

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: Сервис поиск переименованных в советское время топонимов в СПб

Студент гр. 1304

Кривоченко Д.И.

Студент гр. 1304

Мусаев А.И.

Студент гр. 1304

Поршнеv Р.А.

Преподаватель

Заславский М. М.

Санкт-Петербург

2024

ЗАДАНИЕ

Студент Кривоченко Д.И.

Студент Мусаев А.И.

Студент Поршнева Р.А.

Группа 1304

Тема: Сервис поиска переименованных в советское время топонимов в СПб

Исходные данные:

Необходимо создать веб-приложение с использованием СУБД neo4j, с помощью которого можно просматривать информацию об переименованных топонимах Санкт-Петербурга и фильтровать их по заданным критериям.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Сценарии использования»

«Модель данных»

«Разработанное приложение»

«Выводы»

«Приложения»

«Список использованных источников»

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 05.09.2024

Дата сдачи реферата: 22.12.2024

Дата защиты реферата: 24.12.2024

Студент	_____	Кривоченко Д. И.
Студент	_____	Мусаев А. И.
Студент	_____	Поршнев Р. А.
Преподаватель	_____	Заславский М. М.

АННОТАЦИЯ

В рамках ИДЗ было разработано веб-приложение для просмотра информации об переименованных топонимах Санкт-Петербурга. Также разработана фильтрация топонимов по таким критериям, как тип топонима, стиль, наличие фотографии, ФИО архитектора, информация в карточке топонима, адрес, название, год начала строительства, год конца строительства, год первого переименования и год последнего переименования. Также реализован импорт и экспорт данных о топонимах.

В процессе разработки были использованы технологии Angular, nginx, Flask, СУБД neo4j и Docker Compose.

Исходный код доступен по ссылке:
<https://github.com/moevm/nosql2h24-rename>

SUMMARY

As part of an individual homework assignment, a web application was developed to view information about the renamed toponyms of St. Petersburg. Toponyms have also been filtered according to criteria such as the type of toponym, style, photo, architect's full name, information in the toponym card, address, name, year of construction start, year of end of construction, year of first renaming and year of last renaming. The import and export of toponymic data is also implemented.

During the development process, Angular, nginx, Flask, neo4j, and Docker Compose technologies were used.

The source code is available at the link:
<https://github.com/moevm/nosql2h24-rename>

СОДЕРЖАНИЕ

1.	Введение	7
1.1.	Актуальность проблемы	7
1.2.	Постановка задачи	7
1.3.	Предлагаемое решение	7
1.4.	Качественные требования к решению	8
2.	Сценарии использования	9
2.1.	Макет UI	9
2.2.	Сценарий использования для импорта данных	15
2.3.	Сценарий использования для отображения данных на главной странице	16
2.4.	Сценарий использования для отображения отфильтрованных данных по типу топонима	16
2.5.	Сценарий использования для отображения отфильтрованных данных по стилю топонима	16
2.6.	Сценарий использования для отображения отфильтрованных данных по наличию фотографий	17
2.7.	Сценарий использования для отображения отфильтрованных данных по ФИО архитектору топонимов	17
2.8.	Сценарий использования для отображения отфильтрованных данных по информации в карточках топонимов	18

2.9.	Сценарий использования для отображения топонима по заданному адресу	18
2.10.	Сценарий использования для отображения топонимов с заданным названием	19
2.11.	Сценарий использования для отображения топонимов с годом постройки не менее заданного	19
2.12.	Сценарий использования для отображения топонимов с годом постройки не более заданного	19
2.13.	Сценарий использования для отображения топонимов с годом переименования не менее заданного	20
2.14	Сценарий использования для отображения топонимов с годом переименования не более заданного	20
2.15	Сценарий использования для экспорта данных	21
2.16	Вывод	
3.	Модель данных	22
3.1.	Нереляционная модель данных	22
3.2.	Реляционная модель данных	29
3.3.	Сравнение моделей	36
4.	Разработанное приложение	38
4.1.	Краткое описание приложения	38
4.2.	Использованные технологии	38

4.3.	Схема экранов приложения	38
5.	Выводы	41
5.1.	Достигнутые результаты	41
5.2.	Недостатки и пути для улучшения полученного решения	41
5.3.	Будущее развитие решения	42
6.	Приложения	43
6.1.	Документация по сборке и развертыванию приложения	43
6.2.	Инструкция для пользователя	43
	Список использованных источников	44

1. ВВЕДЕНИЕ

1.1. Актуальность проблемы

В Санкт-Петербурге произошло множество изменений в названиях мостов, площадей, зданий, парков и других географических объектов, особенно в результате политических изменений. Информация о переименованиях является культурным наследием, которое не стоит забывать, в особенности жителям данного города. Создание ресурса для просмотра переименованных топонимов поможет сохранить культурное наследие Санкт-Петербурга, а также – может помогать в образовательных процессах.

1.2. Постановка задачи.

Задача проекта заключается в разработке веб-приложения, которое позволяет пользователям:

- Просматривать топонимы;

- Просматривать годы переименования топонимов, их стили, типы, фотографии, архитекторов, адреса и названия;
- Фильтровать топонимы по типу, стилю, наличию фотографий, архитекторам, адресу, названию, информации в карточках топонимов, году начала постройки, году конца постройки, году начала переименования, году конца переименования;
- Экспортировать и импортировать данные о топонимах;
- Запускать приложение локально с помощью Docker Compose.

1.3. Предлагаемое решение.

Для реализации проекта должно быть создано веб-приложение с использованием Angular для клиентской части, Flask для серверной части, neo4j для хранения данных, Docker Compose для контейнеризации и развертывания вышеописанных частей приложения, а также nginx для обслуживания статических страниц после сборки клиентской части в Docker.

1.4. Качественные требования к решению.

Разрабатываемое веб-приложение должно предоставлять возможность просматривать карточки топонимов, устанавливать фильтры по типу, стилю, наличию фотографий, архитекторам, адресу, названию, информации в карточках топонимов, году начала постройки, году конца постройки, году начала переименования, году конца переименования. Также должна быть предоставлена возможность импорта и экспорта данных и локального развертывания с помощью Docker Compose.

2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

2.1. Макет UI

Ниже представлен макет приложения (рис. 1-13).

ТаблицаКартаО проекте

Тип

Стиль

Наличие фото

Архитектор

Поиск по карточке

Адрес

Название

Дата постройки от

Дата постройки до

Переименован в году от

Переименован в году до

Экспорт

Импорт

Название	Годы	Адрес	Фото	Тип	Стиль	Архитектор
Санкт-Петербургский горный университет императрицы Екатерины II	2023,2016,1956,1924,1896,1866,1804,1773	Лейтенанта Шмидта наб., 45;21-я линия ВО, 2-6;22-я линия ВО, 1		Здание	Классицизм	Воронилин Андрей Никифорович
Парк имени Есенина	2012,1980	Большевиков пр., 16;Дыбенко ул., 28;Товарищеский пр., 13		Парк		
Морской корпус Петра Великого — Санкт-Петербургский военно-морской институт	2001,1926,1867,1762,1715,1701	Лейтенанта Шмидта наб., 17;12-я линия ВО, 1;11-я линия ВО, 2		Здание	Классицизм	Волков Фёдор Иванович
Многопрофильная клиника имени Н.И. Пирогова	1999,1918,1899	Большой пр. ВО, 49-51		Мост		Шмидт Карл Карлович
Санкт-Петербургский государственный технологический институт (Технический университет)	1993,1923,1896,1862,1831	Загородный пр., 49		Здание	Классицизм	Постников Александр Иванович

<

1

2

3

>

Рисунок 1 – Главная страница

ТаблицаКартаО проекте

Тип

Мост, Здание

Стиль

Наличие фото

Архитектор

Поиск по карточке

Адрес

Название

Дата постройки от

Дата постройки до

Переименован в году от

Переименован в году до

Экспорт

Импорт

Название	Годы	Адрес	Фото	Тип	Стиль	Архитектор
Санкт-Петербургский горный университет императрицы Екатерины II	2023,2016,1956,1924,1896,1866,1804,1773	Лейтенанта Шмидта наб., 45;21-я линия ВО, 2-6;22-я линия ВО, 1		Здание	Классицизм	Воронилин Андрей Никифорович
Морской корпус Петра Великого — Санкт-Петербургский военно-морской институт	2001,1926,1867,1762,1715,1701	Лейтенанта Шмидта наб., 17;12-я линия ВО, 1;11-я линия ВО, 2		Здание	Классицизм	Волков Фёдор Иванович
Многопрофильная клиника имени Н.И. Пирогова	1999,1918,1899	Большой пр. ВО, 49-51		Мост		Шмидт Карл Карлович
Санкт-Петербургский государственный технологический институт (Технический университет)	1993,1923,1896,1862,1831	Загородный пр., 49		Здание	Классицизм	Постников Александр Иванович
Российская академия наук	1991,1925,1917,1836,1789	Университетская наб., 5		Здание	Неоклассицизм	Джакомо Кваренги

