

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

Кафедра математического обеспечения и применения ЭВМ

КУРСОВОЙ ПРОЕКТ

по дисциплине «Введение в нереляционные базы данных»

Тема: Разработка приложения «Каталог произведений искусств»

Студент гр. 9381

Игнашов В.М.

Студент гр. 9381

Матвеев А.Н.

Студент гр. 9381

Птичкин С. А.

Студентка гр. 9381

Андрух И.А.

Преподаватель

Заславский М.М.

Санкт-Петербург

2022

ЗАДАНИЕ

НА КУРСОВОЙ ПРОЕКТ

Студенты: Игнашов В.М., Матвеев А.Н., Птичкин С.А., Андрух И.А.

Группа 9381

Тема работы: Разработка приложения «Каталог произведений искусств»

Исходные данные:

Необходимо разработать веб-приложение, в котором будут храниться карточки произведений искусства с нужной информацией. Реализованное приложение должно использовать Memcached в качестве СУБД.

Содержание пояснительной записи:

“Содержание”, “Введение”, “Качественные требования к решению”, “Сценарии использования”, “Модель данных”, “Разработанное приложение” , “Экспериментальная проверка теоретических оценок для Memcached”, “Заключение”, “Приложения”.

Предполагаемый объем пояснительной записи:

Не менее 15 страниц.

Дата выдачи задания: 01.09.2022

Дата сдачи реферата: 22.12.2022

Дата защиты реферата: 22.12.2022

Студент гр. 9381

Игнашов В.М.

Студент гр. 9381

Матвеев А.Н.

Студент гр. 9381

Птичкин С.А.

Студентка гр. 9381

Андрух И.А.

Преподаватель

Заславский М.М.

АННОТАЦИЯ

В данном проекте было разработано веб-приложение для просмотра и фильтрации карточек произведений искусств. В качестве СУБД должен был использоваться Memcached.

В процессе работы был разработан макет пользовательского интерфейса, анализ основных сценариев использования приложения конечным пользователем, построена модель данных, проведен сравнительный анализ решения на основе Memcached и решения на основе реляционной БД Postgres.

Клиентская часть приложения была написана с использованием библиотеки React, серверная часть разработана на python с использованием фреймворка Flask. Разработанное приложение запускается с помощью docker и docker-compose.

SUMMARY

In this project, a web application was developed for viewing and filtering art cards. Memcached should have been used as the DBMS.

In the process of work, a layout of the user interface was developed, an analysis of the main scenarios for the use of the application by the end user, a data model was built, a comparative analysis of the Memcached-based solution and the Postgres relational database-based solution was carried out.

The client part of the application was written using the React library, the server part was developed in python using the Flask framework. The developed application is launched using docker and docker-compose.

СОДЕРЖАНИЕ

| | | |
|-------|---|----|
| 1. | Введение | 6 |
| 2. | Качественные требования к решению | 7 |
| 3. | Сценарии использования | 8 |
| 3.1. | Макет пользовательского интерфейса | 8 |
| 3.2. | Основные сценарии использования | 8 |
| 4. | Модель данных | 13 |
| 4.1. | Схема данных | 13 |
| 4.2. | Нереляционная модель данных | 13 |
| 4.3. | Графическое представление | 14 |
| 4.4. | Описание назначений коллекций, типов данных и сущностей | 14 |
| 4.5. | Оценка удельного объёма информации | 14 |
| 4.6. | Избыточность нереляционной модели | 15 |
| 4.7. | Направление роста модели. | 15 |
| 4.8. | Запросы к модели | 15 |
| 4.9. | Реляционная модель данных | 16 |
| 4.10 | Графическое представление | 17 |
| 4.11. | Описание назначений коллекций, типов данных и сущностей | 18 |
| 4.12 | Оценка удельного объема информации | 18 |
| 4.13 | Избыточность модели | 18 |
| 4.14. | Направление роста модели | 18 |
| 4.15. | Запросы к модели | 19 |
| 4.16. | Сравнение моделей | 20 |
| 5. | Разработанное приложение | 22 |
| 5.1. | Отображение произведений искусств | 22 |
| 5.2. | Добавление произведений искусств | 22 |
| 5.3. | Редактирование произведений искусств | 25 |
| 5.4. | Поиск произведений искусств по фильтрам | 27 |

| | | |
|------|-----------------------------------|----|
| 5.5. | Анализ | 28 |
| 5.6. | Импорт | 32 |
| 5.7. | Экспорт | 34 |
| | Заключение | 36 |
| | Приложение А. Название приложения | 37 |

1. ВВЕДЕНИЕ

Цель данной работы - разработать приложение, которое будет играть роль справочника произведений искусств. Приложение должно носить информационный характер и предоставлять пользователю удобный интерфейс для получения необходимой информации.

