приложение А. Документация по сборке и развертыванию приложения»

Предполагаемый объем пояснительной записи:

Не менее 28 страниц.

Дата выдачи задания: 27.09.2023

Дата сдачи реферата: 01.02.2024

Дата защиты реферата: 01.02.2024

Студенты		Андрющенко К.С. Злобин А.С. Ильин Д.А
Преподаватель		Заславский М.М.

АННОТАЦИЯ

В рамках данного курса предполагалось разработать в команде приложение на одну из предложенных тем. Была выбрана тема создания приложения составление маршрутов интересных пеших прогулок по СПБ. Для выполнения задания предлагается использовать СУБД Neo4j. Найти исходный код и всю дополнительную информацию можно по ссылке:

<https://github.com/moevm/nosql2h23-spb-routes>

SUMMARY

Within this course, the task was to develop an application in a team based on one of the proposed topics. The chosen topic was to create an application for planning interesting walking routes in St. Petersburg. To accomplish this task, it is suggested to use the Neo4j database management system. The source code and any additional information can be found at the following link:

<https://github.com/moevm/nosql2h23-spb-routes>

СОДЕРЖАНИЕ

Введение	7
1. Сценарии использования	8
1.1. Макет UI	8
1.2. Сценарии использования для задачи	9
2. Модель данных	15
2.1. Нереляционная модель данных	15
2.2. Аналог модели данных для SQL СУБД	20
2.3. Сравнение моделей	25
3. Разработанное приложение	26
3.1. Краткое описание	26
3.2. Использованные технологии	26
3.3. Снимки экрана приложения	26
Заключение	33
Список использованных источников	34
Приложение А. Документация по сборке и развертыванию приложения	35

ВВЕДЕНИЕ

Цель работы – создать удобный сервис для составления пеших маршрутов по Санкт-Петербургу, который позволял бы генерировать маршруты, основываясь на предпочтениях пользователя.

Было решено разработать веб-приложение, которое позволит не только генерировать маршруты по заданным пользователем параметрам и сохранять их в своем профиле, но и создавать собственные места и пути прогулок, чтобы другие пользователи могли найти и посмотреть их.

1. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

1.1. Макет UI

Разработанный макет приложения доступен по ссылке:

<https://www.figma.com/file/8Darh3lKIUfuZ0P94lCeB0/Untitled?type=whiteboard&node-id=0%3A1&t=RHGJIusSpOab5ND6-1>

Из-за размеров (память и большое разрешение изображения) макета изображение поделено на 4 части:

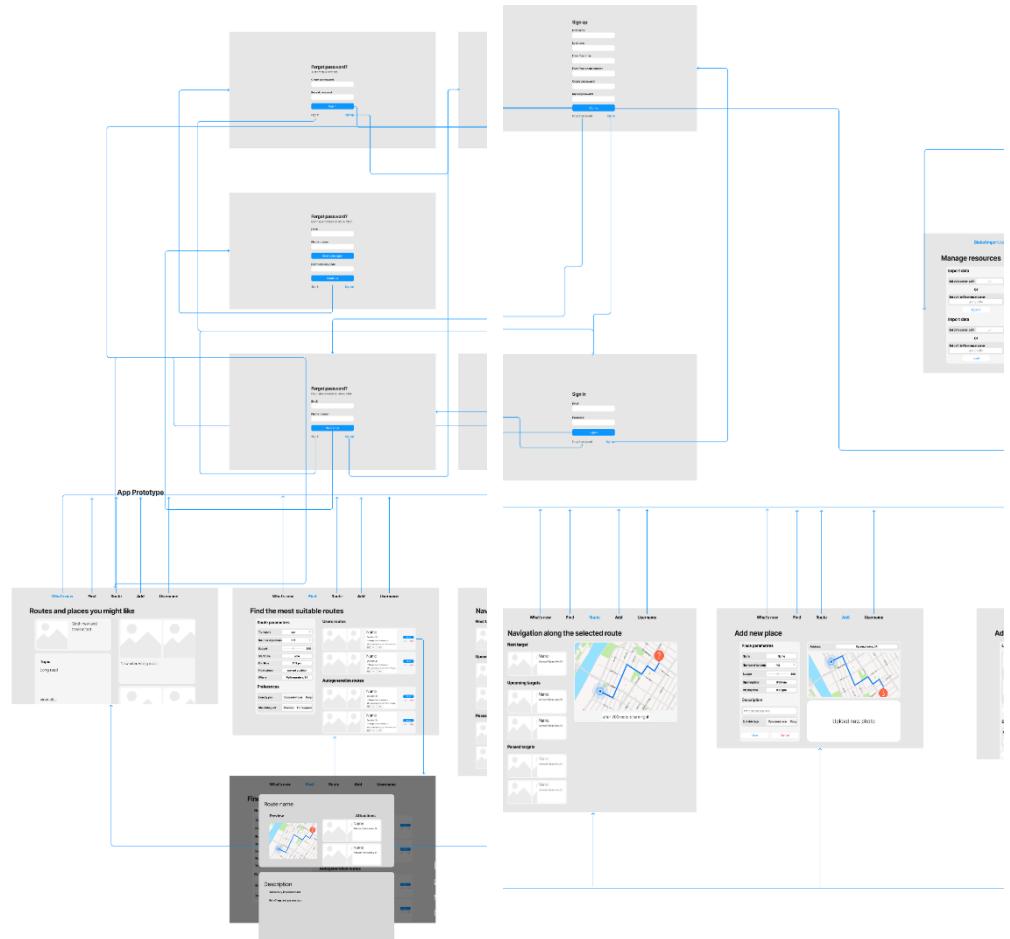




Рисунок 1. Макет UI

1.2. Сценарии использования для задачи

1. Сценарий использования - " поиск интересных мест ":

Основной сценарий:

- Пользователь вводит логин и пароль или создаёт новый аккаунт.
- На главной странице (вкладке) “What's new” - просмотр интересных записей или новостей.
- Вкладка “Find” - поиск интересующих маршрутов по параметрам.
- Вкладка “Route” - навигация по выбранному маршруту.
- Вкладка “Add” - добавление новых мест или маршрутов.
- Вкладка “User” - редактирование информации в профиле пользователя, включая изменения о добавленных маршрутах.

