

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

**ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ
по дисциплине «Разработка ПО информационных
систем»**

Тема: Сервис истории домов Санкт-Петербурга

Студенты гр. 1303 и 1304

Герасименко Я.Д.

Чернякова А.Д.

Клепнев Д.А.

Преподаватель

Заславский М.М.

Санкт-Петербург

ЗАДАНИЕ

Студенты

Герасименко Я.Д.

Чернякова А.Д.

Клепнев Д.А.

Группа 1303 и 1304

Тема проекта: Разработка сервиса истории домов Санкт-Петербурга.

Исходные данные:

Необходимо реализовать собственный вариант Citywalls с большим упором на классификацию, выделение кластеров и визуализацию для СУБД Arango DB.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Качественные требования к решению»

«Сценарий использования»

«Модель данных»

«Разработка приложения»

«Вывод»

«Приложение»

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студенты гр. 1303 и 1304

Герасименко Я.Д.

Чернякова А.Д.

Клепнев Д.А.

Преподаватель

Заславский М.М.

АННОТАЦИЯ

В рамках данного курса предполагалось разработать какое-либо приложение в команде на одну из поставленных тем. Была выбрана тема создания приложения аналог Citywalls для СУБД Arango DB. Во внимание будут приниматься такие аспекты как классификация, выделение кластеров и визуализация. Найти исходный код и всю дополнительную информацию можно по ссылке: <https://github.com/moevm/nosql2h24-history>

ANNOTATION

As part of this course, it was supposed to develop an application in a team on one of the given topics. The topic of creating an application similar to Citywalls for the Arango DB DBMS was chosen. Aspects such as classification, clustering and visualization will be taken into account. You can find the source code and all additional information at the link: <https://github.com/moevm/nosql2h24-history> .

Оглавление

1.	Введение.....	7
2.	Качественные требования к решению.....	7
3.	Сценарии использования.....	7
4.	Модель данных.....	12
5.	Разработанное приложение.....	23
6.	Вывод.....	24
7.	Приложения.....	24
8.	Используемая литература.....	24

1. Введение

Цель работы – создать сервис истории домов Санкт-Петербурга (собственный вариант Citywalls).

Было решено разработать веб-приложение, которое позволит удобно отображать и фильтровать по запросам пользователя дома и улицы Санкт-Петербурга.

2. Качественные требования к решению

Требуется разработать приложение с использованием СУБД Arango DB.

3. Сценарии использования

Макеты UI

1. Экран импорта/экспорта (первая страница, которую видит пользователь)(Рис. 1).

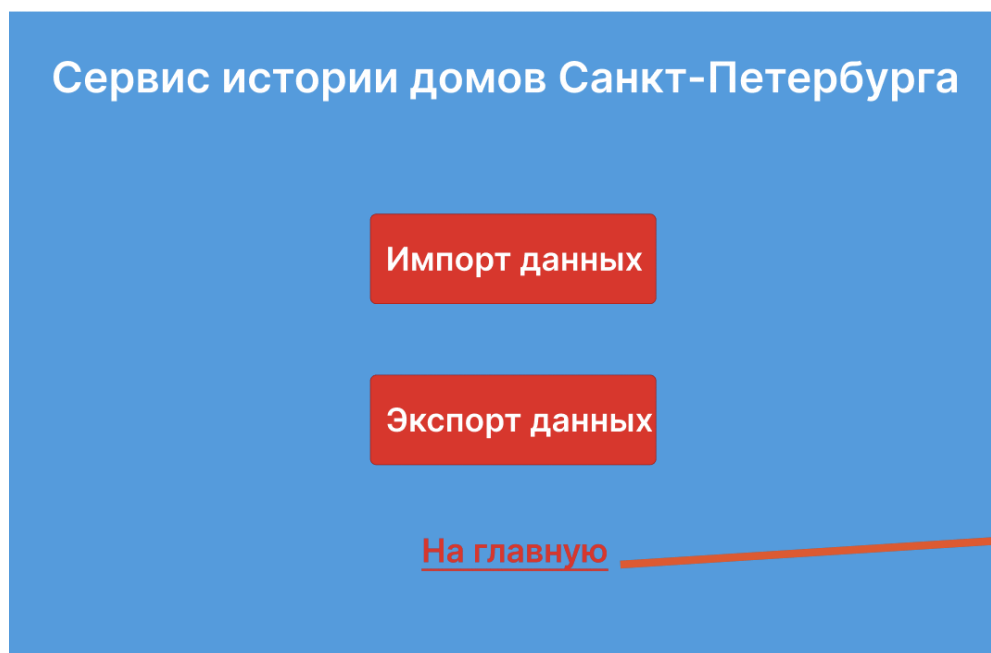


Рисунок 1. Экран импорта/экспорта.

2. Главный экран (Рис. 2).