<

1

2

>

Рисунок 2 – Фильтр по типу топонимов

Таблица

Карта

О проекте

Тип

Стиль

Наличие фото

Архитектор

Поиск по карточке

Адрес

Название

Дата постройки от

Дата постройки до

Переименован в году от

Переименован в году до

Экспорт

Импорт

Название	Годы	Адрес	Фото	Тип	Стиль	Архитектор
Санкт-Петербургский горный университет императрицы Екатерины II	2023, 2016, 1956, 1924, 1896, 1866, 1804, 1773	Лейтенанта Шмидта наб., 45:21-я линия ВО, 2-6:22-я линия ВО, 1		Здание	Классицизм	Воронихин Андрей Никифорович
Морской корпус Петра Великого — Санкт-Петербургский военно-морской институт	2001, 1926, 1867, 1762, 1715, 1701	Лейтенанта Шмидта наб., 17:12-я линия ВО, 1:11-я линия ВО, 2		Здание	Классицизм	Волков Федор Иванович
Санкт-Петербургский государственный технологический институт (Технический университет)	1993, 1923, 1896, 1862, 1831	Загородный пр., 49		Здание	Классицизм	Постников Александр Иванович
Ленинградским опытным заводом электронных приборов времени «Хронопрон»	1975, 1938, 1924, 1918	Достоевского ул., 40-44		Здание	Кирпичный	Воронихин Андрей Никифорович
Патриотический институт	1827, 1813	10-я линия ВО, 3		Здание	Классицизм	Берзен Рихард Андреевич, Китнер Иероним Севастьянович

< 1 >

Рисунок 3 – Фильтр по стилю топонимов

Таблица

Карта

О проекте

Тип

Стиль

Наличие фото

Архитектор

Поиск по карточке

Адрес

Название

Дата постройки от

Дата постройки до

Переименован в году от

Переименован в году до

Экспорт

Импорт

Название	Годы	Адрес	Фото	Тип	Стиль	Архитектор
Многопрофильная клиника имени Н. И. Пирогова	1999, 1918, 1899	Большой пр. ВО, 49-51		Мост		Шмидт Карл Карлович
мост Степана Разина	1939, 1923, 1914	Обводный кан., 1		Мост	Современный	Гутцайт А. Д.

< 1 >

Рисунок 4 – Фильтр по наличию фотографий топонимов

Таблица

Карта

О проекте

Тип

↑

Стиль

↑

Наличие фото

↓

Архитектор

Вороники

×

Поиск по карточке

Адрес

Название

Дата постройки от



Дата постройки до

Переименован в году от

Переименован в году до

Экспорт

Импорт

Название	Годы	Адрес	Фото	Тип	Стиль	Архитектор
Санкт-Петербургский горный университет императрицы Екатерины II	2023,2016,1956,1924,1896,1866,1804,1773	Лейтенанта Шмидта наб., 45;21-я линия ВО, 2-6;22-я линия ВО, 1		Здание	Классицизм	Вороникин Андрей Никифорович
Ленинградским опытным заводом электронных приборов времени «ХроноТрон»	1975,1938,1924,1918	Достоевского ул., 40-44		Здание	Кирпичный	Вороникин Андрей Никифорович

<

1

>

Рисунок 5 – Фильтр по именам архитекторов топонимов

Таблица

Карта

О проекте

Тип

Стиль

Наличие фото

Архитектор

Поиск по карточке институту

Адрес

Название

Дата постройки от


Дата постройки до

Переименован в году от

Переименован в году до

Экспорт

Импорт

Название	Годы	Адрес	Фото	Тип	Стиль	Архитектор
Морской корпус Петра Великого — Санкт-Петербургский военно-морской институт	2001,1926,1867,1762,1715,1701	Лейтенанта Шмидта наб., 17;12-я линия ВО, 1;11-я линия ВО, 2		Здание	Классицизм	Волков Федор Иванович