2. Сценарий использования - "Просмотр наиболее интересных записей или новостей (Routes and places you might like)":

- a. Пользователь после авторизации попадает на страницу “Routes and places you might like”, либо открывает ее с помощью кнопки “What's new” в навигационной панели, которая расположена вверху страницы.
 - b. При пролистывании страницы вниз подгружаются интересные для пользователя маршруты и места.
 - c. При нажатии либо на заголовок новости или на кнопку “view all...” открывается карточка с более подробной информацией об этой записи.
3. Сценарий использования - " Поиск интересующих маршрутов (Find the most suitable routes)":
- a. Страница открывается с кнопки “Find” в навигационной панели, которая расположена вверху страницы.
 - b. В левой части страницы находится список параметров маршрута “Route parameters” и список тэгов для выбора интересующих/или нет пользователя типов развлечений.
 - c. В списке параметров задается точка старта маршрута (графа “Or select your current position on map” с кнопкой “select on map”), по нажатию на кнопку открывается окно с картой на которой можно кликом выбрать текущее положение. Если пользователь определился со стартовой точкой, то по нажатию кнопки “Select” в параметрах маршрута отобразятся координаты выбранной точки.
 - d. Аналогичным образом работает выбор конечной точки. Наличие которой является необязательным. В таком случае маршрут будет строится по ближайшим к стартовой точке интересным местам.
 - e. Ползунок “Number of sights on the route” задает количество интересующих точек на маршруте.
 - f. В графе “Preferences” можно выбрать типы мест, от посещения которых пользователь не отказался (“Exactly yes”) и которые он точно не желает посещать (“Exactly no”).

- g. После нажатия кнопки “Find” справа от фильтров отображаются подходящие под параметры маршруты: сначала под заголовком “User routes” - пользовательские маршруты, а под заголовком “Autogeneration routes” - сгенерированные автоматически.
- h. На кнопку “view info...” открывается более подробная информация о маршруте, включая список входящих в него интересных мест с их названиями и координатами, на кнопку “Close” карточка маршрута закрывается.
- i. Лента изображений на карточке маршрута и на карточке интересных мест с подробной информацией о маршруте прокручивается вправо мышкой (Shift + scroll) или движением стилуса влево по сенсорному экрану или тачпаду.
- j. На кнопку “Start” выбранный маршрут открывается в навигаторе (вкладка “Route”).

4. Сценарий использования - "Навигация по выбранному маршруту (Navigation along the selected route)":

- a. Слева отображается карточка следующей точки на маршруте “Next target”. А также списки предстоящих точек (“Upcoming targets”) и пройденных (“Passed targets”).
- b. Справа отображается карта с построенным маршрутом.

5. Сценарий использования - "Добавление пользователем нового маршрута или точки (Add)":

- a. Под верхней навигационной панелью находится переключатель вида добавляемого объекта (Place/Route). По умолчанию открывается “Route”.
- b. Во вкладке “Route”:
 - Справа находится карта, на которой отображаются все интересные точки. Нажатием мышкой на маркер отображается информация (название и координаты) о выбранном месте. По нажатию кнопки

“Add to route” выбранное место отображается в списке мест слева “List of places”. Таким образом пользователь добавляет желаемые места посещения на маршруте.

- В “List of places” каждая карточка места имеет кнопку “Up”, перемещающую ее ближе к началу маршрута, кнопку “Down”, опускающую ее на одно место ниже в списке и кнопку “Delete”, удаляющую ее из списка.
- Под List of places находятся поля для редактирования (“Description”) описания и названия (“Name”) маршрута.
- После нажатия кнопки “Save” внесенные изменения сохраняются.
- При нажатии кнопки “Cancel” изменения удаляются.

c. Во вкладке “Place”:

- Справа на карте выбираются координаты точки.
- Слева находится окно для редактирования параметров места (“Place parameters”) и тэгов, отражающих принадлежность места к определенному виду развлечений (“Preferences”), аналогично “Preferences” на странице “Find”.
- После нажатия кнопки “Save” внесенные изменения сохраняются.
- При нажатии кнопки “Cancel” изменения удаляются.

6. Сценарий использования - " Страница пользователя (User)":

- a. Открывается нажатием на кнопку “User” в навигационной панели сверху.
- b. Ниже навигационной панели находится карточка текущего пользователя. На ней есть кнопка “Edit profile”, открывающая окно для редактирования информации о себе, а также кнопка “Log out”, отвечающая за выход из системы.
- c. Ниже (под заголовком “My Posts”) находится список записей (добавленных интересных мест или созданных маршрутов) текущего пользователя. Список работает аналогично новостной странице (“What's

new”), за исключением того, что в карточке новости появляется кнопка “delete” с возможностью её удалить.

6. Сценарий использования - "Страница администратора":

- a. Если пользователь на странице логина вводит логин и пароль администратора, то открывается панель управления. Вкладки на панели управления работают аналогичным образом (как у пользователя).
- b. По умолчанию открывается вкладка “Global import / export”. Справа отображается окно с картой, на которой отмечены все интересные места, хранящиеся в базе данных. По нажатию на какое-либо место, открывается его карточка, с отображением всей информации о месте, так же доступна возможности её редактировать.
- c. Слева находится окно для импорта / экспорта данных. В соответствующем поле (“Import Data” или “Export data”) вводится название файла, а затем нажимается соответствующая кнопка, которая запускает либо импорт, либо экспорт данных.
- d. В навигационной панели так же присутствуют вкладки “All routes” и “Statistic”.
- e. Во вкладке “all routes” отображается список всех маршрутов, созданных пользователями.
- f. Во вкладке “Statistic” отображается состояние базы данных : список всех типов мест, общее число мест, общее число маршрутов, общее число пользователей.

7. Сценарий использования - "Альтернативный сценарии":

- a. Логин или пароль неправильный:
 - Пользователь вводит неправильные данные в форму авторизации, высвечивается окно с ошибкой.
- b. Пользователь забыл пароль:
 - Пользователь нажимает кнопку “forgot password”;

- Пользователь вводит e-mail и номер телефона в форму;
 - После перехода, или ввода кода из письма или смс (после нажатия кнопки “Send code”) пользователь задаёт новый пароль.
- c. Случайное переключение вкладки внутри приложения:
- Пользователь забыл нажать кнопку “save” в окне создания маршрута;
 - После переключения обратно, можно продолжить редактирование.