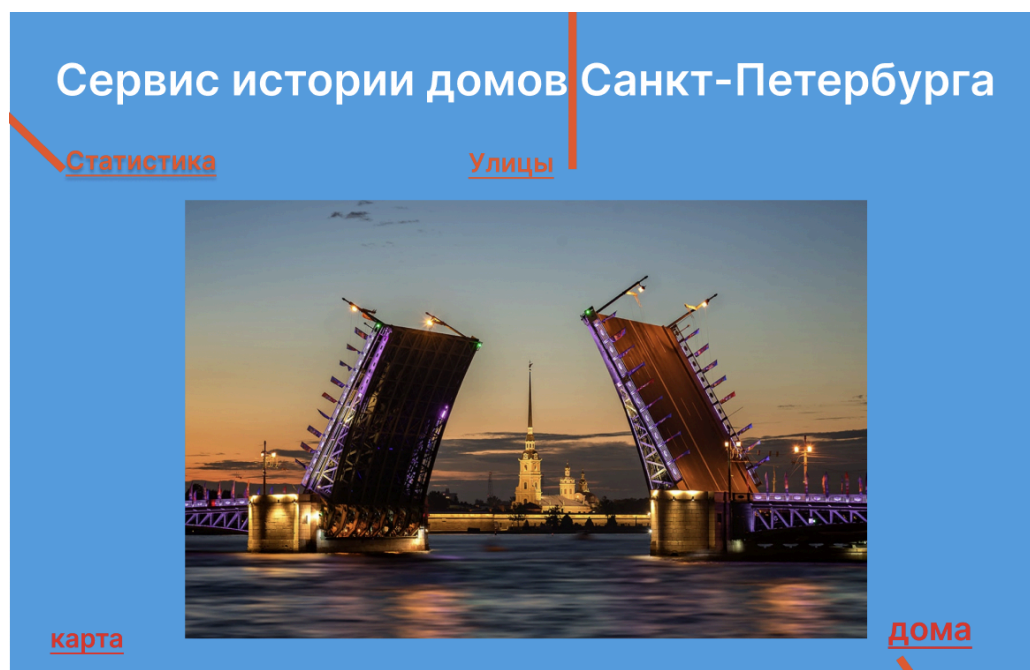


Рисунок 2. Главный экран.

3. Экран с фильтрацией улиц (Рис. 3).

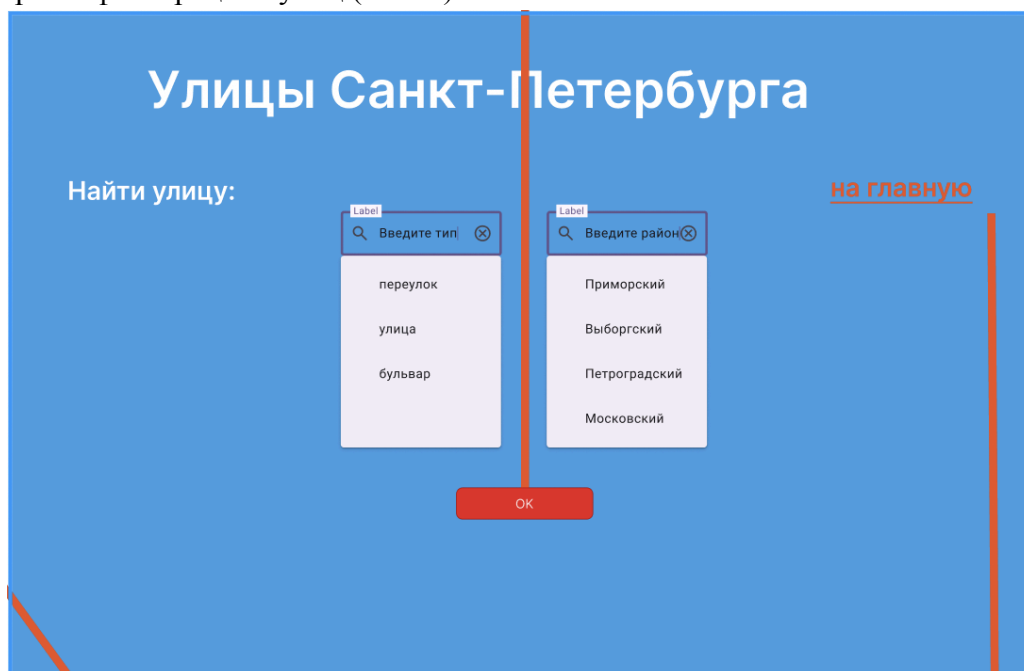


Рисунок 3. Экран с фильтрацией улиц

4. Экран со списком отфильтрованных улиц (Рис. 4).