Среди задач веб-приложения можно выделить:

- хранение и отображение информации;
- добавление информации о новых произведениях;
- редактирование уже сохраненной информации;
- фильтрация карточек произведений искусств по различным параметрам;
- импорт данных в приложение;
- экспорт данных из приложения;
- получение диаграммы по результатам поиска карточек произведений

Для решения данных задач было разработано веб-приложение на стеке React+Flask с использованием Memcached в качестве основного инструмента СУБД.

1. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Разработанное решение должно обладать следующими свойствами/решать следующие задачи:

- В качестве НСУБД используется Memcached;
- Существует возможность импорта и экспорта БД;
- В качестве БД используется Postgres;
- В приложении можно осуществлять фильтрацию произведений искусства по численным характеристикам;
- Приложение может быть развернуто средствами контейнеризации.

3. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

3.1. Макет пользовательского интерфейса

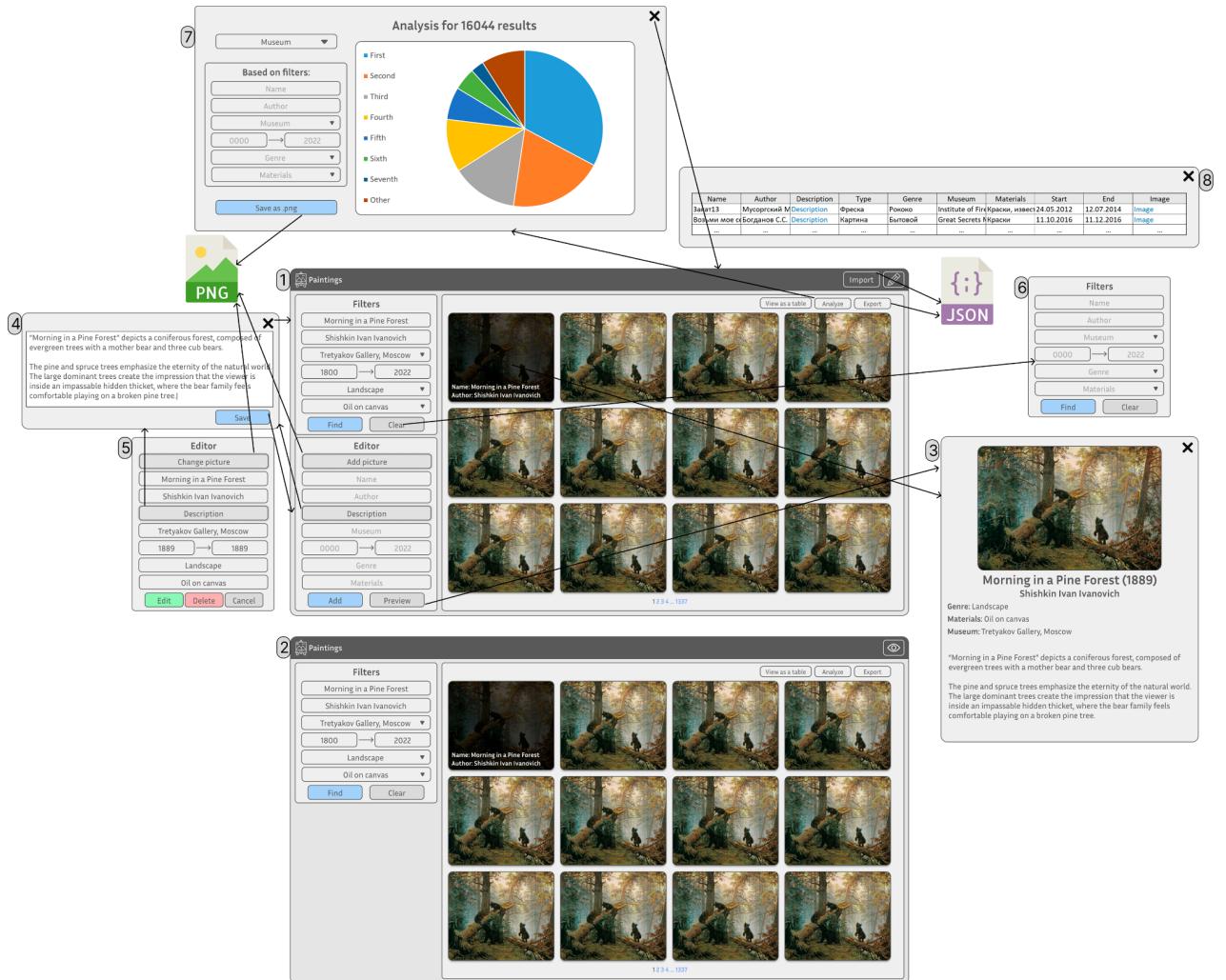


Рис. 1. Макет пользовательского интерфейса

3.2. Основные сценарии использования

Действующее лицо: Пользователь.

Сценарий “Поиск нужной картины”

Основной сценарий.

1. Пользователь открывает приложение.
2. Пользователь заполняет поля фильтра.