2. МОДЕЛЬ ДАННЫХ

2.1. Нереляционная модель данных

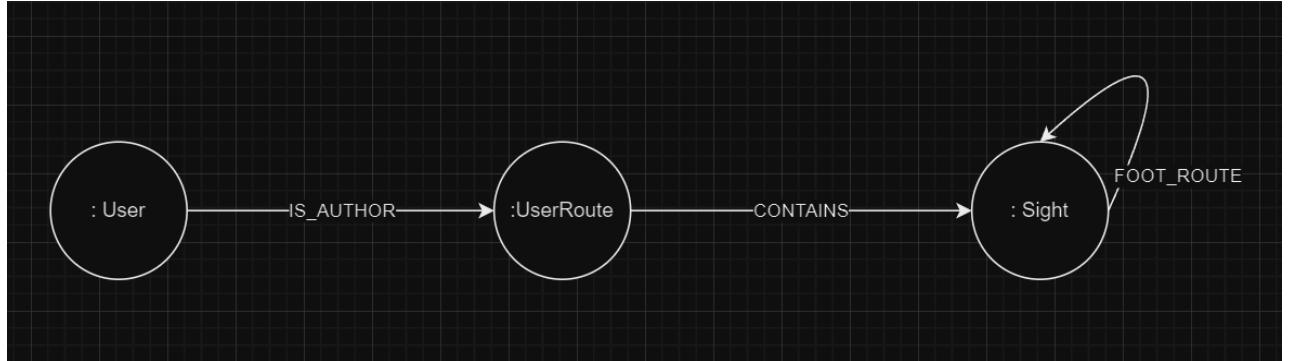


Рисунок 2. Графическое представление нереляционной модели

1. Описание и оценка объема.

Достопримечательность (Sight):

id - идентификатор достопримечательности. Объем 128 байт (string).

lat - широта в градусах от экватора. Объем 8 байт (double).

lon - долгота в градусах от Гринвичского меридиана. Объем 8 байт (double).

name - название достопримечательности. Объем 128 байт (string).

rate - оценка пользователей. Объем 4 байта (int).

Общий объем Sight = 276 байт + 15 байт (объем для хранения каждого узла) = 291.

Пользователь (User):

address - адрес пользователя. Объём 128 байт (string).

email - электронная почта пользователя, предполагается, что она у каждого уникальна. Объём 128 байт (string).

firstName - имя пользователя. Объём 128 байт (string).

lastName - фамилия пользователя. Объём 128 байт (string).

password - пароль пользователя. Объём 128 байт (string).

phone - телефон пользователя. Объём 128 байт (string).

Общий объём User = 783 400 байт

Пользовательский маршрут (UserRoute) :

description - Описание маршрута, которое придумал пользователь. Объём 128 байт (string).

name - название маршрута. Объём 128 байт (string).

sightsSubsequenceIds - список строк (id достопримечательностей, которые входят в маршрут в порядке, заданном пользователем) Объём 128 байт (array).

Общий объём UserRoute = 399 2200 байт

Связь между узлами:

связь CONTAINS - связь между узлом пользовательского маршрута и достопримечательностью - 34 байта

Метки ребер:

FOOT_ROUTE - ребро представляет собой связь между различными объектами карты.

CONTAINS - ребро представляет собой связь между пользовательским маршрутом и достопримечательностями.

IS_AUTHOR - ребро представляет собой связь между пользователем и созданным им маршрутом.

2. Зависимость объема модели от количества узлов (достопримечательностей).

В базе данных заранее можно предсказать только число достопримечательностей (узлы-достопримечательности представляют собой связанный граф).

1. Оценка объема сущностей.

Пусть $M(\text{Sight})$ – объем занимаемый узлом достопримечательности, $M(\text{Relationship})$ – связью между узлами, $M(\text{User})$ – узлом пользователя, $M(\text{UserRoute})$ – маршрутом пользователя. При этом память, занимаемая узлами равна:

$$M(\text{Sight}) = 291 \text{ байт};$$

$$M(\text{Relationship}) = 38 \text{ байта};$$

$$M(\text{User}) = 783 \text{ байт};$$

$$M(\text{UserRoute}) = 399 \text{ байт}.$$

Количество связей между достопримечательностями зависит от количества достопримечательностей в графе (n - количество достопримечательностей в графе) : $N(n) = \frac{n*(n-1)}{2}$.

Пусть $N(\text{UserRoute})$ - общее число пользовательских маршрутов в базе, $N(\text{User})$ - общее число пользователей в базе.

Обозначим суммарный объем памяти, необходимый для хранения пользователей и сохраненных пользовательских маршрутов:
 $c = N(\text{User}) * M(\text{User}) + N(\text{UserRoute}) * M(\text{UserRoute})$.

Тогда память, требуемая моделью равна:

$M(n) = M(Sight) * n + M(Relationship) * N(n) + c$, где n - количество достопримечательностей. Используя известный объем сущностей, получаем

$$M(n) = 2n^2 + 134n + c$$

Если предположить, что в системе зарегистрировано около 40 пользователей и общее число пользовательских маршрутов равняется 60, то получаем зависимость объема модели от количества узлов:

$$M(n) = 2n^2 + 134n + 55260.$$

3. Избыточность модели (отношение между фактическим объемом модели и “чистым” объемом данных).

Избыточность узла достопримечательности – 140 байт;

Избыточность узла User – 383 байта;

Избыточность узла UserRoute – 399 байт;

В целях оптимизации хранения данных neo4j

использует 128 байт для

хранения атрибутов строкового типа, не зависимо от их длины. Следовательно, объём памяти, занимаемой моделью будет значительно больше суммарного объёма хранимых данных. В среднем, при добавлении узлов, которые используются в представленной модели данных, объём модели будет увеличиваться на 300 байт.

То есть реальный объём равен : $Mr = M + 300n$, соответственно, реальный объём БД в 1,3676 раз больше идеального.

4. Направление роста модели при увеличении количества объектов каждой сущности.