<

1

>

Рисунок 6 – Фильтр по содержанию карточки топонимов

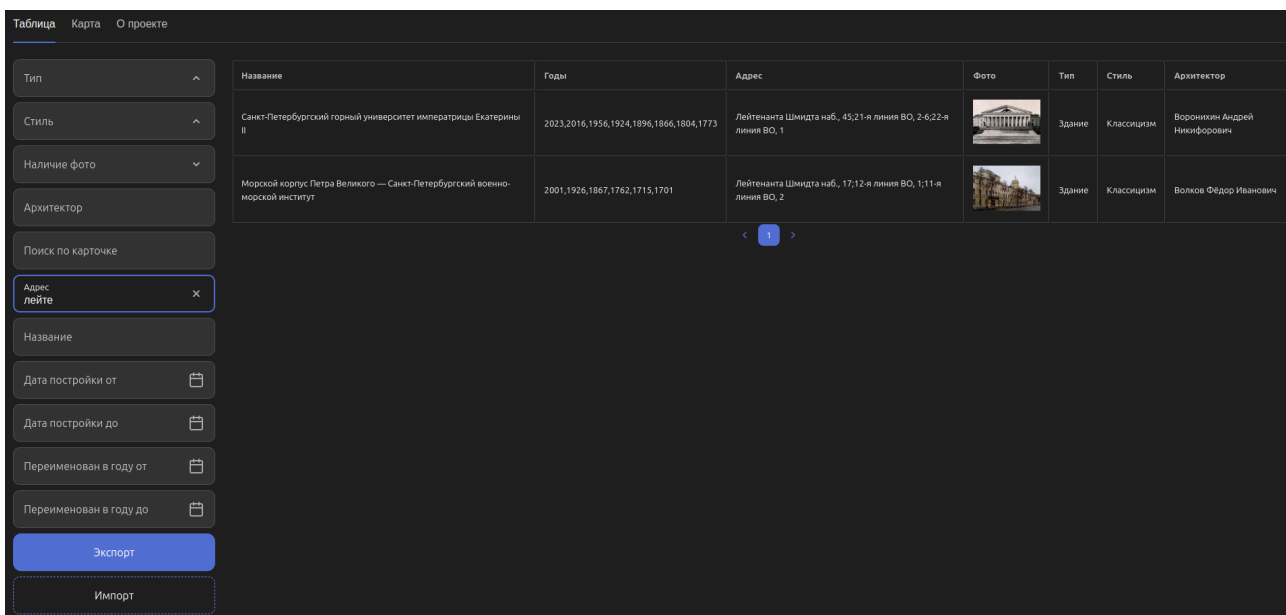


Рисунок 7 – Фильтр по адресу топонимов

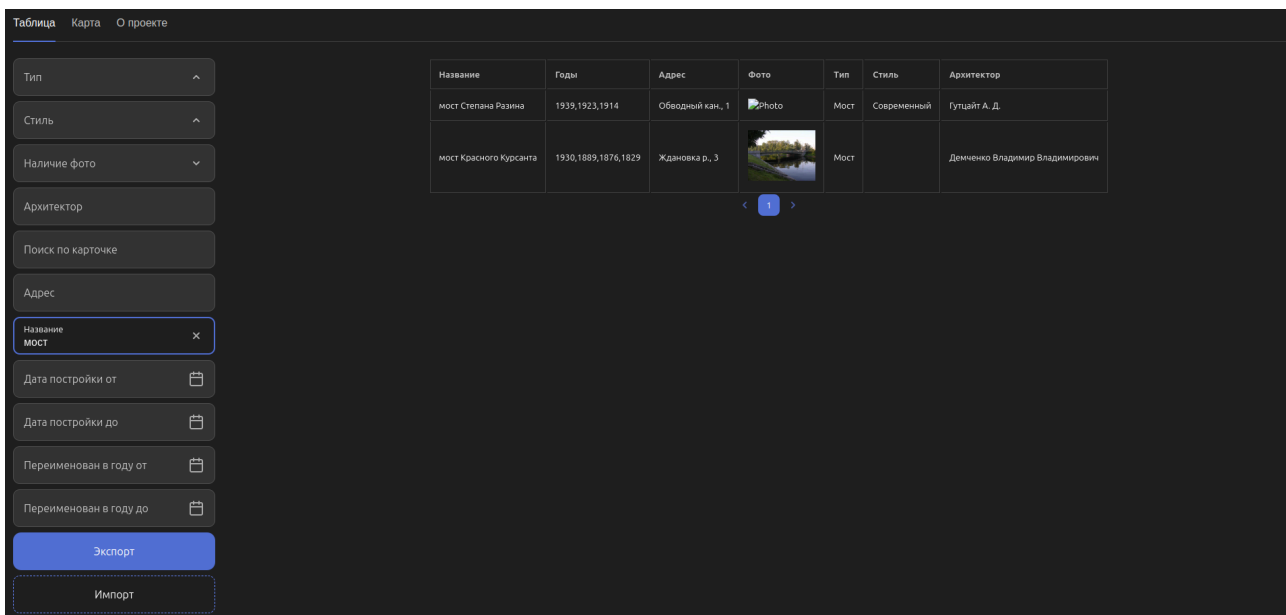


Рисунок 8 – Фильтр по названию топонимов

ТаблицаКартаО проекте

Тип

↑

Стиль

↑

Наличие фото

↓

Архитектор

Поиск по карточке

Адрес

Название

Дата постройки от

Дата постройки до

Переименован в году от

Переименован в году до

Экспорт

Импорт

Название	Годы	Адрес	Фото	Тип	Стиль	Архитектор
Парк имени Есенина	2012,1980	Большевиков пр., 16;Дыбенко ул., 28;Товарищеский пр., 13		Парк		
Многопрофильная клиника имени Н.И. Пирогова	1999,1918,1899	Большой пр. ВО, 49-51		Мост		Шмидт Карл Карлович
Санкт-Петербургский государственный технологический институт (Технический университет)	1993,1923,1896,1862,1831	Загородный пр., 49		Здание	Классицизм	Постников Александр Иванович
Ленинградским опытным заводом электронных приборов времени «ХроноТрон»	1975,1938,1924,1918	Достоевского ул., 40-44		Здание	Кирпичный	Воронихин Андрей Никифорович
Центральная городская библиотека им. В. В. Маяковского	1954,1928,1919,1868	Фонтанки наб., 44		Здание	Неорусский	Горностаев Алексей Максимович

<

1

2

>

Рисунок 11 – Фильтр по году начала переименования

Таблица

Карта

О проекте

Тип

↑

Стиль

↑

Наличие фото

↓

Архитектор

Поиск по карточке

Адрес

Название

Дата постройки от

📅

Дата постройки до

📅

Переименован в году от

📅

Переименован в году до





1954

✕

📅

Экспорт

Импорт

Название	Годы	Адрес	Фото	Тип	Стиль	Архитектор
Центральная городская библиотека им. В. В. Маяковского	1954,1928,1919,1868	Фонтанки наб., 44		Здание	Неорусский	Горностаев Алексей Максимович
мост Степана Разина	1939,1923,1914	Обводный кан., 1		Мост	Современный	Гутцайт А. Д.
мост Красного Курсанта	1930,1889,1876,1829	Ждановка р., 3		Мост		Демченко Владимир Владимирович
Патриотический институт	1827,1813	10-я линия ВО, 3		Здание	Классицизм	Берзен Рихард Андреевич,Китнер Иероним Севастьянович

<

1

>

Рисунок 12 – Фильтр по году конца переименования

Таблица Карта О проекте

Тип

Стиль

Наличие фото

Архитектор

Поиск по карточке

Адрес

Название

Дата постройки от

Дата постройки до

Переименован в году от

Переименован в году до

Экспорт

Импорт

Название	Годы	Адрес	Фото	Тип	Стиль	Архитектор
Российская академия наук	1991,1925,1917,1836,1789	Университетская наб., 5		Здание	Неоклассицизм	Джакомо Кваренги
Ленинградским опытным заводом электронных приборов времени «ХроноТрон»	1975,1938,1924,1918	Достоевского ул., 40-44		Здание	Кирпичный	Воронихин Андрей Никифорович
Центральная городская библиотека им. В. В. Маяковского	1954,1928,1919,1868	Фонтанки наб., 44		Здание	Неорусский	Горюхаев Алексей Максимович
мост Степана Разина	1939,1923,1914	Обводный кан., 1		Мост	Современный	Гутцайт А. Д.
мост Красного Курсанта	1930,1889,1876,1829	Ждановка р., 3		Мост		Демченко Владимир Владимирович

Рисунок 13 – Пагинация

2.2. Сценарий использования для импорта данных

Действующее лицо: пользователь.

Описание: пользователь догружает данные в формате json в базу данных для их дальнейшего использования в рамках приложения.

Предусловие: пользователь должен иметь активное интернет-соединение.

Основной сценарий:

1. Пользователь открывает веб-приложение.
2. Пользователь переносит данные в окошко либо нажимает на кнопку “Импорт” и выбирает данные на своём компьютере.
3. Уведомление об успешном завершении операции.

Альтернативный сценарий:

1. Пользователь открывает веб-приложение.
2. Пользователь переносит данные в окошко либо нажимает на кнопку “Импорт” и выбирает данные на своём компьютере.
3. Уведомление о неуспешном завершении операции ввиду несоответствия структуры json-файла.

Результат:

1. Уведомление об успешном импорте данных.
2. Уведомление о неуспешном импорте данных.

2.3. Сценарий использования для отображения данных на главной странице

Действующее лицо: пользователь.

Описание: пользователь открывает веб-приложение для просмотра таблицы топонимов.

Предусловие: пользователь должен иметь стабильное интернет-соединение.

Основной сценарий:

1. Пользователь открывает веб-приложение.

Результат: пользователь видит таблицу со всеми доступными для просмотра топонимами.

2.4. Сценарий использования для отображения отфильтрованных данных по типу топонима

Действующее лицо: пользователь.

Описание: пользователь использует фильтр по типу для получения необходимой информации.

Предусловие: пользователь должен иметь стабильное интернет-соединение.

Основной сценарий:

1. Пользователь открыл веб-приложение.
2. Пользователь нажал на фильтр по типу топонимов и выбрал необходимые ему значения из следующего списка: “Мост”, “Здание”, “Парк”, “Другие”.

Результат: обновление таблицы в соответствии с установленными значениями фильтра по типу топонимов.

2.5. Сценарий использования для отображения отфильтрованных данных по стилю топонима

Действующее лицо: пользователь.

Описание: пользователь использует фильтр по стилю для получения необходимой информации.

Предусловие: пользователь должен иметь стабильное интернет-соединение.

Основной сценарий:

1. Пользователь открыл веб-приложение.
2. Пользователь нажал на фильтр по стилю топонимов и выбрал необходимые ему значения из следующего списка: “Классицизм”, “Неоклассицизм”, “Кирпичный”, “Современный”.

Результат: обновление таблицы в соответствии с установленными значениями фильтра по стилю топонимов.

2.6. Сценарий использования для отображения отфильтрованных данных по наличию фотографий

Действующее лицо: пользователь.

Описание: пользователь использует фильтр по наличию фотографий для получения необходимой информации.

Предусловие: пользователь должен иметь стабильное интернет-соединение.

Основной сценарий:

1. Пользователь открыл веб-приложение.
2. Пользователь нажал на фильтр по наличию фотографий и выбрал одно из значений: “Есть”, “Нет”, “Не выбрано”.

Результат: обновление таблицы в соответствии с установленными значениями фильтра по наличию фотографий топонимов.

2.7. Сценарий использования для отображения отфильтрованных данных по ФИО архитектору топонимов

Действующее лицо: пользователь.

Описание: пользователь использует фильтр по архитектору для получения необходимой информации.

Предусловие: пользователь должен иметь стабильное интернет-соединение.

Основной сценарий:

1. Пользователь открыл веб-приложение.
2. Пользователь ввёл данные из ФИО архитектора в поле с названием “Архитектор”.

Результат: обновление таблицы в соответствии с данными из ФИО архитектора топонимов.

2.8. Сценарий использования для отображения отфильтрованных данных по информации в карточках топонимов

Действующее лицо: пользователь.

Описание: пользователь использует фильтр по информации в карточках топонимов для получения обновлённого списка топонимов.

Предусловие: пользователь должен иметь стабильное интернет-соединение.

Основной сценарий:

1. Пользователь открыл веб-приложение.
2. Пользователь ввёл данные в текстовом формате в поле с названием “Поиск по карточке”.