Пример заполнения:

Name - Morning in a Pine Forest

Author - Shishkin Ivan Ivanovich

Museum - Tretyakov Gallery, Moscow

1889->1889

Genre - Landscape

Materials - Oil on canvas

3. Пользователь нажимает “Find”.
4. Пользователь получает набор картин, подходящих под заданные параметры фильтра, выбирает нужную, открывает на просмотр. в карточке картины видит саму картину и сведения о ней.

Альтернативный сценарий.

1. Пользователь не заполнил поля фильтра.
2. Пользователь заполнил не все параметры фильтра.
3. Пользователь не смог заполнить поля, так как не понял как заполнять поля с выпадающим списком.
4. Пользователь заполнил поля так, что ни одна карточка не подошла под параметр.
5. Пользователь не нажал “Find”.
6. Пользователь передумал отфильтровывать карточку и нажал “Clear”.

Сценарий “Создание новой картины”.

Основной сценарий.

1. Пользователь открывает приложение.
2. Пользователь нажимает “Add picture”.
3. Пользователь загружает изображение картины.
4. Пользователь заполняет поля “Name”, “Author”.
5. Пользователь нажимает “description”, в результате открывается окно для ввода текста описания.

6. Пользователь вводит описание.
7. Пользователь нажимает “Save”.
8. Пользователь заполняет поля Museum, Genre, Materials при помощи выпадающего списка с вариантами, а также годы создания.
9. Пользователь нажимает “Preview”, открывает предпросмотр карточки.
10. Пользователь нажимает “Add”.

Альтернативный сценарий.

1. Пользователь не заполнил поля, сразу нажал “Add”.
2. Пользователь заполнил не все поля.
3. Пользователь не смог заполнить поля с выпадающим списком.
4. Пользователь закрыл окно с описанием, нажав на крестик. Пользователь передумал регистрировать новую картину, нажал “delete”.
5. Пользователь загрузил некорректную картинку.
6. Пользователь загружает карточку без предпросмотра.

Сценарий “Редактирование картины”.

Основной сценарий.

1. Пользователь открывает приложение.
2. Пользователь ищет картину, используя фильтры.
3. Пользователь выбирает нужную картину и нажимает “Change picture” и загружает нужную картинку.
4. Пользователь редактирует данные.
5. Пользователь нажимает “Edit”, видит выделенную карточку в картотеке с сохраненными изменениями.

Альтернативный сценарий.

1. Пользователь передумал редактировать карточку, выходит из редактора.
2. Пользователь не внес изменения, но нажал “Edit”.

3. Пользователь попытался сохранить карточку с не до конца заполненными полями.
4. Пользователь открыл описание на изменение, изменил его, и закрыл редактор на крестик.

Сценарий “Просмотр и сохранение статистики”

Основной сценарий:

1. Пользователь открывает приложение.
2. Пользователь сортирует картины при помощи фильтра.
3. Пользователь нажимает “Analysis”, открывается окно статистики.
4. Пользователь выбирает параметр карточки, по которому подводится статистика (выпадающий список “Museum”).
5. Пользователь нажимает “Save as png”.
6. Пользователь закрывает окно на крестик.

Альтернативный сценарий:

1. Пользователь закрывает окно, не сохранив статистику.

Сценарий “Импорт данных”.

Основной сценарий:

1. Пользователь открывает приложение.
2. Пользователь находится в режиме редактирования.
3. Пользователь нажимает “Import”.
4. Пользователь выбирает готовый json.
5. После загрузки добавляется карточка с данными из json.

Альтернативный сценарий:

1. Пользователь выбрал другой формат файла.
2. Пользователь загрузил сломанный json, не соответствующий по структуре существующим параметрам карточки.

Сценарий “Экспорт данных”.

Основной сценарий.

1. Пользователь открывает приложение.
2. Пользователь сортирует нужные карточки при помощи фильтра.
3. Пользователь нажимает “Export”.
4. Пользователь находит в загрузках json.

4. МОДЕЛЬ ДАННЫХ

4.1. Схема данных

В программе "Каталог произведений искусства" используется единственная сущность, Произведение искусства. Данная сущность описывается следующими свойствами:

Таблица 1 - Описание свойств произведения искусства.

| Описание | Название свойства | Данные |
|------------------------|-------------------|----------------|
| ID произведения | artworkId | PK |
| Название | name_ | VARCHAR(200) |
| Автор | author_ | VARCHAR(200) |
| Описание | description | VARCHAR(10000) |
| Год начала создания | start_year | INT |
| Год окончания создания | end_year | INT |
| Материалы | materials | VARCHAR(200) |
| Тип | type_name | VARCHAR(100) |
| Название музея | museum_name | VARCHAR(200) |
| Расположение музея | museum_address | VARCHAR(200) |
| Жанр | genre_name | VARCHAR(100) |
| Изображение | URL | VARCHAR(10000) |

4.2. Нереляционная модель данных

В качестве нереляционной БД в приложении используется Memcached. Memcached позволяет хранить отображения вида строка-строка.

4.3. Графическое представление.