Опираясь на вышеизложенные рассуждения, можно сделать вывод, что число узлов модели будет расти линейно, однако число связей между узлами будет расти квадратично.

5. Запросы к модели, с помощью которых реализуются сценарии использования

1. Создание узла достопримечательности

```
CREATE (n:Sight:bank {id: '15834514', lon : 30.3335514, lat : 59.9347916, name : 'Nevskiy 46', rate : 6, kinds:'historic_architecture,architecture,interesting_places,bank,banks,tourist_facilities,other_buildings_and_structures'}) RETURN n.name AS nodeName
```

2. Создание узла пользователя

```
CREATE (:User {email : 'email@gmail.com', password : "password", firstName : 'firstName', lastName : 'lastName', phone : 'phone', address : 'address'})
```

3. Создание узла пользовательского маршрута

```
CREATE (ur:UserRoute{id : 'route_id', name : 'routeName', description : 'routeDescription', sightsSubsequenceIds : ['4986790','5003629','7062698'] }) WITH ur MATCH (sight:Sight) WHERE sight.id IN ur.sightsSubsequenceIds CREATE (ur)-[:CONTAINS]->(sight)
```

4. Получение всех маршрутов конкретного пользователя

```
match (a:User {email : "zlobinandrey0707@gmail.com"}) -[:IS_AUTHOR]->(n) return a, n
```

5. Получение ближайших узлов достопримечательностей к заданным координатам

```
MATCH (node)

WHERE exists(node.lat) AND exists(node.lon)

WITH node, distance(point({latitude: node.lat,
longitude: node.lon}), point({latitude: 59.93429,
longitude: 30.3728965})) AS dist

RETURN node, dist

ORDER BY dist

LIMIT 10
```

6. Построение маршрута от начальной точки до конечной

```
MATCH (joe:Sight {name: "Nekrasov Apartment Museum"})

CALL apoc.path.expandConfig(p, {

    relationshipFilter: "FOOT_ROUTE",
    labelFilter: "+biographical_museums",
    minLevel: 3,
    maxLevel: 4,
    terminateNodes: [joe]})

YIELD path

RETURN path, length(path) AS hops

ORDER BY hops

LIMIT 1)
```

Первые три запроса обеспечивают добавление сущностей в БД. Четвертый запрос позволяет проверить существование пользователя в системе (каждый пользователь имеет уникальный e-mail). Остальные запросы участвуют в генерации маршрутов.

2.2. Аналог модели данных для SQL СУБД

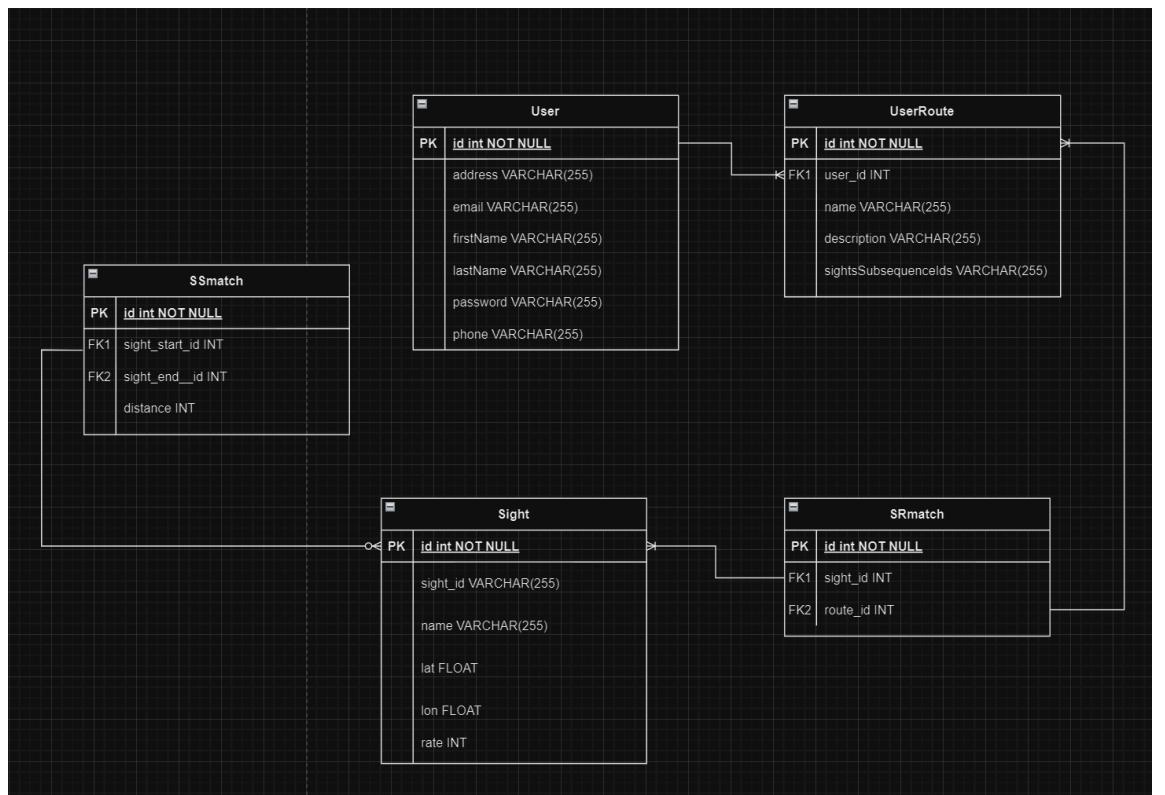


Рисунок 3. Графическое представление реляционной модели

1. Оценка удельного объёма информации, хранимой в модели.

Сущности:

Таблица User:

id - INT: Объём 4 байта;

address - VARCHAR(255): Объём 255 байт;

email - VARCHAR(255): Объём 255 байт;

firstName - VARCHAR(255): Объём 255 байт;

lastName - VARCHAR(255): Объём 255 байт;

password - VARCHAR(255): Объём 255 байт;

phone - VARCHAR(255): Объём 255 байт.

3. Таблица UserRoute:

id - INT: Объём 4 байта;

user_id - INT: Объём 4 байта;

name - VARCHAR(255): Объём 255 байт;

description - VARCHAR(255): Объём 255 байт;

sightsSubsequenceIds - VARCHAR(255) : Объём 255 байт.