Результат: обновление таблицы в соответствии с информацией в поле “Поиск по карточке”.

2.9. Сценарий использования для отображения топонима по заданному адресу

Действующее лицо: пользователь.

Описание: пользователь использует фильтр по адресу топонима для получения необходимых данных.

Предусловие: пользователь должен иметь стабильное интернет-соединение.

Основной сценарий:

1. Пользователь открыл веб-приложение.

2. Пользователь ввёл адрес топонима в поле с названием “Адрес”.

Результат: обновление таблицы в соответствии с введённым адресом топонима.

2.10. Сценарий использования для отображения топонимов с заданным названием

Действующее лицо: пользователь.

Описание: пользователь использует фильтр по названию топонима для получения необходимых данных.

Предусловие: пользователь должен иметь стабильное интернет-соединение.

Основной сценарий:

1. Пользователь открыл веб-приложение.
2. Пользователь ввёл часть названия топонима в поле “Название” для получения необходимых данных.

Результат: обновление таблицы в соответствии с введённым названием топонима.

2.11. Сценарий использования для отображения топонимов с годом постройки не менее заданного

Действующее лицо: пользователь.

Описание: пользователь использует фильтр по минимального году постройки для получения необходимых данных.

Предусловие: пользователь должен иметь стабильное интернет-соединение.

Основной сценарий:

1. Пользователь открыл веб-приложение.
2. Пользователь ввёл минимальный год постройки.

Результат: обновление таблицы в соответствии с введённым минимальным годом постройки.

2.12. Сценарий использования для отображения топонимов с годом постройки не более заданного

Действующее лицо: пользователь.

Описание: пользователь использует фильтр по максимальному году постройки для получения необходимых данных.

Предусловие: пользователь должен иметь стабильное интернет-соединение.

Основной сценарий:

1. Пользователь открыл веб-приложение.
2. Пользователь ввёл максимальный год постройки.

Результат: обновление таблицы в соответствии с введённым максимальным годом постройки.

2.13. Сценарий использования для отображения топонимов с годом переименования не менее заданного

Действующее лицо: пользователь.

Описание: пользователь использует фильтр по минимального году переименования для получения необходимых данных.

Предусловие: пользователь должен иметь стабильное интернет-соединение.

Основной сценарий:

1. Пользователь открыл веб-приложение.
2. Пользователь ввёл минимальный год переименования.

Результат: обновление таблицы в соответствии с введённым минимальным годом переименования.

2.14. Сценарий использования для отображения топонимов с годом переименования не более заданного

Действующее лицо: пользователь.

Описание: пользователь использует фильтр по максимальному году переименования для получения необходимых данных.

Предусловие: пользователь должен иметь стабильное интернет-соединение.

Основной сценарий:

1. Пользователь открыл веб-приложение.
2. Пользователь ввёл минимальный год переименования.

Результат: обновление таблицы в соответствии с введённым минимальным годом переименования.

2.15. Сценарий использования для экспорта данных

Действующее лицо: пользователь.

Описание: пользователь выгружает данные в формате json из базы данных.

Предусловие: пользователь должен иметь активное интернет-соединение.

Основной сценарий:

1. Пользователь открывает веб-приложение.
2. Пользователь нажимает на кнопку “Экспорт”
3. Уведомление об успешном завершении операции.

Результат: уведомление об успешном экспорте данных.

2.16. Вывод

На основании описанных сценариев использования, а также из данной предметной области, можно сделать вывод, что преобладающей операцией будет чтение, так как новые топонимы будут добавляться довольно редко, ведь большинство из них хорошо изучено. Следовательно, данные будут редко добавляться в систему из-за их небольшого количества. Добавлению данных в базу соответствует операция импорта данных.

3. МОДЕЛЬ ДАННЫХ.

3.1. Нереляционная модель данных.

Графическое представление нереляционной модели базы данных neo4j представлена на рисунке 14.

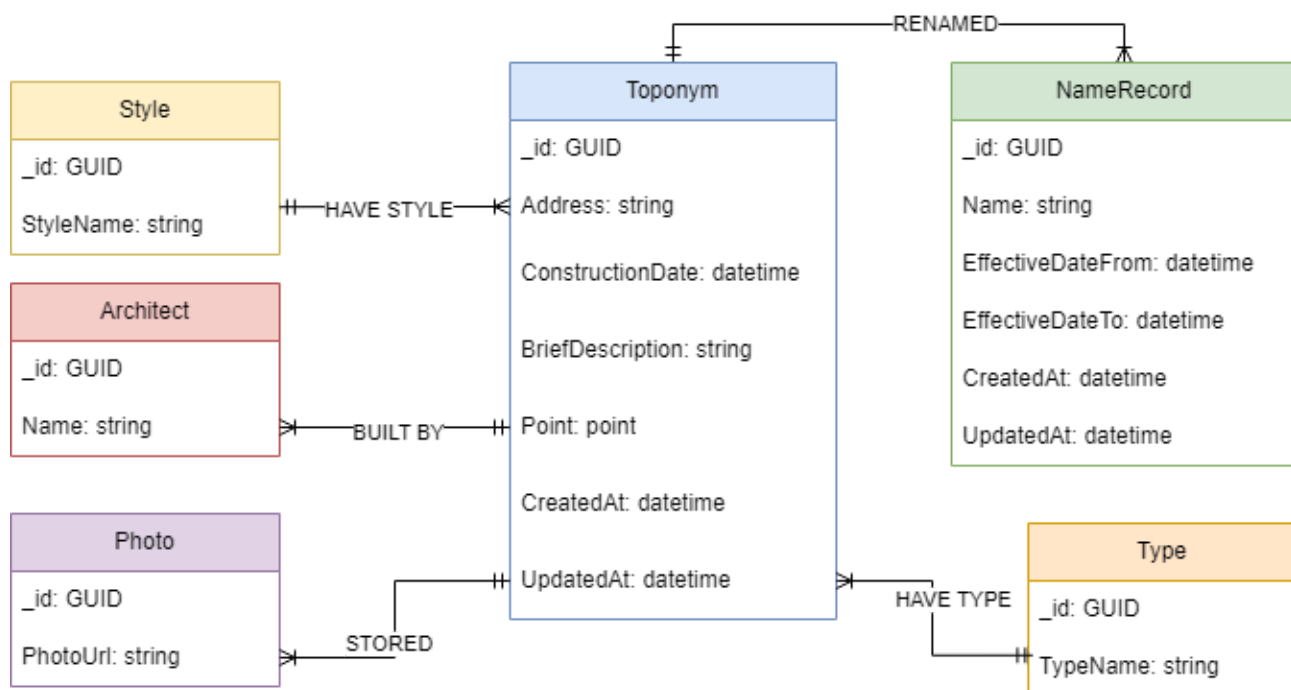


Рисунок 14 - Графическое представление нереляционной модели.

Коллекции и сущности.

Название: Toponym.

Назначение: хранение информации о топониме и ссылок на соответствующие данные.

Типы данных полей и их назначение:

- _id - идентификатор топонима, тип данных - GUID;
- Address - адрес, на котором находится топоним, тип данных - String;
- ConstructionDate - дата постройки, тип данных - datetime;
- BriefDescription - краткое описание топонима, тип данных - String;
- Point - географические координаты, тип данных - point;
- CreatedAt - время добавления топонима в БД, тип данных - datetime;
- UpdatedAt - время изменения топонима в БД, тип данных - datetime.

Название: NameRecord.

Назначение: хранение переименования топонима.

Тип данных полей и их назначение:

- `_id` - идентификатор переименования, тип данных - GUID;
- `Name` - новое название топонима, тип данных - String;
- `EffectiveDateFrom` - дата вступления переименования в силу, тип данных - datetime;
- `EffectiveDateTo` - дата отмены данного переименования, тип данных - datetime;
- `CreatedAt` - дата создания записи, тип данных - datetime;
- `UpdatedAt` - дата изменения записи, тип данных - datetime.

Название: Type.

Назначение: хранение данных о типе топонима.

Тип данных полей и их назначение:

- `_id` - идентификатор поля типа, тип данных - GUID;
- `TypeName` - название типа, тип данных - GUID.

Название: Style.

Назначение: хранение данных о стиле топонима.

Тип данных полей и их назначение:

- `_id` - идентификатор поля стиля, тип данных - GUID;
- `StyleName` - название стиля, тип данных - String.

Название: Architect.

Назначение: хранение имён архитекторов.

Тип данных полей и их назначение:

- `_id` - идентификатор архитектора, тип данных - GUID;
- `Name` - имя архитектора, тип данных - String.