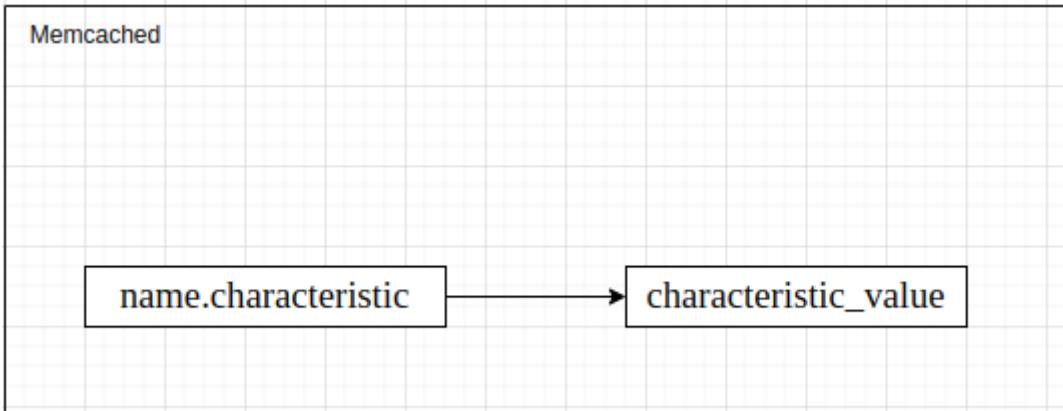


Рис. 2. Структура нереляционной БД

4.4. Описание назначений коллекций, типов данных и сущностей.

Будем хранить 2 типа отображений: all_arts,
{art_name}.{characteristic_name}.

Первое отображение специальное, имеет зарезервированный ключ и позволяет получить строку со списком всех названий произведений.

Отображения вида {art_name}.{characteristic_name} позволяют хранить его поля. Например, Mona_Lisa.museum содержит "Louvre".

4.5. Оценка удельного объёма информации

Пусть N -- количество произведений искусства в базе.

Для каждого поля создаётся свой ключ с помощью уникального id и названия поля. Таким образом верхняя оценка объёма: $N * Artwork_Size + Num_key * N * Key_len$, где Num_key - число ключей, равно 11, а Key_len - длина ключа. Наша верхняя оценка для длины ключа составляет 21(15 - самый длинный ключ, 7 - порядок уникального id, ~10млн сущностей).

4.6. Избыточность нереляционной модели

Чистый объём модели составляет $N * Artwork_Size$, где $Artwork_Size = 21219 \text{ bytes}$.

Фактический объём модели:

$$21219 \text{ bytes} * N + 11 * 21 * N = 21450 * N \text{ bytes}$$

Избыточность модели составляет приблизительно:

$$\frac{21450 * N \text{ bytes}}{21219 * N \text{ bytes}} \approx 1,011$$

4.7. Направление роста модели.

Объем данных растет линейно относительно количества записей.

4.8. Запросы к модели

Создание новой картины:

```
:~$ echo -e 'set 1.genre_name 0 0 7\r\nBarokko\r' | nc localhost 11211

$ echo -e 'set 1.museum_address 0 0 30\r\nLorem ipsum dolor sit amet, co\r' | nc localhost 11211

$ echo -e 'set 1.museum_name 0 0 30\r\nLorem ipsum dolor sit amet, co\r' | nc localhost 11211

echo -e 'set 1.materials 0 0 20\r\nLorem ipsum dolor si\r' | nc localhost 11211

...
andrey@andrey-Lenovo-IdeaPad-Z500:~$ memcdump --servers=localhost
1.genre_name
1.type_name
1.materials
1.end_year
1.start_year
1.author
1.name_
1.URL
1.museum_address
1.museum_name
1.description
```

Рис. 3-8

Редактирование картины

```
1.description  
andrey@andrey-Lenovo-IdeaPad-Z500:~$ echo -e 'get 1.author\r' | nc localhost 11211  
VALUE 1.author 0 11  
Amber Smith  
END  
  
andrey@andrey-Lenovo-IdeaPad-Z500:~$ echo -e 'replace 1.author 0 0 13\r\nPaul Martinez\r' | nc localhost 11211  
STORED  
  
andrey@andrey-Lenovo-IdeaPad-Z500:~$ echo -e 'get 1.author\r' | nc localhost 11211  
VALUE 1.author 0 13  
Paul Martinez  
END
```

Рис. 9-12

Удаление, просмотр полей - аналогично при помощи операции {get/delete} key.

Т.о. можно сказать, что все запросы к Memcached имеют вид {get/add/replace/set/delete} key.

4.9. Реляционная модель данных

Модель состоит из:

- идентификатора
- названия
- 2 характеристик типа DATE
- 8 характеристик типа VARCHAR

В качестве SQL решения будем опираться на Postgres.