4. Таблица Sight:

id - INT: Объём 4 байта

sight_id - VARCHAR(255): Объём 255 байт;

name - VARCHAR(255): Объём 255 байт;

lat - FLOAT: Объём 4 байта;

lon - FLOAT: Объём 4 байта;

rate - INT: Объём 4 байта.

2. Таблица SRmatch:

id - INT: Объём 4 байта;

sight_id - INT: Объём 4 байта;

route_id - INT: Объём 4 байта.

1. Таблица SSmatch:

id - INT : Объём 4 байта;

sight_start_id - INT : Объём 4 байта;

sight_end_id - INT: Объём 4 байта;

distance - INT: Объём 4 байта.

Таблицы SSmatch и SRmatch используются для связи достопримечательностей между собой и связи пользовательского маршрута и достопримечательностей, входящих в него, соответственно.

2. Объем модели от количества узлов (достопримечательностей).

Расчёт объёма модели аналогичен расчету объема нереляционной модели.

$M(\text{Запись таблицы Sight}) = 526 \text{ байт};$

$M(\text{Запись таблицы SSmatch}) = 16 \text{ байт};$

$M(\text{Запись таблицы SRmatch}) = 16 \text{ байт};$

$M(\text{Запись таблицы User}) = 1530 \text{ байт};$

$M(\text{Запись таблицы UserRoute}) = 773 \text{ байт}.$

В случае реляционной модели, связи между достопримечательностями определяются таблицей SSmatch, число связей в которой равно (n - количество достопримечательностей в графе): $N(n) = n * (n - 1)$.

Пусть $N(\text{UserRoute})$ - общее число пользовательских маршрутов в базе, $N(\text{User})$ - общее число пользователей в базе.

Стоит учитывать, что при добавлении нового узла достопримечательности, количество возможных пользовательских маршрутов UserRoute возрастает, при этом количество записей SRmatch возрастает линейно относительно добавления новых записей в UserRoute.

Обозначим суммарный объем памяти, необходимый для хранения пользователей и сохраненных пользовательских маршрутов:

$$c = N(User) * M(User) + N(UserRoute) * M(UserRoute).$$

Тогда память, требуемая моделью равна:

$M(n) = M(Sight) * n + M(Relationship) * N(n) + c$, где n - количество достопримечательностей. Используя известный объем сущностей, получаем

$$M(n) = 12n^2 + 514n + 62033.$$

3. Избыточность модели

В данной модели избыточность будет проявляться в необходимости хранить таблицы, размер которых больше, чем число определяемых ими связей в графе. В среднем, при добавлении записей, которые определены в этой модели, избыточный объем составит приблизительно 660 байт.

Тогда, фактический объем : $Mr = M + 660$, следовательно фактический объем больше чистого в 2,2 раза.

4. Направление роста модели при увеличении количества объектов каждой сущности.

Рост модели будет нелинейным. При добавлении большого количества данных модель возрастает в более чем n^2 раз.

5. Запросы к модели, с помощью которых реализуются сценарии использования.

1. Добавление нового пользователя

```
INSERT INTO "User" (email, password, firstName, lastName, phone, address)
VALUES ('uniq4_email@gmail.com', 'test_password',
'test_first_name3', 'test_last_name4', '+7 (928)-469-49-21',
'address4');
```

2. Поиск пользователя по email

```
SELECT * FROM "User" WHERE email = 'uniq4_email@gmail.com'
```

3. Создание узла достопримечательности

```
INSERT INTO "Sight" (sight_id, name, lat, lon, rate) VALUES ('12376548', 'Nevskiy 46', '59.934123', '30.3333', '6');
```

4. Добавление UserRoute

```
INSERT INTO "UserRoute" (user_id, name, description, sightSubsequenceIds) VALUES ('1', 'lightWalk', 'routeDescription', 'historic');
```

5. Вывод маршрутов пользователя с заданным email

```
SELECT *  
FROM "UserRoute" ur  
JOIN "User" u ON ur.user_id = u.Id  
WHERE u.email = 'uniq1_email@gmail.com';
```

Количество запросов для совершения юзкейсов в реляционной модели может увеличиваться с увеличением числа объектов в БД. Это связано с тем, что в реляционной модели связи между объектами хранятся в отдельных таблицах. Запросы, опирающиеся на построение маршрутов по нескольким критериям не возможны, либо сложно реализуемы.

2.3. Сравнение моделей

1. Удельный объем информации

В реляционной модели необходимо создавать дополнительные таблицы для хранения связей между элементами, при этом число элементов в них будет превышать число связей.

2. Количество запросов для совершения юзкейсов в зависимости от числа объектов в БД

В реляционной базе данных число и сложность запросов, необходимых для реализации функционала построения маршрута, значительно больше, чем в нереляционной графовой БД.

В реляционной модели количество запросов может увеличиваться с увеличением числа объектов в БД.

3. Вывод

Для реализации предложенной задачи больше подходит нереляционная модель. Также стоит учитывать преимущества нереляционной модели, связанные с масштабируемостью и гибкостью модели.

3. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

3.1. Краткое описание

Разработанное приложение выполняет следующие функции:

1. Отображение ленты новостей.
2. Поиск и генерация маршрута по следующим параметрам:
 - a. Начальная точка.
 - b. Конечная точка (необязательно).
 - c. Максимальное количество интересных мест на маршруте.
 - d. Тэги, указывающие на принадлежность места к определенной категории развлечений.
3. Возможность просмотра профиля пользователя.
4. Возможность создания пользовательских маршрутов.
5. Панель администратора предоставляет инструменты для массового импорта / экспорта данных и просмотра содержимого базы данных.