Название: Photo.

Назначение: хранение фотографий топонима.

Тип данных полей и их назначение:

- `_id` - идентификатор фотографии, тип данных - GUID;

- PhotoUrl - ссылка на фотографию, тип данных - String.

Оценка удельного объёма информации

Значения размеров типов данных приведены в таблице 1.

Таблица 1 – Размеры типов данных

Тип данных	Размер
INTEGER	4 байта
FLOAT	4 байта
STRING	каждый символ – два байта
DATETIME	8 байт
POINT	8 байт

Расчёты для каждой сущности представлены ниже:

- Toponym:

$$4 + 2 \cdot n + 8 + 2 \cdot n + 8 + 8 + 8 = 36 + 4 \cdot n;$$

- Style: $4 + 2 \cdot n$;
- Architect: $4 + 2 \cdot n$;
- Photo: $4 + 2 \cdot n$;
- NameRecord: $4 + 2 \cdot n + 8 + 8 + 8 + 8 = 36 + 2 \cdot n$;
- Type: $4 + 2 \cdot n$.

В среднем топоним будет:

- иметь 4 фотографии переименован 3 раза;
- спроектирован одним человеком;
- иметь 1 стиль;
- иметь 1 тип.

Пусть количество топонимов равно x , $n = 500$ – число символов в строке. Значит, объём данных в зависимости от количества топонимов и вышеописанных данных вычисляется по следующей формуле:

$$V(x) = 12172 \cdot x$$

Избыточность данных

Избыточные данные в каждой сущности:

- Toponym: _id;
- NameRecord: _id;
- Type: _id;
- Style: _id;
- Architect: _id;
- Photo: _id.

Тогда чистый объём для каждого объекта:

- Toponym: $4 + 2 \cdot n + 8 + 2 \cdot n + 8 + 8 + 8 - 4 = 32 + 4 \cdot n$

;

- Style: $4 + n \cdot 2 - 4 = 2 \cdot n$;
- Architect: $4 + n \cdot 2 - 4 = 2 \cdot n$;
- Photo: $4 + n \cdot 2 - 4 = 2 \cdot n$;
- NameRecord: $4 + n \cdot 2 + 8 + 8 + 8 + 8 - 4 = 32 + 2 \cdot n$;
- Type: $4 + n \cdot 2 - 4 = n \cdot 2$.

Пусть количество топонимов равно x , $n = 500$ – число символов в строке. Значит, чистый объём данных в зависимости от количества топонимов и вышеописанных данных вычисляется по следующей формуле:

$$V_{clean}(x) = 12128 \cdot x$$

Избыточность выражается следующим выражением:

$$\frac{V(x)}{V_{clean}(x)} = \frac{121172}{12128} = 1.000036$$

Примеры запросов

Применение фильтров на главной странице:

```
MATCH (t:Toponym)-[:HAVE_TYPE]->(type:Type {TypeName: 'Классицизм'}),
      (t)-[:HAVE_STYLE]->(style:Style {StyleName: 'Модерн'}),
      (t)-[:BUILT_BY]->(arch:Architect {Name: 'Архитектор Иванов'}),
      (t)-[:RENAMED]->(nr:NameRecord),
      (t)-[:STORED]->(photo:Photo)
WHERE nr.EffectiveDateFrom > datetime('1900-01-01')
RETURN
  t._id AS ToponymId,
```

```

t.Address AS Address,
t.BriefDescription AS Description,
type.TypeName AS Type,
style.StyleName AS Style,
arch.Name AS Architect,
photo.PhotoUrl AS PhotoUrl,
nr.Name AS PreviousName,
nr.EffectiveDateFrom AS NameEffectiveFrom,
nr.EffectiveDateTo AS NameEffectiveTo

```

Количество запросов: 1.

Количество задействованных коллекций: 5 (Toponym, Type, Style, Architect, NameRecord, Photo).

Открытие страницы с конкретным топонимом:

```

MATCH (t:Toponym {_id: '61f0c404-5cb3-11e7-907b-a6006ad3dba1'})
OPTIONAL MATCH (t)-[:HAVE_TYPE]->(type:Type)
OPTIONAL MATCH (t)-[:HAVE_STYLE]->(style:Style)
OPTIONAL MATCH (t)-[:BUILT_BY]->(arch:Architect)
OPTIONAL MATCH (t)-[:STORED]->(photo:Photo)
OPTIONAL MATCH (t)-[:RENAMED]->(nr:NameRecord)
RETURN
  t._id AS ToponymId,
  t.Address AS Address,
  t.BriefDescription AS Description,
  type.TypeName AS Type,
  style.StyleName AS Style,
  COLLECT(arch.Name) AS Architects,
  COLLECT(photo.PhotoUrl) AS Photos,
  COLLECT({
    Name: nr.Name,
    EffectiveFrom: nr.EffectiveDateFrom,
    EffectiveTo: nr.EffectiveDateTo
  }) AS NameRecords

```

Количество запросов: 1.

Количество задействованных коллекций: 5 (Toponym, Type, Style, Architect, Photo, NameRecord).

Количество переименований конкретного топонима:

```

MATCH (t:Toponym)-[:RENAMED]->(nr:NameRecord)
RETURN
  t._id AS ToponymId,
  t.Address AS Address,
  COUNT(nr) AS RenameCount
ORDER BY RenameCount DESC

```

Количество запросов: 1.

Количество задействованных коллекций: 2 (Toponym, NameRecord).

Группировка по годам:

```
MATCH (t:Toponym)-[:RENAMED]->(nr:NameRecord)
WITH datetime(nr.EffectiveDateFrom).year AS Year, nr
RETURN
    Year,
    COUNT(nr) AS RenameCount
ORDER BY Year
```

Количество запросов: 1.

Количество задействованных коллекций: 2 (Toponym, NameRecord).

Применение запроса на странице статистики:

```
MATCH (t:Toponym)
OPTIONAL MATCH (t)-[:HAVE_STYLE]->(s:Style)
OPTIONAL MATCH (t)-[:HAVE_TYPE]->(ty:Type)
OPTIONAL MATCH (t)-[:BUILT_BY]->(a:Architect)
OPTIONAL MATCH (t)-[:STORED]->(p:Photo)
OPTIONAL MATCH (t)-[:RENAMED]->(nr:NameRecord)
WHERE
    (:typeId IS NULL OR t.TypeId = :typeId) AND
    (:styleId IS NULL OR t.StyleId = :styleId) AND
    (:hasPhoto IS NULL OR (:hasPhoto = true AND p IS NOT NULL) OR
    (:hasPhoto = false AND p IS NULL)) AND
    (:architect IS NULL OR a.Name CONTAINS :architect) AND
    (:renamedTo IS NULL OR nr.Name CONTAINS :renamedTo) AND
    (:constructionDateFrom IS NULL OR t.ConstructionDate >=
datetime(:constructionDateFrom)) AND
    (:constructionDateTo IS NULL OR t.ConstructionDate <=
datetime(:constructionDateTo)) AND
    (:searchText IS NULL OR (
        t.Address CONTAINS :searchText OR
        t.BriefDescription CONTAINS :searchText OR
        a.Name CONTAINS :searchText OR
        nr.Name CONTAINS :searchText
    ))
WITH
    CASE
        WHEN :group = 0 THEN t.ConstructionDate.year
        WHEN :group = 1 THEN s.StyleName
        WHEN :group = 2 THEN ty.TypeName
    END AS GroupField,
    COUNT(t) AS ToponymCount
RETURN
    GroupField,
    ToponymCount
ORDER BY
    CASE
        WHEN :sort = 1 THEN ToponymCount DESC
        WHEN :sort = 2 THEN ToponymCount ASC
        ELSE GroupField
```

```
END;
```

Количество запросов: 1.

Количество задействованных коллекций: 5 (Toponym, Style, Type, Architect, Photo, NameRecord).

Запрос "Архитекторы, у которых были самые быстрые переименования":

```
MATCH
(a:Architect) <-[:BUILT_BY]-(t:Toponym)-[:RENAMED]->(nr:NameRecord)
RETURN
  a.Name AS ArchitectName,
  MIN(duration.between(t.ConstructionDate,
nr.EffectiveDateFrom).days) AS MinTimeUntilRenaming
ORDER BY MinTimeUntilRenaming ASC;
```

Кол-во запросов: 1.

Кол-во задействованных коллекций: 3.