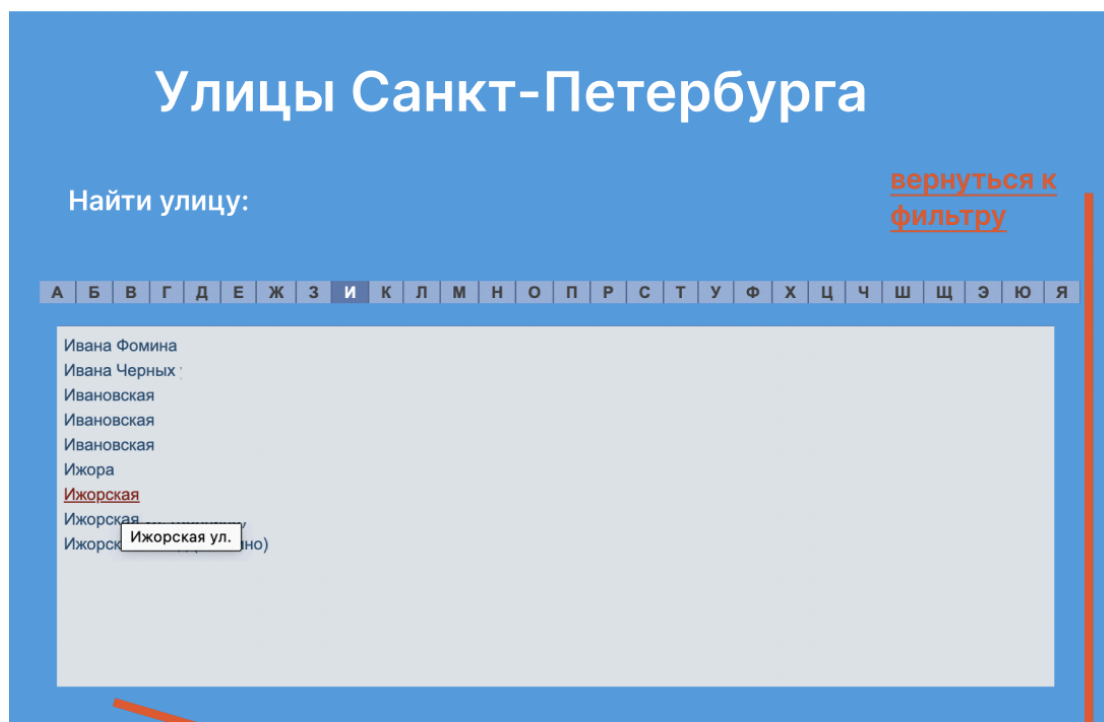


Рисунок 4. Экран со списком отфильтрованных улиц.

5. Экран с домами конкретной улицы



Рисунок 5. Экран с домами конкретной улицы.

6. Экран карточки дома

Адрес дома: пр-кт Авиаконструкторов, 2 Литер А



Год: 2008
Количество этажей: 25
Количество квартир: 493
Состояние: исправный
УК: ТСЖ "Авиатор-2"
Серия, тип: индивидуальный
Район: Петроградский

К списку домов на улице Ижорская по

Рисунок 6. Экран карточки дома.

7. Экран фильтрации домов

Дома Санкт-Петербурга

[На главную](#)

Выберите фильтры для поиска домов:

Введите год постройки

- до 1900
- 1900-1950
- 1950-2000
- 2000-2024

Введите район

- Петроградский
- Приморский
- Выборгский
- Московский

Введите количество этажей

Value

Введите УК

Value

Введите количество квартир

Value

Введите состояние дома

Value

Введите улицу

Value

OK

Рисунок 7. Экран фильтрации домов.

8. Экран отфильтрованных домов



Дома Санкт-Петербурга								
вернуться к фильтру								
Фото	Адрес	Год	Количество этажей	Количество квартир	Состояние	УК	Серия, тип	Район
	пр-кт Авиакон структор ов, 1 Литер А	1988	16	463	Исправн ый	УК «ЖКС №4 приморс кого района»	137	Приморский
	пр-кт Авиакон структор ов, 2 Литер А	2008	25	493	Исправн ый	ТСЖ «Авиато р-2»	Индивид уальный	Петроградский

Рисунок 8. Экран отфильтрованных домов.

Описание сценариев использования

Единственная роль в системе – незарегистрированный пользователь.

Пользователь входит на страницу. Он видит две кнопки (импорт, экспорт) и ссылку для перехода “на главную страницу”. При успешном импорте у пользователя всплывет соответствующее сообщение, при экспорте все данные из БД скачаются на устройство пользователя как один файл exported.json.

При клике на “главную” пользователь окажется на главной странице, с которой он может попасть на страницу улиц или домов.

При клике на “дома” пользователь окажется на странице с фильтрами для отображения списка домов. Пользователь указывает необходимые параметры: год постройки, район, количество этажей, количество квартир, улица, УК, состояние дома. Далее отобразится список домов под запрос пользователя в виде таблицы. Если домов подходящих под данные критерии нет, то будет написано соответствующее сообщение. В таблице адрес дома является ссылкой для перехода на страницу этого дома. Страница дома содержит фотографию и информацию о данном доме, а также ссылку для перехода к списку домов данной улицы.