3.2. Использованные технологии

БД: Neo4j (язык запросов Cypher)

Backend: Python (FastApi)

Frontend: Vue, HTML, CSS, JavaScript

3.3. Снимки экрана приложения

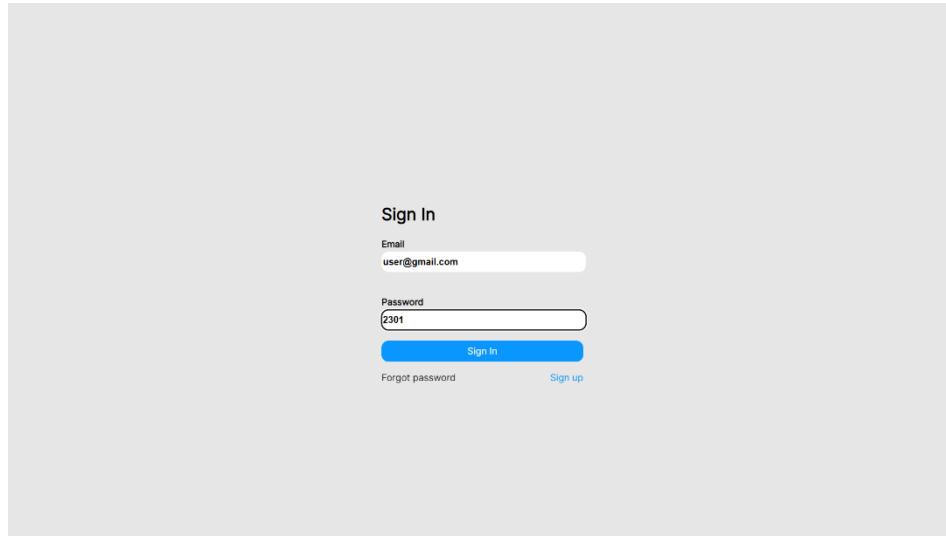


Рисунок 4. Авторизация в приложении.

A screenshot of a mobile application's sign-up screen. The title "Sign Up" is at the top. There are fields for "First name" and "Last name", both currently empty. Below them is an "Enter Your Email" field containing "user@gmail.com". Next is an "Enter Your phone number" field, which is empty. Then is an "Enter Your address" field, also empty. Following that is a "Create a password" field containing "2301". Below it is a "Repeat password" field. A red error message "Password mismatch" is displayed above the "Sign In" button. A "Forgot password" link is to the left of the button, and a "Sign in" link is to its right.

Рисунок 5. Регистрация нового пользователя.

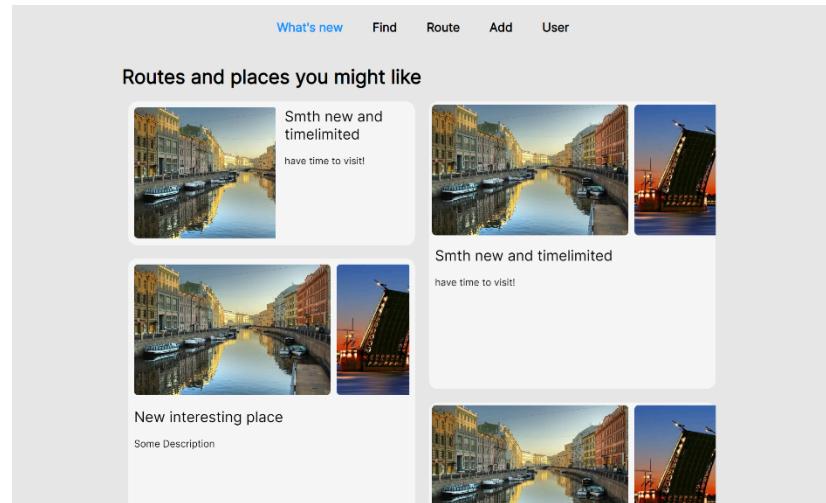


Рисунок 6. Лента новостей.

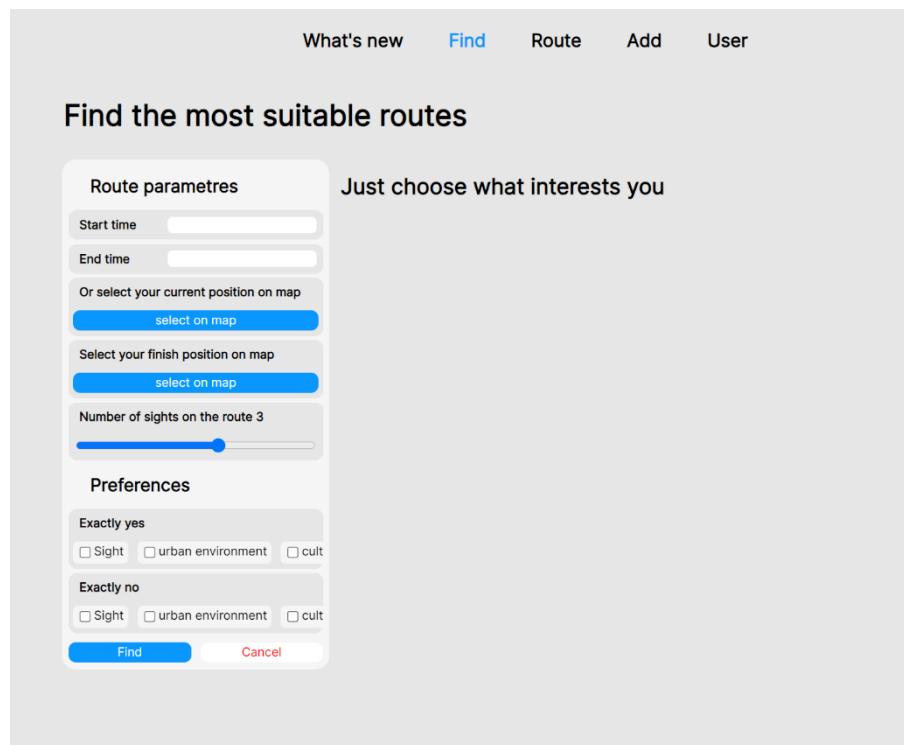


Рисунок 7. Окно поиска и генерации маршрутов.