4.10. Графическое представление

| ArtWork | |
|---------|---|
| PK1 | <u>arkworkId: SERIAL PRIMARY KEY</u> |
| | name_: VARCHAR(200) author_: VARCHAR(200) description: VARCHAR(10000) start_year: DATE end_year: DATE materials: VARCHAR(200) type_name: VARCHAR(100) museum_name: VARCHAR(200) museum_address: VARCHAR(200) genre_name: VARCHAR(100) URL: VARCHAR(10000) |

Рис. 13. Структура реляционной БД

ArtWork

arkworkId

name

author

description

start_year

end_year

materials

type_name

museum_name

museum_address

genre_name

URL

Рис. 14. ER-диаграмма

4.11. Описание назначений коллекций, типов данных и сущностей

ArtWork - представляет из себя карточку произведения. Содержит его название, автора, краткое описание, год начала, год окончания и материалы, из которых оно сделано, ссылку на изображение, тип произведения, его жанр, музей в котором находится, адрес музея.

4.12. Оценка удельного объема информации

Таблица 2 - Описание размера полей

| Тип | Количество полей в ArtWork | Размер (байт) |
|----------------|----------------------------|---------------|
| SERIAL | 1 | 4 |
| DATE | 2 | 3 |
| VARCHAR(100) | 2 | 101 |
| VARCHAR(200) | 5 | 201 |
| VARCHAR(10000) | 2 | 10001 |

Исходя из выше приведенной таблицы, одна запись содержит

$$4 + 3 * 2 + 101 * 2 + 201 * 5 + 10001 * 2 = 21219 \text{ bytes}$$

А соответственно оценка для всей базы данных: 21219 bytes , где N - количество объектов.

4.13. Избыточность модели

В модели нет избыточности.

4.14. Направление роста модели

Так как в модели одна сущность, то рост модели будет зависеть только от количества этих сущностей.

4.15. Запросы к модели

- Сценарий “Поиск нужной картины”

```
select * FROM Main.artwork
WHERE name_ = 'Возьми мое сердце' AND
      author_ = 'Богданов Степан Семёнович' AND
      museum_name = 'Great Secrets Museum' AND
      museum_address = '109 Community College Rd Ahoskie NC
27910-9522 USA' AND
      materials = 'Краски' AND
      genre_name = 'Пейзаж';
```

- Сценарий "Создание новой картины"

```
DO $$  
DECLARE  
    new_type varchar(100) = 'Глина';  
    new_genre varchar(100) = 'Рококо';  
    new_name varchar(200) = 'Закат13';  
    new_author varchar(200) = 'Мусоргский М.П.';  
    new_description varchar(10000) = '[2022-10-16 23:05:22]  
Connecting to postgres@localhost...';  
    new_museum_name VARCHAR(200) = 'Institute of Fire';  
    new_museum_address VARCHAR(200) = ' 1102 N Massey Blvd';  
    new_materials varchar(200) = 'Краски, известъ';  
    new_start_year DATE = '2012-05-24';  
    new_end_year DATE = '2014-07-12';  
    new_url varchar(10000) =  
'https://www.imgur.com/examples/bee-on-daisy.jpg';  
BEGIN  
    INSERT INTO Main.artwork(type_name, genre_name, name_,  
author_, description, museum_name, museum_address, materials,  
start_year_, end_year_)  
        select new_type, new_genre, new_name, new_author,  
new_description, new_museum_name, new_museum_address,  
new_materials, new_start_year, new_end_year  
        where NOT EXISTS (SELECT type_name, genre_name, name_,  
author_, description, museum_name, museum_address, materials,  
start_year_, end_year_  
                            FROM Main.artwork WHERE type_name =  
new_type AND  
                                genre_name =  
new_genre AND  
                                name_ = new_name  
AND  
                                author_ =
```

```

new_author AND                                     description =
new_description AND                               museum_name =
new_museum_name AND                             museum_address =
new_museum_address AND                         materials =
new_materials AND                                start_year_ =
new_start_year AND                               end_year_ =
new_end_year AND                                 URL = new_url);
END $$;

```

- Сценарий "Редактирование картины"

```

UPDATE Main.artwork
SET name_ = 'No name1'
WHERE Main.artwork.artworkid = 2;

```

- Сценарий "Удаление картины"

```

DELETE   FROM Main.artwork
WHERE Main.artwork.artworkid = 3 ;
SELECT * from Main.artwork order by artworkid;

```

4.16. Сравнение моделей

Оценка для нереляционной модели составляет $21450N$ байтов, где N - количество произведений искусства. Это оценка сверху, так как она получена с рассчетом на максимальный размер хранящихся в свойствах данных. Фактическое наполнение свойств, например, с текстовым типом данных, будет значительно меньше. Memcached позволит хранить эти данные без дополнения до максимальных значений. Но данная модель обладает избыточностью, связанной с необходимостью реализации ключей.

Для реляционной модели оценка составляет $21219N$ байтов, при этом данные хранятся без избыточности. Объем данных для обеих моделей растет линейно, при увеличении количества произведений искусства.

При работе с реляционной моделью для каждого действия достаточно 1 запроса, в то время как в нереляционной модели для выполнения тех же операций сложнее поддерживать согласованность данных, количество запросов слишком большое даже для простейших CRUD операций.

5. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

5.1. Отображение произведений искусства

Отображение реализовано при помощи метода:

- `get_arts()` - делает запрос к БД и выбирает все элементы из БД.
- на клиенте за отображение отвечает компонента `App.js`, в особенности её метод `DataDisplay()`.

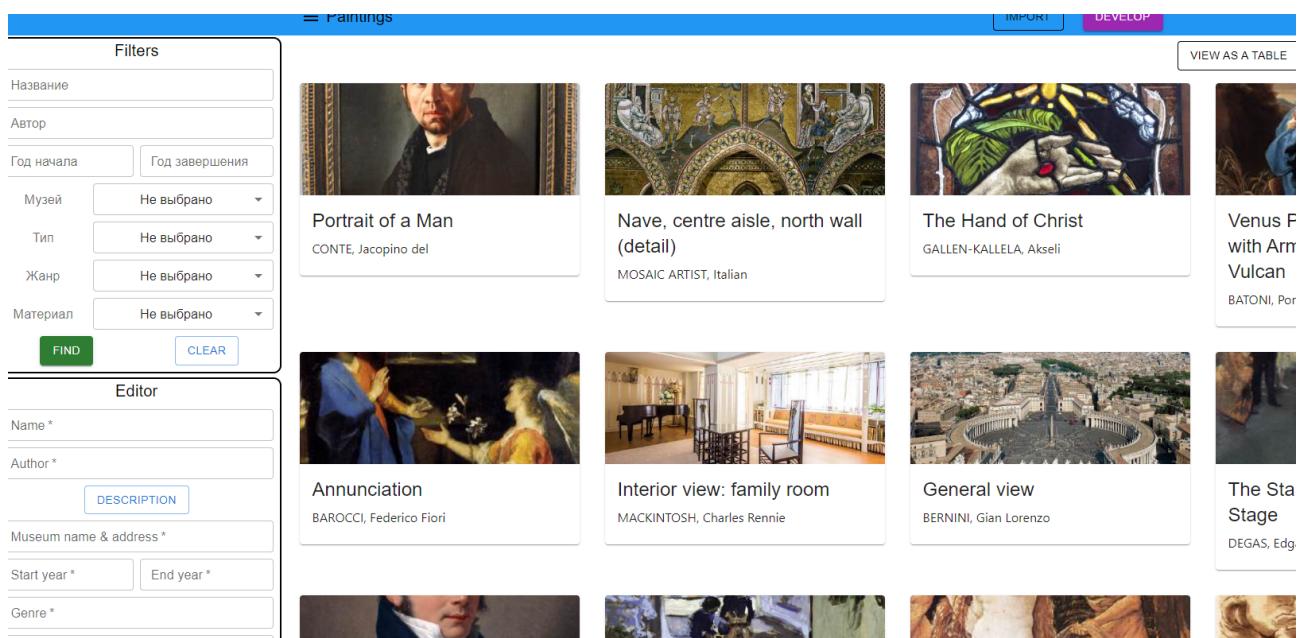


Рис. 15. Отображение.

5.2. Добавление произведений искусства

- на сервере за добавление отвечает метод `add_art()` - он добавляет произведение искусства в БД и memcached.
- На клиенте за добавление отвечает компонента `EditorComp`, в частности его метод `submit()`, который отправляет запрос на сервер с данными нового элемента.

Filters

Название

Автор

Год начала Год завершения

Музей

Тип

Жанр

Материал

Editor

Name *

Author *

Museum name & address *

Start year * End year *

Genre *

Materials *

Type *

JRL *



Portrait of a Man
CONTE, Jacopino del



Nave, centre aisle, r (detail)
MOSAIC ARTIST, Italian



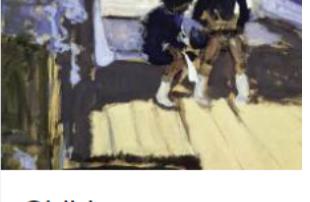
Annunciation
BAROCCI, Federico Fiori



Interior view: family
MACKINTOSH, Charles Rennie



Portrait of a Man
JRL



Children
JRL

Рис. 16. Произведения до добавления

Editor

Name * _____
Nikola

Author * _____
Tesla

DESCRIPTION

Museum name & address * _____
Gallery

Start year * _____ 1900 End year * _____ 1902

Genre * _____ historic

Materials * _____ photo

Type * _____ card

URL * _____ <https://ethnomir.ru/upload/medialibrary/ae2/>

ADD **PREVIEW**

Рис. 17. Добавление

Filters

Название _____

Год начала _____ Год завершения _____

Год завершения: Не выбрано

Год начала: Не выбрано

Год завершения: Не выбрано

Материал: Не выбрано

A black and white photograph of Nikola Tesla, an electrical engineer and physicist, resting his chin on his hand.

Nikola
Tesla

Рис. 18. Отображение изменений после добавления

5.3. Редактирование произведений искусства.