При клике на “улицы” пользователь окажется на странице с фильтрами для отображения списка улиц. Пользователь указывает необходимые параметры: тип улицы и район. Далее

отобразится список улиц под запрос пользователя. Если домов подходящих под данные критерии нет, то будет написано соответствующее сообщение. Улица является ссылкой для перехода к таблице домов данной улицы. В таблице адрес дома является ссылкой для перехода на страницу этого дома. Страница дома содержит фотографию и информацию о данном доме, а также ссылку для перехода к списку домов данной улицы.

4. Модель данных

Нереляционная модель данных

Нереляционная модель данных представлена на рисунке 8.

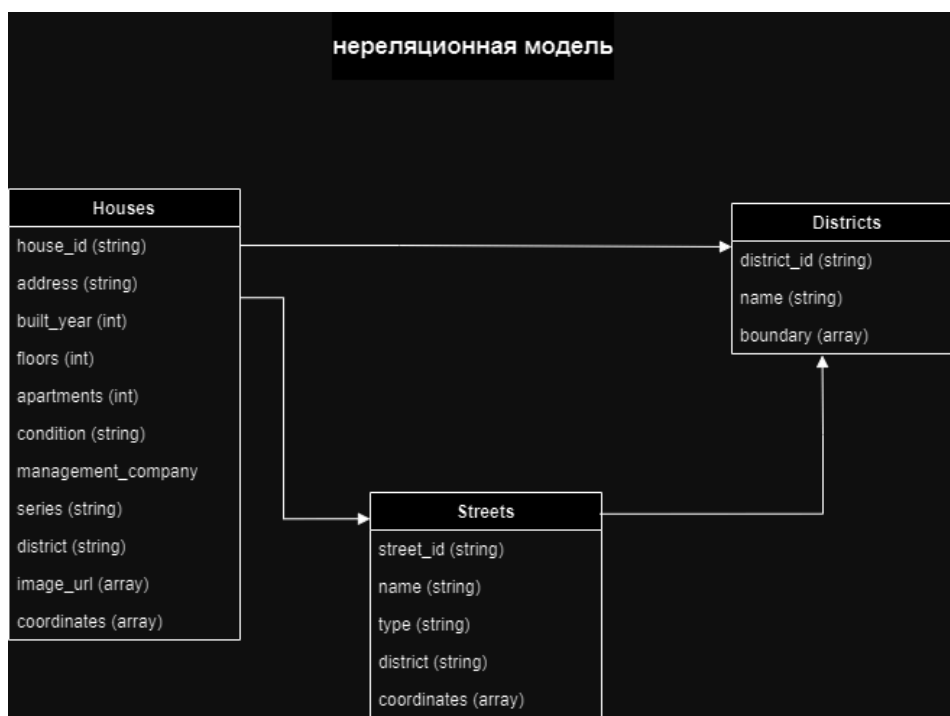


Рисунок 8. Нереляционная модель данных

Коллекция houses

- Назначение: Хранение сведений о домах, предоставление информации для поиска и фильтрации по улице, району, году постройки, состоянию и другим параметрам.
- Тип данных: Объект с вложенными объектами.
- Сущности:
 - house_id (строка) — уникальный идентификатор дома.
 - address (объект) — содержит street_id (ID улицы) и number (номер дома).
 - built_year (целое число) — год постройки дома.

- floors (целое число) — количество этажей.
- apartments (целое число) — количество квартир.
- condition (строка) — состояние дома.
- management_company (строка) — управляющая компания дома.
- series (строка) — серия или тип дома.
- district (строка) — район, в котором расположен дом.
- image_url (массив строк с форматом URI) — массив ссылок на фотографии дома.
- coordinates (массив из двух чисел) — условная центральная точка дома в формате [широта, долгота].

Коллекция streets

- Назначение: Позволяет фильтровать дома по улицам и районам, поддерживает статистику улиц.
- Тип данных: Объект.
- Сущности:
 - street_id (строка) — уникальный идентификатор улицы.
 - name (строка) — название улицы.
 - type (строка) — тип улицы (например, переулок, улица, бульвар, проезд, тупик, линия, проспект, шоссе, набережная, площадь, аллея, трасса).
 - district (строка) — район, в котором расположена улица.
 - coordinates (массив массивов из двух чисел) — координаты ломаной линии, задающей улицу, в формате [широта, долгота].

Коллекция districts

- Назначение: Хранение информации о районах Санкт-Петербурга, включая их границы и возможность отображения на карте. Используется для фильтрации домов и улиц по районам.
- Тип данных: Объект с массивами.
- Сущности:
 - district_id (строка) — уникальный идентификатор района.
 - name (строка) — название района.
 - boundary (массив массивов из двух чисел) — границы района, заданные массивом точек в формате [широта, долгота].