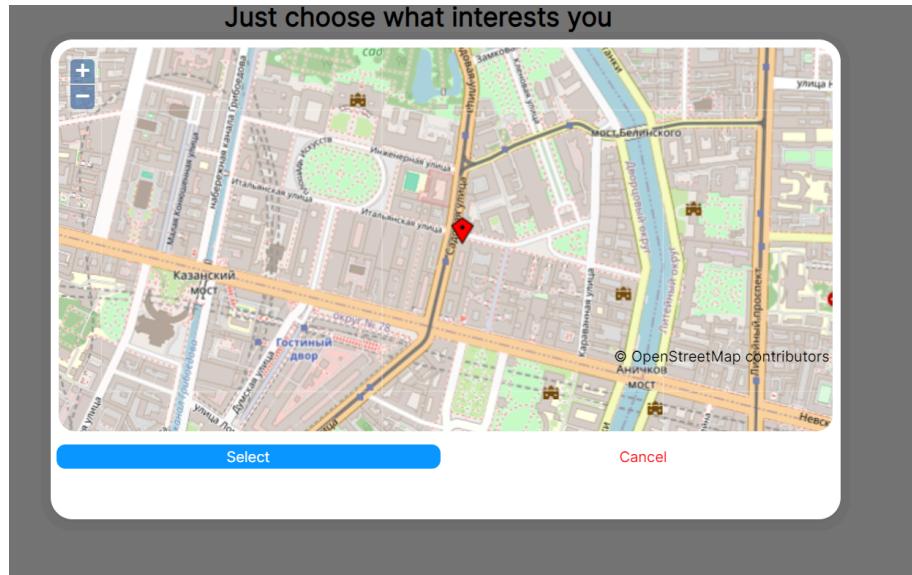


Рисунок 8. Выбор точки старта (точка финиша выбирается аналогично).

Рисунок 9. Найденные User routes относительно заданной точки старта (точка финиша отсутствует), с определенным количеством узлов на маршруте и тэгами.

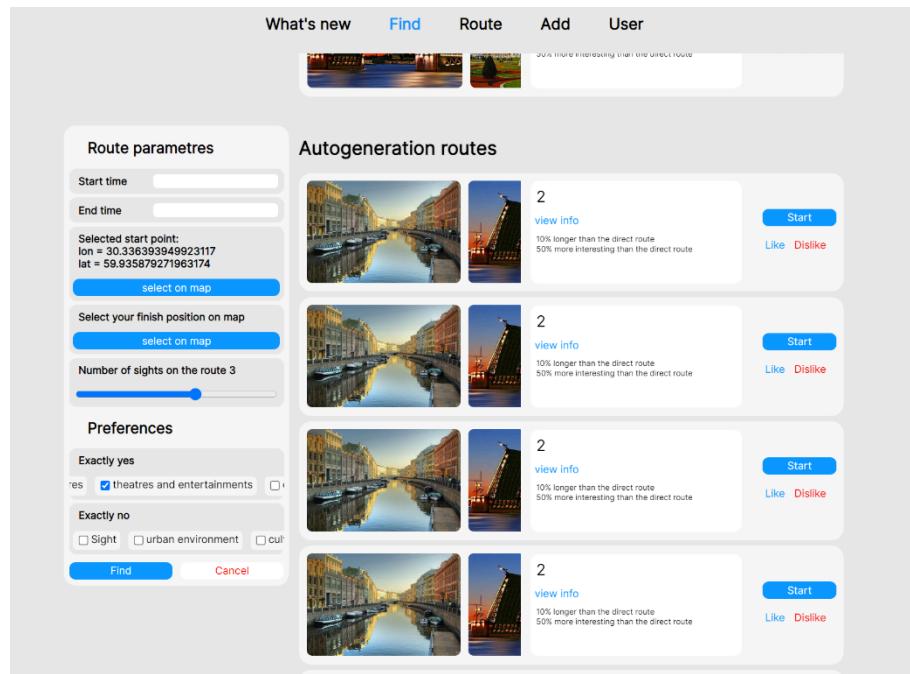


Рисунок 10. Сгенерированные маршруты относительно заданной точки старта (точка финиша отсутствует), с определенным количеством узлов на маршруте и тэгами.

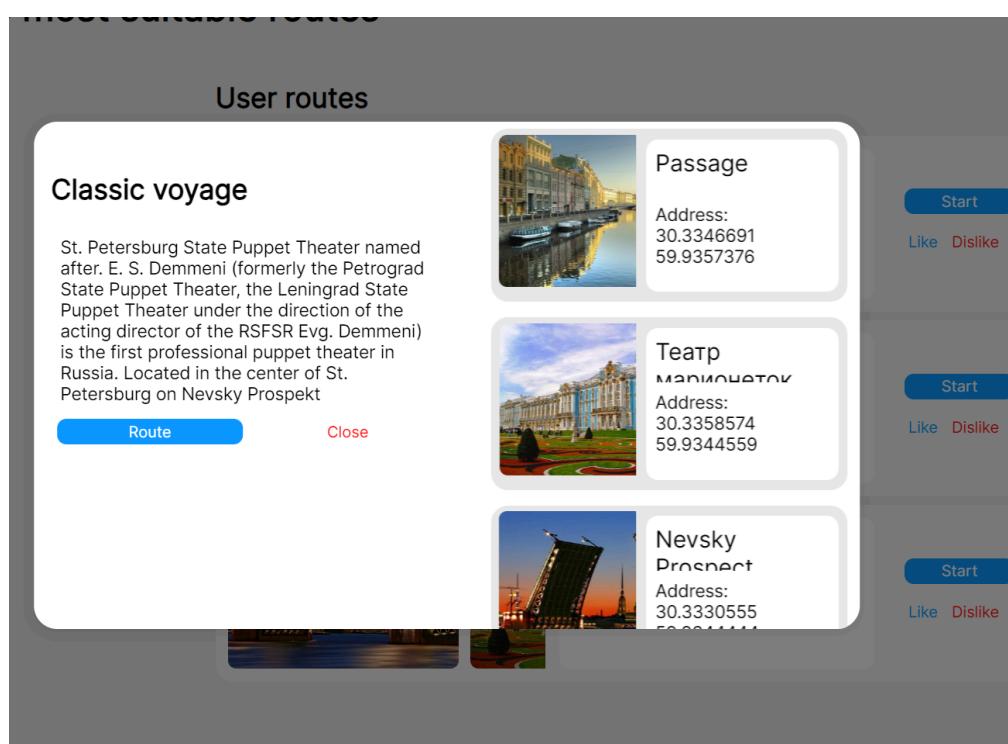


Рисунок 11. Карточка одного маршрута.