Запрос "Топонимы, которые прожили меньше всего времени":

```
MATCH (t:Toponym)-[:RENAMED]->(nr:NameRecord)
RETURN
  t.Name AS ToponymName,
  MIN(duration.between(t.ConstructionDate,
nr.EffectiveDateFrom).days) AS TimeUntilFirstRenaming
ORDER BY TimeUntilFirstRenaming ASC;
```

Кол-во запросов: 1.

Кол-во задействованных коллекций: 2.

3.2. Реляционная модель.

Графическое представление реляционной модели базы данных PostgreSQL представлена на рисунке 17:

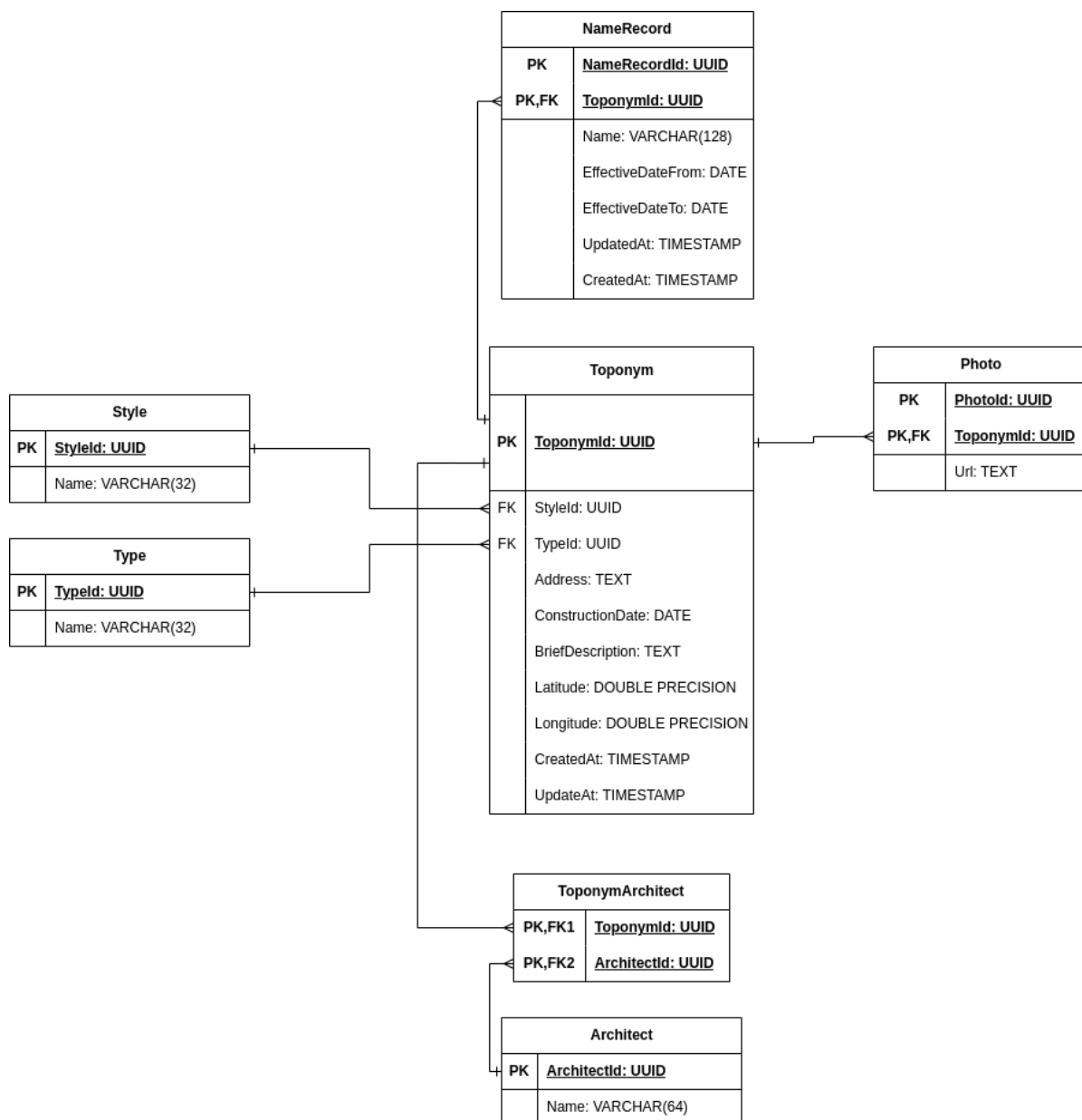


Рисунок 17 - Реляционная модель данных (PostgreSQL).

Коллекции и сущности.

Значение в скобках после типа означает его размер в байтах, n -- количество символов в переменной типа TEXT.

Топонум. Хранение информации о топониме.

```
ToponymId UUID (16) -- идентификатор топонима;  
TypeId UUID (16) -- идентификатор типа топонима;  
StyleId UUID (16) -- идентификатор стиля топонима;  
Adress TEXT (n) -- адрес топонима;  
ConstructionDate DATE (4) -- дата постройки;  
BriefDescription TEXT (n) -- краткое описание;  
Latitude DOUBLE PRECISION (8) -- широта топонима;  
Longitude DOUBLE PRECISION (8) -- долгота топонима;  
CreatedAt TIMESTAMP (8) -- время добавления топонима в БД;  
UpdatedAt TIMESTAMP (8) -- время изменения топонима в БД.
```

Style. Назначение: хранение информации о стилях топонимов.

```
StyleId UUID (16) -- идентификатор стиля топонима;  
Name VARCHAR (32) -- название стиля топонима.
```

Type. Назначение: хранение информации о типах топонимов.

```
TypeId UUID (16) -- идентификатор типа топонима;  
Name VARCHAR (32) -- название типа топонима.
```

Photo. Назначение: хранит информацию о фото топонимов.

```
PhotoId UUID (16) -- идентификатор фото топонима;  
ToponymId UUID (16) -- идентификатор топонима;  
Url TEXT (n) -- url-адрес топонима на сервере.
```

NameRecord. Назначение: хранит информацию о переименованиях топонима.

```
NameRecordId UUID (16) -- идентификатор переименования топонима;  
ToponymId UUID (16) -- идентификатор топонима;  
Name VARCHAR (128) -- имя, на которое топоним был переименован;  
EffectiveDateFrom DATE (4) -- дата вступления переименования в силу;  
EffectiveDateTo DATE (4) -- дата отмены данного переименования;  
CreatedAt TIMESTAMP (8) -- время добавления топонима в БД;  
UpdatedAt TIMESTAMP (8) -- время изменения топонима в БД.
```

ToponymArchitect. Назначение: хранит информацию об идентификаторах топонимов и соответствующих архитекторах.

```
ToponymId UUID (16) -- идентификатор топонима;  
ArchitectId UUID (16) -- идентификатор архитектора
```

Architect. Назначение: хранит информацию об архитекторах.

ArchitectId UUID (16) -- идентификатор архитектора;
Name VARCHAR (64) -- ФИО архитектора.

Оценка объема информации.

Размер одной записи в:

- Toponym: $16 \cdot 3 + 4 + 8 \cdot 4 + 2 \cdot n = 84 + 2 \cdot n$;
- Style: $16 + 32 = 48$;
- Type: $16 + 32 = 48$;
- Photo: $16 \cdot 2 + n = 32 + n$;
- NameRecord: $16 \cdot 2 + 4 \cdot 2 + 8 \cdot 2 + 128 = 184$;
- ToponymArchitect: $16 \cdot 2 = 32$;
- Architect: $16 + 64 = 80$.

В среднем каждый топоним будет:

- иметь 4 фотографии;
- переименован 3 раза;
- спроектирован одним человеком;
- иметь 1 стиль (всегда);
- иметь 1 тип (всегда);

При количестве топонимов, равным x и при $n=500$:

$$V(x) = (84 + 2 \cdot n + 48 \cdot 2 + (32 + n) \cdot 4 + 184 \cdot 3 + 32 + 80) \cdot x = (972 + 6 \cdot n) \cdot x = 3972 \cdot x.$$

Избыточность данных.

В Toponym избыточными атрибутами являются TypeId и StyleId, что составляет 32 байта.

В Photo избыточным атрибутом является ToponymId, что составляет 16 байт.

В NameRecord избыточным атрибутом является ToponymId, что составляет 16 байт.

В ToponymArchitect избыточными атрибутами являются ToponymId и ArchitectId, что составляет 32 байта.