Оценка объема информации, хранимой в модели

Для оценки объема информации, хранимой в модели данных, рассмотрим каждый из типов объектов (домов и улиц) и оценим размер их хранения в памяти. Затем выразим общий объем через переменную, представляющую количество объектов одного типа.

Шаг 1: Оценка объема хранения для одного объекта

1.1. Коллекция houses

- Фактический объем хранения для одного объекта:
 - house_id: 36 байт
 - address:
 - street_id: 24 байта
 - number: 10 байт
 - built_year: 4 байта
 - floors: 4 байта
 - apartments: 4 байта
 - condition: 20 байт
 - management_company: 30 байт
 - series: 15 байт
 - district: 20 байт
 - image_url: 100 байт
 - coordinates: 16 байт (2 числа по 8 байт для широты и долготы)

Общий объем для одного объекта house:

$$V_{\text{house}} = 36 + 24 + 10 + 4 + 4 + 4 + 20 + 30 + 15 + 20 + 100 + 16 = 283 \text{ байт}$$

Чистый объем хранения для одного объекта house (минимально необходимые поля):

- house_id: 36 байт
- address (номер дома и ID улицы): 24 байта + 10 байт
- coordinates: 16 байт

$$V_{\text{houseclean}} = 36 + 24 + 10 + 16 = 86 \text{ байт}$$

1.2. Коллекция streets

- Фактический объем хранения для одного объекта:
 - street_id: 36 байт
 - name: 30 байт
 - type: 10 байт
 - district: 20 байт
 - coordinates: 128 байт (напр., массив из 8 точек, каждая по 16 байт)

Общий объем для одного объекта street:

$$V_{\text{street}} = 36 + 30 + 10 + 20 + 128 = 224 \text{ байт}$$

Чистый объем хранения для одного объекта street (минимально необходимые поля):

- street_id: 36 байт
- name: 30 байт
- coordinates: 128 байт

$$V_{\text{streetclean}} = 36 + 30 + 128 = 194 \text{ байт}$$

1.3. Коллекция districts

- Фактический объем хранения для одного объекта:
 - district_id: 36 байт
 - name: 30 байт
 - coordinates: 256 байт (напр., массив из 16 точек, каждая по 16 байт)

Общий объем для одного объекта district:

$$V_{\text{district}} = 36 + 30 + 256 = 322 \text{ байт}$$

Чистый объем хранения для одного объекта district (минимально необходимые поля):

- district_id: 36 байт
- coordinates: 256 байт

$$V_{\text{districtclean}} = 36 + 256 = 292 \text{ байт}$$

Шаг 2: Общий объем хранения в зависимости от количества объектов

Пусть количество домов (houses) — N_h , количество улиц (streets) — N_s , количество районов (districts) — N_d .

Общий объем хранения данных:

$$V_{\text{total}} = N_h \cdot V_{\text{house}} + N_s \cdot V_{\text{street}} + N_d \cdot V_{\text{district}}$$

Подставляя значения:

$$V_{\text{total}} = N_h \cdot 283 + N_s \cdot 224 + N_d \cdot 322$$

Примеры запросов.

Запрос для выгрузки всех домов в формате JSON

```
FOR house IN houses  
RETURN house
```

Запрос для массового добавления данных в коллекцию houses

```
INSERT {  
  house_id: @house_id,  
  address: @address,  
  built_year: @built_year,  
  floors: @floors,  
  apartments: @apartments,  
  condition: @condition,  
  management_company: @management_company,  
  series: @series,  
  district: @district,  
  image_url: @image_url
```



```
} INTO houses
```

Запрос на получение списка улиц по фильтру

```
FOR street IN streets
  FILTER street.type == @type AND street.district == @district
  RETURN street
```

Запрос на получение домов на выбранной улице

```
FOR house IN houses
  FILTER house.address.street_id == @street_id
  RETURN house
```

Запрос на получение информации о конкретном доме

```
FOR house IN houses
  FILTER house.house_id == @house_id
  RETURN {
    "built_year": house.built_year,
    "floors": house.floors,
    "apartments": house.apartments,
    "condition": house.condition,
    "management_company": house.management_company,
    "series": house.series,
    "district": house.district,
    "image_url": house.image_url
  }
```

Запрос на получение домов по параметрам фильтра

```
FOR house IN houses
  FILTER house.built_year == @built_year
  AND house.district == @district
  AND house.floors == @floors
  AND house.apartments == @apartments
```

```
AND house.address.street_id == @street_id
AND house.management_company == @management_company
AND house.condition == @condition
RETURN house
```