The screenshot shows a user interface for route planning. On the left, 'Route parameters' include start and end times, coordinates, and a preference for 'Exact yes' (checkboxes for 'sights', 'religion', 'churches', 'oth'). Below are sections for 'Number of sights on the route 3' and 'Autogeneration routes'. The 'Autogeneration routes' section displays two routes labeled 1 and 2, each with a thumbnail image of a canal scene, a route summary ('10% longer than the direct route', '50% more interesting than the direct route'), and 'Start', 'Like', and 'Dislike' buttons.

Classic voyage
view info Start Like Dislike

Histirical monuments
view info Start Like Dislike

Autogeneration routes

1
view info Start Like Dislike

2
view info Start Like Dislike

House with Fair
view info

Armenian apostolic
view info

Домовая церковь
view info

Рисунок 12. Найденные User routes и сгенерированные маршруты Autogeneration routes относительно заданной точки старта и финиша, с определенными количеством узлов на маршруте и тэгами (как положительными, так и отрицательными).

The screenshot shows a 'Add new route' form. It includes a 'List of places' section with three items: 'Theatre of Musical Comedy' (Address: 30.3329411, 59.9360161), 'Museum of Lünnino' (Address: 30.3373775, 59.9358292), and 'Летняя выставка' (Address: 30.3380413, 59.9352074). A 'Description' field contains 'Summer walk' and 'Very very interesting'. To the right, a 'Place' section shows a map with numerous red location markers and a 'Grand Palace' thumbnail. Another 'Add new route' form on the right shows a similar list of places and a map.

Рисунок 13. Добавление нового пользовательского маршрута.

The screenshot shows a user profile for 'Andrew Zlobin' from Saint Petersburg. The profile picture is a canal scene. The 'My posts' section displays four images with captions: 'Smth new and timelimited have time to visit!', 'Smth new and timelimited have time to visit!', and two other images. At the top, there's a navigation bar with 'What's new', 'Find', 'Route', 'Add', and 'User'.

Рисунок 14. Профиль пользователя.

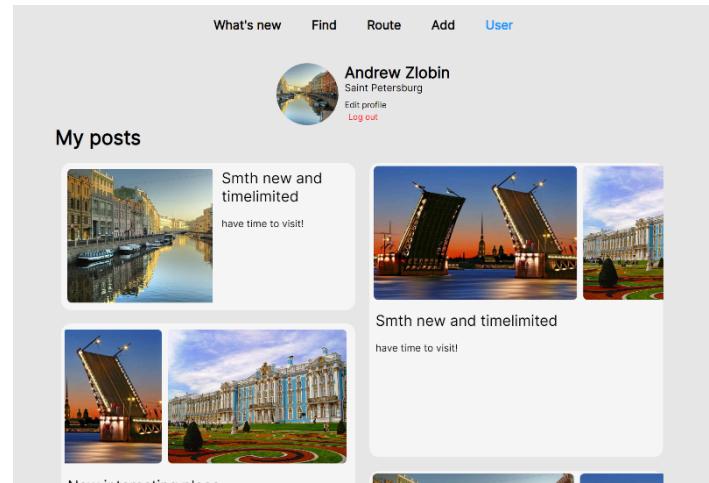


Рисунок 15. Прототип навигации по выбранному маршруту.

ЗАКЛЮЧЕНИЕ

Фильтрация маршрутов и генерация наиболее подходящих под заданные параметры была успешно интегрирована в web-приложение. Так же было реализовано добавление пользовательских маршрутов, которые могут предлагаться вместе со сгенерированными, если они соответствуют заданным фильтрам. В дальнейшем развитии планируется добавление большего числа узлов достопримечательностей в БД, увеличение количества параметров генерации и улучшение алгоритмов генерации маршрутов и их поиска.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Репозиторий веб-приложения: [электронный ресурс]. URL:
<https://github.com/moevm/nosql2h23-spb-routes> (дата обращения 01.02.2024).
2. Документация по Neo4j: [электронный ресурс]. URL:
<https://neo4j.com/docs/getting-started/> (дата обращения 01.02.2024).
3. Документация по Cypher: [электронный ресурс]. URL:
<https://neo4j.com/docs/cypher-manual/> (дата обращения 01.02.2024).
4. Документация по FastApi: [электронный ресурс]. URL:
<https://fastapi.tiangolo.com/> (дата обращения 01.02.2024).

ПРИЛОЖЕНИЕ А

ДОКУМЕНТАЦИЯ ПО СБОРКЕ И РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ

Сборка сервера с СУБД Neo4j:

docker-compose up --build

При первом запуске (с пустой базой данных) создаётся два пользователя - "user@gmail.com" и "admin@gmail.com". Для заполнения базы данных узлами использовался opentripmap api. В файле importJsonWithSights содержится обёртка для этого апи, которая сохраняет участок bbox в json. Для нахождения кратчайших маршрутов между этими точками использовался сервис OSRM (в дальнейшем его планировалось использовать для вкладки навигации). Для его запуска необходимо выполнить следующие команды:

```
wget https://download.geofabrik.de/russia/northwestern-fed-district-latest.osm.pbf
```

```
docker run -t -v "${PWD}:/data" ghcr.io/project-osrm/osrm-backend osrm-extract -p /opt/foot.lua /data/northwestern-fed-district-latest.osm.pbf echo "osrm-extract failed"
```

```
docker run -t -v "${PWD}:/data" ghcr.io/project-osrm/osrm-backend osrm-partition /data/northwestern-fed-district-latest.osrm echo "osrm-partition failed"
```

```
docker run -t -v "${PWD}:/data" ghcr.io/project-osrm/osrm-backend osrm-customize /data/northwestern-fed-district-latest.osrm || echo "osrm-customize failed"
```

```
docker run -t -i -p 5000:5000 -v "${PWD}:/data" ghcr.io/project-osrm/osrm-backend
osrm-routed --algorithm mld /data/northwestern-fed-district-latest.osrm
```

В файле DBdriver.py реализованы методы для взаимодействия с базой данных. Если запустить его напрямую, то он попытается подключиться к сервису OSRM на порту 5000 и построить узлы для достопримечательностей из списка data.json, а также рассчитать длину соединений между ними по времени.