При количестве топонимов, равным x и при $n=500$:

$$V_{\text{clean}}(x) = (52 + 2 \cdot n + 48 \cdot 2 + (32 + n - 16) \cdot 4 + (184 - 16) \cdot 3 + 0 + 80) \cdot x = (796 + 6 \cdot n) \cdot x = 3796 \cdot x$$

$$V(x) V_{\text{clean}}(x) = 3972 \cdot x 3796 \cdot x \approx 1.046$$

Направление роста модели.

При добавлении записей в таблицу Toponym будет добавляться число записей в таких таблицах, как Photo, NameRecord, ToponymArchitect. При добавлении записей в остальные таблицы ничего изменяться не будет.

Примеры запросов

Применение фильтров на главной странице

```
SELECT
    t.*,
    s.Name AS StyleName,
    ty.Name AS TypeName,
    a.Name AS ArchitectName,
    p.Url AS PhotoUrl,
    nr.Name AS RenamedToName
FROM Toponym t
LEFT JOIN Style s ON t.StyleId = s.StyleId
LEFT JOIN Type ty ON t.TypeId = ty.TypeId
LEFT JOIN ToponymArchitect ta ON t.ToponymId = ta.ToponymId
LEFT JOIN Architect a ON ta.ArchitectId = a.ArchitectId
LEFT JOIN Photo p ON t.ToponymId = p.ToponymId
LEFT JOIN NameRecord nr ON t.ToponymId = nr.ToponymId
-- Фильтр по типу
WHERE (:typeId IS NULL OR t.TypeId = :typeId)
-- Фильтр по стилю
AND (:styleId IS NULL OR t.StyleId = :styleId)
-- Фильтр по наличию фото
AND (:hasPhoto IS NULL OR (CASE WHEN :hasPhoto = TRUE THEN p.PhotoId
IS NOT NULL ELSE p.PhotoId IS NULL END))
-- Фильтр по архитектору
AND (:architect IS NULL OR a.Name ILIKE '%' || :architect || '%')
-- Фильтр по переименованию
AND (:renamedTo IS NULL OR nr.Name ILIKE '%' || :renamedTo || '%')
-- Фильтр по дате постройки (от)
AND (:constructionDateFrom IS NULL OR t.ConstructionDate >=
:constructionDateFrom)
-- Фильтр по дате постройки (до)
AND (:constructionDateTo IS NULL OR t.ConstructionDate <=
:constructionDateTo)
-- Фильтр по текстовому поиску
```



```

AND (:searchText IS NULL OR (
    t.Adress ILIKE '%' || :searchText || '%' OR
    t.BriefDescription ILIKE '%' || :searchText || '%' OR
    a.Name ILIKE '%' || :searchText || '%' OR
    nr.Name ILIKE '%' || :searchText || '%'
))
-- Пагинация: ограничение 6 карточек на странице
LIMIT 6
OFFSET :offset

```

Кол-во запросов: 1.

Кол-во задействованных коллекций: 7 (Toponym, Style, Type, ToponymArchitect, Architect, Photo, NameRecord).

Открытие страницы с конкретным топонимом

```

SELECT
    t.*,
    p.PhotoId,
    p.Url AS PhotoUrl
FROM Toponym t
LEFT JOIN Photo p ON t.ToponymId = p.ToponymId
WHERE t.ToponymId = :toponymId;

```

Кол-во запросов: 1.

Кол-во задействованных коллекций: 2 (Toponym, Photo).

Применение фильтров на странице статистики

```

SELECT
    CASE
        WHEN :group = 0 THEN EXTRACT(YEAR FROM t.ConstructionDate) --
        Группировка по году постройки
        WHEN :group = 1 THEN s.Name --
        Группировка по стилю
        WHEN :group = 2 THEN ty.Name --
        Группировка по типу
    END AS GroupField,
    COUNT(t.ToponymId) AS ToponymCount
FROM Toponym t
LEFT JOIN Style s ON t.StyleId = s.StyleId
LEFT JOIN Type ty ON t.TypeId = ty.TypeId
LEFT JOIN ToponymArchitect ta ON t.ToponymId = ta.ToponymId
LEFT JOIN Architect a ON ta.ArchitectId = a.ArchitectId
LEFT JOIN Photo p ON t.ToponymId = p.ToponymId
LEFT JOIN NameRecord nr ON t.ToponymId = nr.ToponymId
-- Фильтры с левой стороны страницы
AND (:typeId IS NULL OR t.TypeId = :typeId)
AND (:styleId IS NULL OR t.StyleId = :styleId)
AND (:hasPhoto IS NULL OR (CASE WHEN :hasPhoto = TRUE THEN

```

```

p.PhotoId IS NOT NULL ELSE p.PhotoId IS NULL END))
  AND (:architect IS NULL OR a.Name ILIKE '%' || :architect || '%')
  AND (:renamedTo IS NULL OR nr.Name ILIKE '%' || :renamedTo ||
'%')
  AND (:constructionDateFrom IS NULL OR t.ConstructionDate >=
:constructionDateFrom)
  AND (:constructionDateTo IS NULL OR t.ConstructionDate <=
:constructionDateTo)
  AND (:searchText IS NULL OR (
    t.Adress ILIKE '%' || :searchText || '%' OR
    t.BriefDescription ILIKE '%' || :searchText || '%' OR
    a.Name ILIKE '%' || :searchText || '%' OR
    nr.Name ILIKE '%' || :searchText || '%'
  ))
-- Группировка по выбранному параметру
GROUP BY
  CASE
    WHEN :group = 0 THEN EXTRACT(YEAR FROM t.ConstructionDate) --
Группировка по году постройки
    WHEN :group = 1 THEN s.Name --
Группировка по стилю
    WHEN :group = 2 THEN ty.Name --
Группировка по типу
  END
-- Сортировка по выбранному параметру
ORDER BY
  CASE
    WHEN :sort = 1 THEN COUNT(t.ToponymId) DESC --
Сортировка по убыванию
    WHEN :sort = 2 THEN COUNT(t.ToponymId) ASC --
Сортировка по возрастанию
    ELSE NULL --
По умолчанию без сортировки
  END;

```

Кол-во запросов: 1.

Кол-во задействованных коллекций: 7 (Toponym, Style, Type, ToponymArchitect, Architect, Photo, NameRecord).

Запрос для сценария "Количество переименований"

```

SELECT
  t.ToponymId AS ToponymId,
  t.Name AS ToponymName,
  COUNT(nr.Name) AS RenamingCount
FROM Toponym t
LEFT JOIN NameRecord nr ON t.ToponymId = nr.ToponymId
GROUP BY t.ToponymId, t.Name
ORDER BY
  CASE
    WHEN :sort = 1 THEN COUNT(nr.Name) DESC -- Сортировка по

```

```

количеству переименований по убыванию
      WHEN :sort = 2 THEN COUNT(nr.Name) ASC      -- Сортировка по
количеству переименований по возрастанию
      ELSE NULL                                    -- Без сортировки
END;

```

Кол-во запросов: 1.

Кол-во задействованных коллекций: 2 (Toponym, NameRecord).

Запрос для сценария "Группировка переименований по годам"

```

SELECT
    EXTRACT(YEAR FROM nr.RenamedDate) AS Year,
    COUNT(*) AS RenamingCount
FROM NameRecord nr
WHERE nr.RenamedDate IS NOT NULL
GROUP BY EXTRACT(YEAR FROM nr.RenamedDate)
ORDER BY
    CASE
        WHEN :sort = 1 THEN COUNT(*) DESC      -- Сортировка по
количеству переименований по убыванию
        WHEN :sort = 2 THEN COUNT(*) ASC      -- Сортировка по
количеству переименований по возрастанию
        ELSE NULL                                -- Без сортировки
    END;

```

Кол-во запросов: 1.

Кол-во задействованных коллекций: 1 (NameRecord).

Запрос "Архитекторы, у которых были самые быстрые переименования"

```

SELECT
    a.Name AS ArchitectName,
    MIN(nr.EffectiveDateFrom - t.ConstructionDate) AS
MinTimeUntilRenaming
FROM Toponym t
JOIN ToponymArchitect ta ON t.ToponymId = ta.ToponymId
JOIN Architect a ON ta.ArchitectId = a.ArchitectId
JOIN NameRecord nr ON t.ToponymId = nr.ToponymId
WHERE t.ConstructionDate IS NOT NULL
GROUP BY a.Name
ORDER BY MinTimeUntilRenaming ASC

```

Кол-во запросов: 1.

Кол-во задействованных коллекций: 4.