Запрос на получение домов с заданными параметрами для отображения на карте

```
FOR house IN houses
FILTER house.built_year == @built_year
  AND house.district == @district
  AND house.floors == @floors
  AND house.apartments == @apartments
  AND house.address.street_id == @street_id
  AND house.management_company == @management_company
  AND house.condition == @condition
RETURN {
  "coordinates": house.coordinates,
  "house_id": house.house_id,
  "description": {
    "built_year": house.built_year,
    "floors": house.floors,
    "apartments": house.apartments,
    "condition": house.condition,
    "management_company": house.management_company
  }
}
```

Запрос на получение статистики по улицам

```
FOR street IN streets
COLLECT type = street.type WITH COUNT INTO count
RETURN { "type": type, "count": count }
```

Запрос на получение статистики по домам

```
FOR house IN houses
COLLECT year = house.built_year WITH COUNT INTO count
```

```
RETURN { "built_year": year, "count": count }
```

Запрос на получение статистики: сколько домов на какой улице было построено в каком десятилетии

```
FOR house IN houses
```

```
LET decade = FLOOR(house.built_year / 10) * 10
```

```
COLLECT street_id = house.address.street_id, decade INTO grouped
```

```
LET count = LENGTH(grouped)
```

```
RETURN {
```

```
  "street_id": street_id,
```

```
  "decade": decade,
```

```
  "house_count": count
```

```
}
```

Реляционная модель данных

Реляционная модель данных представлена на рисунке 9.

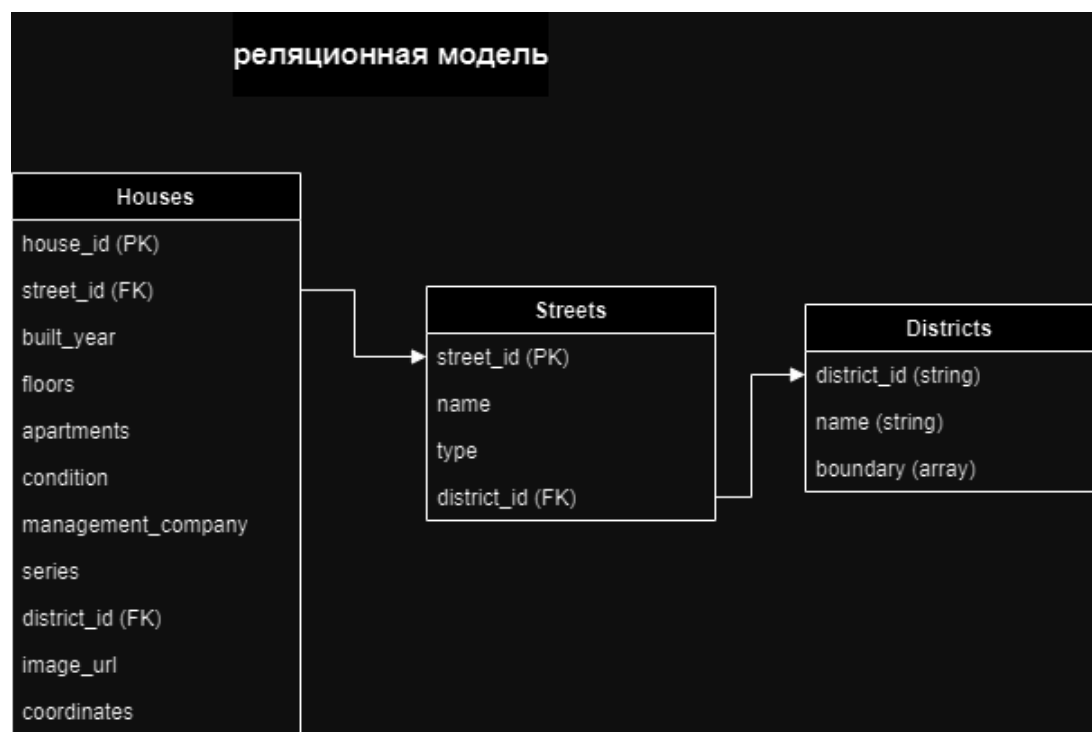


Рисунок 9. Реляционная модель данных

ER-диаграмма для реляционной модели, включающая основные сущности houses и streets, а также связи между ними:

- **Таблица houses:**

- Поля:

- house_id (PRIMARY KEY)
 - street_id (FOREIGN KEY ссылается на streets)
 - number
 - built_year
 - floors
 - apartments
 - condition
 - management_company
 - series
 - image_url
 - coordinates

- **Таблица streets:**

- Поля:

- street_id (PRIMARY KEY)
 - name
 - type
 - district_id (FOREIGN KEY)
 - coordinates

- **Таблица districts:**

- Поля:

- district_id (PRIMARY KEY)
 - name
 - boundary

Описание назначений таблиц, типов данных и сущностей

- **Таблица houses**

- Назначение: Хранение информации о домах с привязкой к улицам, состоянию, управляющей компании и т. д.
 - Типы данных: Идентификаторы и строки, целые числа, ссылки.
 - Сущности:

- house_id (строка) — уникальный идентификатор дома.
 - street_id (строка) — ID улицы.
 - number (строка) — номер дома.
 - built_year (целое число) — год постройки.
 - floors (целое число) — этажность.
 - apartments (целое число) — количество квартир.
 - condition (строка) — состояние.
 - management_company (строка) — управляющая компания.
 - series (строка) — серия или тип.
 - image_url (массив строк) — массив ссылок на изображения.
 - coordinates (массив из двух чисел) — условная центральная точка дома в формате [широта, долгота].
- Таблица streets
 - Назначение: Хранение информации о улицах.
 - Типы данных: Идентификаторы, строковые значения.
 - Сущности:
 - street_id (строка) — уникальный идентификатор улицы.
 - name (строка) — название.
 - type (строка) — тип улицы.
 - district_id (строка) — ID района, в котором расположена улица.
 - coordinates (массив массивов из двух чисел) — координаты ломаной линии, задающей улицу, в формате [широта, долгота]
 - Таблица districts
 - Назначение: Хранение информации о районах города, включая их границы, используемое для фильтрации домов и улиц по районам.
 - Типы данных: Идентификаторы, строки, массивы.
 - Сущности:
 - district_id (строка) — уникальный идентификатор района.
 - name (строка) — название района.
 - boundary (массив массивов из двух чисел) — границы района, заданные массивом точек в формате [широта, долгота]

Оценка объема информации, хранимой в модели

Аналогично нереляционной модели.

Примеры запросов.

Выгрузка всех данных из таблицы houses

```
SELECT * FROM houses;
```

Вставка новых данных в таблицу houses

```
INSERT INTO houses (house_id, street_id, number, built_year, floors, apartments, condition,  
management_company, series, image_url, coordinates)  
VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);
```

Получение списка улиц по фильтру

```
SELECT * FROM streets  
WHERE type = ? AND district_id = ?;
```

Получение списка домов на выбранной улице

```
SELECT * FROM houses  
WHERE street_id = ?;
```

Получение информации о конкретном доме

```
SELECT * FROM houses  
WHERE house_id = ?;
```

Получение списка домов по заданным параметрам

```
SELECT h.*  
  
FROM houses h  
  
JOIN streets st ON h.street_id = st.street_id  
  
WHERE (h.built_year = ? OR ? IS NULL)  
  
AND (st.district_id = ? OR ? IS NULL)
```

AND (h.floors = ? OR ? IS NULL)

AND (h.apartments = ? OR ? IS NULL)

AND (h.street_id = ? OR ? IS NULL)

AND (h.management_company = ? OR ? IS NULL)

AND (h.condition = ? OR ? IS NULL);

Получение списка домов с заданными параметрами для отображения на карте

SELECT

h.house_id,

CONCAT(st.name, ' ', h.number) AS address,

h.built_year,

h.floors,

h.apartments,

h.condition,

h.management_company,

h.series,

st.district_id,

h.image_url

FROM

houses h

JOIN

streets st ON h.street_id = st.street_id

WHERE

(h.built_year = ? OR ? IS NULL)

AND (st.district_id = ? OR ? IS NULL)

AND (h.floors = ? OR ? IS NULL)

AND (h.apartments = ? OR ? IS NULL)

AND (h.street_id = ? OR ? IS NULL)

AND (h.management_company = ? OR ? IS NULL)

AND (h.condition = ? OR ? IS NULL);

Получение статистики по улицам

SELECT COUNT(street_id), type, district_id

FROM streets

GROUP BY type, district_id;

Получение статистики по домам

SELECT COUNT(house_id), built_year, floors, apartments, management_company, condition

FROM houses

GROUP BY built_year, floors, apartments, management_company, condition;

Сколько домов на какой улице было построено в каком десятилетии

SELECT

st.name AS street_name,

FLOOR(h.built_year / 10) * 10 AS decade,

COUNT(h.house_id) AS house_count

FROM

houses h

JOIN

streets st ON h.street_id = st.street_id

GROUP BY

st.name,

FLOOR(h.built_year / 10)

ORDER BY

st.name,

decade;

Сравнение моделей

В реляционной модели данные структурированы в таблицах, и при наличии большого количества взаимосвязанных данных требуется больше памяти на хранение. Для хранения одной записи может потребоваться дополнительное место для хранения индексов и для выполнения операций JOIN, которые могут увеличивать объем хранимых данных. В нереляционной модели данные могут храниться в виде документов, и взаимосвязанные данные могут храниться вместе в одном документе, что снижает избыточность. На основании вышесказанного, можно отметить, что при прочих равных условиях реляционная модель может занимать больший объем памяти по сравнению с нереляционной моделью из-за избыточности.