Редактирование существующих карточек реализовано при помощи методов :

- `build_update_query(id: str, update: json)` - серверная функция, принимает на вход `id` и обновленную версию элемента, производит запрос на обновление элемента в базе.
- `update_art(id)` - серверная функция, обертка над `build_update_query`, принимает на вход `id` элемента, который требуется обновить.
- `UpdaterComp` - компонента на клиенте, позволяющая обновить данные уже существующего элемента и отправить изменения на сервер.

Скриншоты работы приложения:

The screenshot displays a user interface for managing artwork records. The top header is 'Paintings' with 'IMPORT' and 'VIEW AS A TABLE' buttons. The left sidebar contains a 'Filters' section with dropdown menus for 'Название', 'Автор', 'Год начала', 'Год завер...', 'Музей' (set to 'Не выбрано'), 'Тип' (set to 'Не выбрано'), 'Жанр' (set to 'Не выбрано'), 'Матери...', and buttons 'FIND' and 'CLEAR'. Below the filters is an 'Editor' section with fields for 'Name*', 'Author*', 'DESCRIPTION' (button), 'Museum name & address*', 'Start year*' and 'End year*', 'Genre*', 'Materials*', 'Type*', 'URL*', and buttons 'ADD' and 'PREVIEW'. The main content area shows two cards: one for 'hurts' by 'кисик' featuring a red tree image, and another for 'супер коты' by 'ира андрух' featuring a cat image. A small number '1' is visible in the bottom right corner.

Рис. 19. Карточка картины до редактирования.

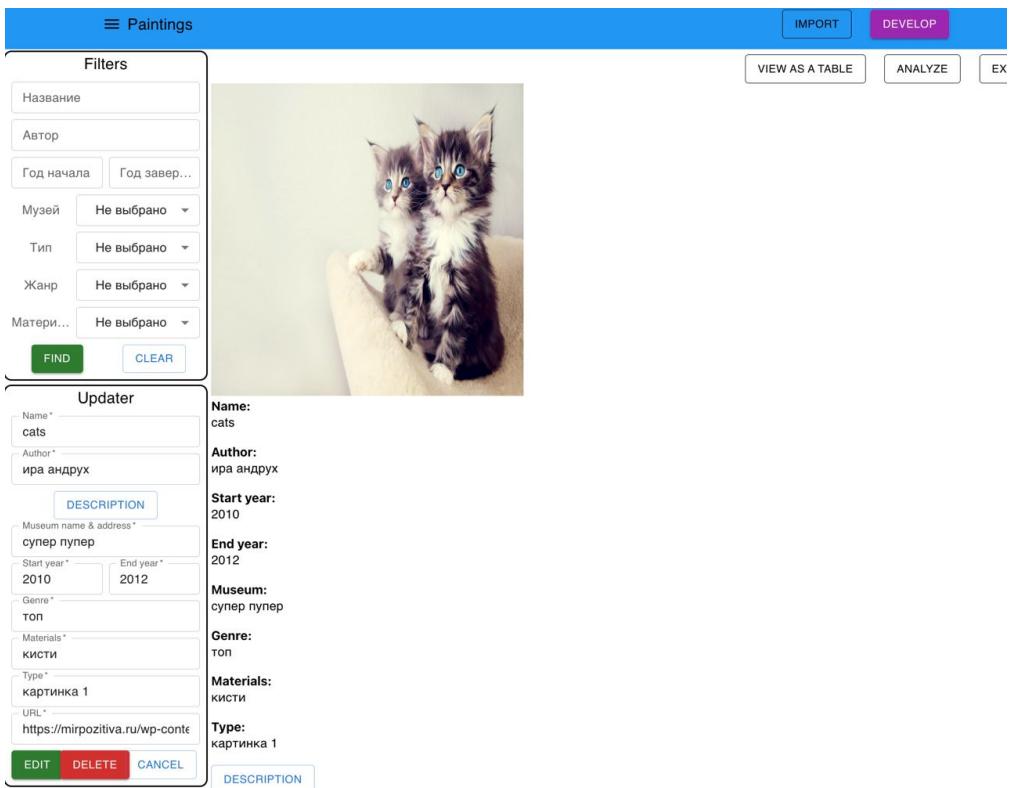


Рис.20. Изменение поля “Name”