Запрос "Топонимы, которые прожили меньше всего времени"

```

SELECT
    t.Name AS ToponymName,

```

```
MIN(nr.EffectiveDateFrom - t.ConstructionDate) AS  
TimeUntilFirstRenaming  
FROM Toponym t  
JOIN NameRecord nr ON t.ToponymId = nr.ToponymId  
GROUP BY t.ToponymId, t.Name, t.ConstructionDate  
ORDER BY TimeUntilFirstRenaming ASC;
```

3.3. Сравнение моделей

Удельный объем информации

Реляционная модель имеет большой удельный объем информации по сравнению с графовой моделью.

Сравнения для моделей

- Toponym: 3972 байт (PostgreSQL) vs 12172 байт (Neo4j)
- Style: 48 байт (PostgreSQL) vs 1008 байт (Neo4j)
- Type: 48 байт (PostgreSQL) vs 1008 байт (Neo4j)
- Photo: 532 байт (PostgreSQL) vs 1008 байт (Neo4j)
- NameRecord: 184 байт (PostgreSQL) vs 1032 байт (Neo4j)
- ToponymArchitect: 32 байта (PostgreSQL) vs 0 байт (Neo4j) (в графовой модели связи хранятся отдельно и не учитываются в удельном объеме каждой сущности)
- Architect: 80 байт (PostgreSQL) vs 1008 байт (Neo4j)

Запросы по отдельным юзкейсам.

Сравнения количества запросов для отдельных юзкейсов

- Применение фильтров на главной странице: 1 запрос (Neo4j) vs 1 запрос (PostgreSQL)
- Открытие страницы с конкретным топонимом: 1 запрос (Neo4j) vs 1 запрос (PostgreSQL)
- Количество переименований конкретного топонима: 1 запрос (Neo4j) vs 1 запрос (PostgreSQL)
- Группировка по годам: 1 запрос (Neo4j) vs 1 запрос (PostgreSQL)

Количество задействованных коллекций

- Применение фильтров на главной странице: 5 (Neo4j) vs 7 (PostgreSQL)
- Открытие страницы с конкретным топонимом: 5 (Neo4j) vs 2 (PostgreSQL)
- Количество переименований конкретного топонима: 2 (Neo4j) vs 2 (PostgreSQL)
- Группировка переименований по годам: 2 (Neo4j) vs 1 (PostgreSQL)
- Применение запроса на странице статистики: 5 (Neo4j) vs 7 (PostgreSQL)

Вывод

На основе проведенного анализа можно сделать следующие выводы:

- Удельный объем информации: Реляционная модель обладает меньшим удельным объемом информации, что делает её более эффективной с точки зрения использования памяти и хранения данных.
- Количество запросов: Оба подхода показывают схожие результаты по количеству запросов для выполнения основных юзкейсов. Однако графовая модель может предложить преимущества в скорости выполнения сложных запросов благодаря естественной поддержке связей.
- Количество задействованных коллекций: Графовая модель требует меньше коллекций для выполнения отдельных юзкейсов, что может способствовать более простой и быстрой работе с данными. В реляционной модели требуется больше таблиц, что может усложнить структуру базы данных и увеличить накладные расходы на управление таблицами.

Несмотря на то, что Neo4j может облегчить некоторые задачи за счёт естественного представления связей между данными, это также может привести к увеличению объёма данных и усложнению модели. Дополнительное хранение связей может увеличить общий объем хранимой информации.

Таким образом, реляционная модель предпочтительнее для данного проекта с точки зрения эффективности использования ресурсов и меньшего объема данных. Однако, если проект требует выполнения сложных запросов, связанных

с множественными связями между данными, графовая модель предоставит преимущества в производительности и простоте написания запросов.

4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

4.1. Краткое описание приложения.

Фронтенд создан на Angular и взаимодействует с серверной частью через GET- и POST-запросы, используя формат JSON. Полученные данные удобно отображаются на экране в виде таблицы.

Серверная часть написана на Python с использованием Flask и предоставляет API для обработки GET- и POST-запросов. GET-запросы позволяют получать информацию о кредитах или пользователях, а POST-запросы — отправлять данные, например, запрос на импорт информации.

Для удобства развертывания и обеспечения изоляции все части приложения упакованы в контейнеры с помощью Docker и docker-compose. Этот подход упрощает запуск приложения в разных средах.

4.2. Используемые технологии.

Frontend: Angular 18.

Backend: Flask (Python).

DB: Neo4j.

4.3. Схема экранов приложения.

На рис.1-13 представлены схема экранов приложения.

5. ВЫВОДЫ

5.1. Достигнутые результаты.

По итогу выполнения работы реализовано веб-приложение для просмотра, поиска, импорта и экспорта переименованных в советское время топонимов в СПб.

Реализована главная страница с таблицей, отображающая топонимы и главную информацию о них, включая фото. У пользователя есть возможность фильтрации и поиска по всем полям.

Разработанное приложение поддерживает импорт/экспорт базы данных в формате json, обеспечивая возможность дополнительного контроля.

5.2. Недостатки и пути для улучшения полученного решения.

На данный момент приложение имеет недостатки, устранение которых позволит улучшить пользовательский опыт и функциональность приложения:

1. Отсутствие дополнительных экранов, кроме главного.

Приложение имеет только главную страницу. На ней отображена таблица с пагинацией, в которой отображена информация о топонимах, и блок с фильтрами слева. Приложение можно дополнить заложенными в макет экранами, чтобы улучшить пользовательский опыт и расширить функциональность.

2. Отсутствие системы пользователей

В текущей версии у приложения ограниченная область применения ввиду отсутствия системы пользователей (гость/администратор/пользователь и пр.). Реализация такой системы сделает её более гибкой и позволит совершить больше доработок.

3. Отсутствие возможности редактировать данные с помощью UI

В данный момент в приложении нельзя редактировать данные о топонимах напрямую (через кнопки/поля ввода), только с помощью кнопки “Импорт”. Хотя это и удобно для массового редактирования базы - это не очень удобно с пользовательской точки зрения.

5.3. Будущее развитие решения.

В перспективе планируется расширение приложения за счёт добавления дополнительных экранов и разделов: появятся отдельные страницы для более детального отображения информации о топонимах, а также интерфейс личного кабинета и/или панели администратора для управления данными и настройками. Планируется добавить страницу с интерактивной картой, на которой можно смотреть топонимы. Кроме того, планируется разработать систему пользователей с разными уровнями доступа (например, гость, пользователь и администратор), что повысит гибкость и безопасность при работе с данными. Наконец, в приложении появится возможность редактирования информации о топонимах напрямую в пользовательском интерфейсе, дополняя уже доступную возможность массового импорта.

6. ПРИЛОЖЕНИЯ

6.1. Документация по сборке и развертыванию приложения.

1. Склонировать репозиторий с проектом [1] по ссылке: <https://github.com/moevm/nosql2h24-rename>.
2. Перейти в корневую директорию проекта.
3. Собрать приложение с помощью команды: `docker-compose build --no-cache`.
4. Запустить приложение с помощью команды: `docker-compose up`.
5. Открыть приложение в браузере по адресу `http://127.0.0.1:80`.

6.2. Инструкция для пользователя.

- Просмотр топонимов

На главной странице слева выберите параметры для отображения топонимов в таблице. Доступна фильтрация по типу топонима, стилю, наличию фото, имени архитектора, словам в карточке, адресу, названию, дате постройки и дате переименования. Фильтры применяются автоматически после выбора (нет кнопки “применить”). Фильтры полностью комбинируемы между собой. Перемещаться между страницами таблицы можно с помощью кликабельной нумерации страниц под таблицей.

- Импорт/экспорт данных

На главной странице нажмите кнопку "Импортировать". Выберите или перетяните файл в поддерживаемом формате (.json). Для экспорта данных нажмите кнопку "Экспортировать". Данные будут выгружены в файл с информацией о топонимах в формате (.json).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ссылка на код проекта на GitHub. – [Электронный ресурс]. – URL: <https://github.com/moevm/nosql2h24-rename>
2. Документация Flask. – [Электронный ресурс]. – URL: <https://flask.palletsprojects.com/en/latest/> (дата обращения 22.12.2024).
3. Документация Angular. – [Электронный ресурс]. – URL: <https://angular.io/docs> (дата обращения 22.12.2024).
4. Документация Neo4j. – [Электронный ресурс]. – URL: <https://neo4j.com/docs/> (дата обращения 22.12.2024).
5. Документация Taiga UI. – [Электронный ресурс]. – URL: <https://taiga-ui.dev/> (дата обращения 22.12.2024).