В запросах для ArangoDB и SQL требуется одинаковое количество операций.

Вывод

Для проекта, ориентированного на хранение информации о домах, как в историческом аспекте, так и в контексте удобного доступа и анализа, обе модели (SQL и NoSQL) имеют свои преимущества.

- SQL модель: Подходит для строго структурированных данных с постоянными отношениями и обеспечивает строгую целостность данных. Она лучше для моделей, где требуется точное соблюдение связей, например, для реализации сложных отчетов и аналитики. SQL отлично поддерживает стандартизированные и табличные данные, что делает её оптимальной для моделей с фиксированными схемами и ограниченным количеством вложений и связей.
- NoSQL модель (в частности, ArangoDB): Предлагает гибкость в структуре данных, что

упрощает хранение и добавление новых свойств без необходимости изменять всю схему. ArangoDB, как база графов и документов, позволяет эффективно моделировать сложные отношения и вложения. В проектах с динамической структурой данных и при необходимости горизонтального масштабирования NoSQL будет более подходящим.

С учетом требований к проекту — высокой гибкости в хранении, работы с вложенными данными и возможных изменений в структуре данных — NoSQL может быть предпочтительным выбором, поскольку позволит легче адаптироваться к изменяющимся данным и требованиям.

Разработанное приложение

Краткое описание

Back-end представляет из себя JavaScript-приложение.

Front-end – это web-приложение, которое отображает данные удобным образом для пользователя.

Схема экранов приложения

Экраны приложения и переходы между ними отображены на рисунке 10.

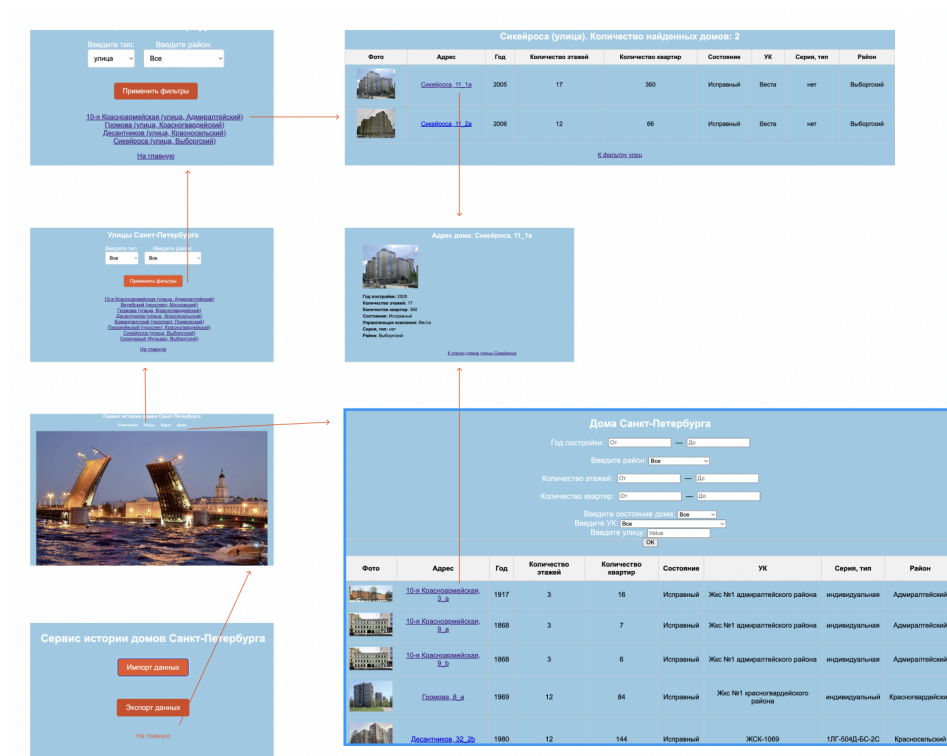


Рисунок 10. Схема экранов приложения.

Использованные технологии

БД: Arango DB

Back-end: JavaScript

Front-end: HTML, CSS, JavaScript.

Вывод

Результаты

В ходе работы было разработано приложение для отображения улиц и домов Санкт-Петербурга по запросам пользователя для СУБД Arango DB.

Будущее развитие решения

В дальнейшем планируется разработать графическое отображение домов на карте и построение графиков (статистику) с осями, которые пользователь может сам задавать.

Приложения

Документация по сборке и разворачиванию приложения

1. Скачать проект из репозитория (указан в литературе)
2. Зайти в папку с проектом
3. Запустить приложение командой `docker-compose up`
4. Открыть приложение в браузере по адресу `127.0.0.1:3000`
5. Просмотреть содержимое бд можно по адресу: `127.0.0.1:8529`

Используемая литература

1. Документация MongoDB: <https://docs.mongodb.com/manual/>
2. Ссылка на репо с проектом: <https://github.com/moevm/nosql2h24-history>