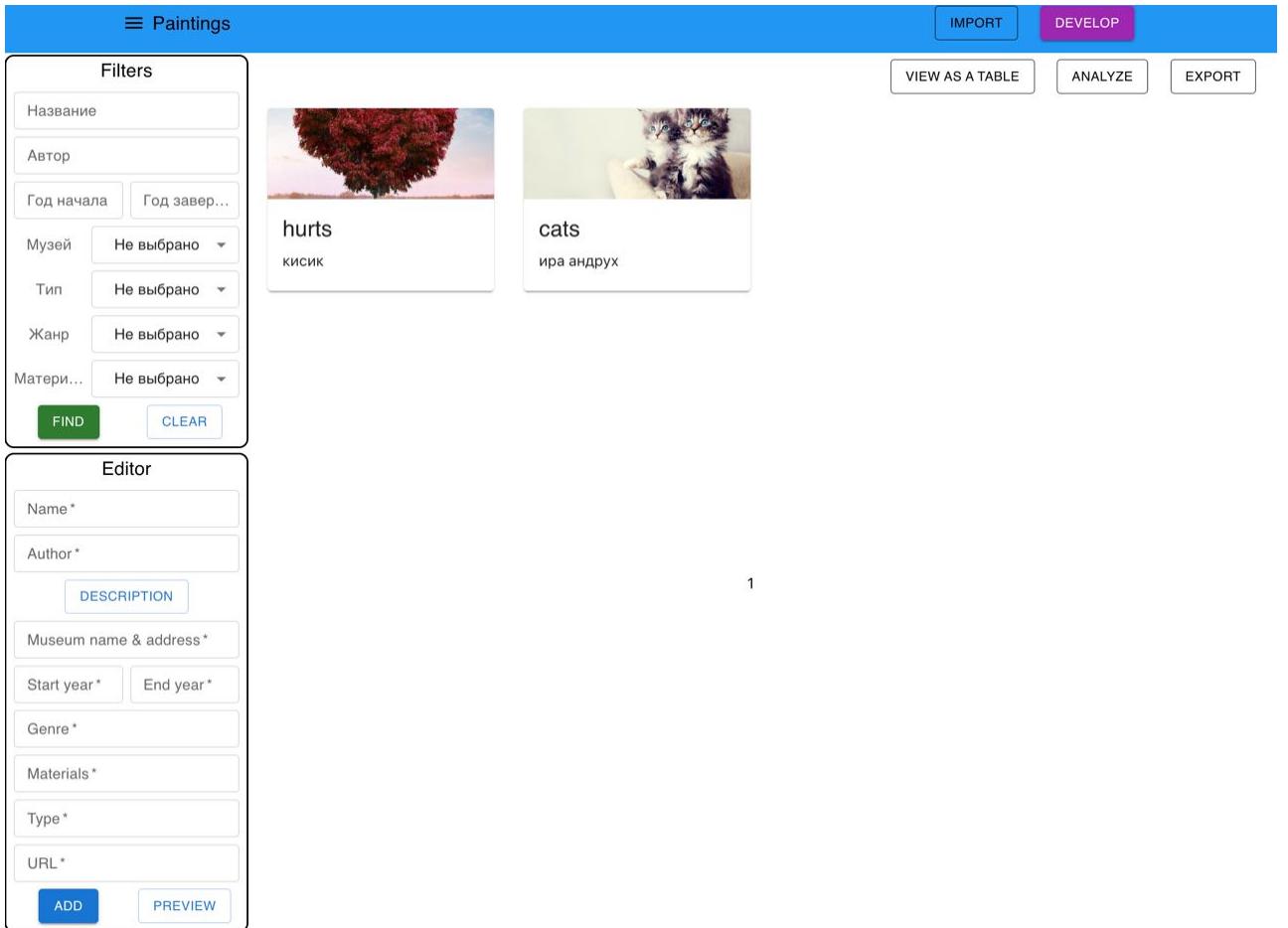


Рис. 21. Отображение сохраненных изменений

5.4. Поиск произведений искусства по фильтрам

- функция `get_arts_by_filter()` - на сервере принимает json с фильтрами, производит запрос на фильтрацию к БД и возвращает результат.
- `FilterComp` - компонента на клиенте, отвечающая за фильтрацию. Пользователь может отфильтровать по названию, автору, году начала, году завершения, музею, типу, жанру, материалу. метод `findByFilter` отправляет запрос, получает ответ и устанавливает соответствующие фильтры.

Filters

Название
Автор
Год начала Год завершения
Музей Не выбрано
Тип mythological
Жанр graphics
Материал Не выбрано

FIND CLEAR

Editor

Name *
Author *
DESCRIPTION
Museum name & address *
Start year * End year *
Genre *
Materials *
Type *
URL *
ADD PREVIEW

VIEW AS A TABLE ANALYZE

Portrait of a Man CONTE, Jacopo del
Nave, centre aisle, north wall (detail) MOSAIC ARTIST, Italian
The Hand of Christ GALLEN-KALLELA, Akseli
Venus Presenting Aeneas with A Forged by Vulcan BATONI, Pompeo

Annunciation BAROCCI, Federico Fiori
Interior view: family room MACKINTOSH, Charles Rennie
General view BERNINI, Gian Lorenzo

Portrait of a Man FABRE, François-Xavier
Children VUILLARD, Édouard
Three Ages of the Woman and the Death BALDUNG GRIEN, Hans

The Star - Dancer on the Stage DEGAS, Edgar
Study for an Equestrian Statue c XIV BERNINI, Gian Lorenzo

1 2 3 4 5 6 ...

Рис. 22. Выбор фильтров

Filters

Название
Автор
Год начала Год завершения
Музей Не выбрано
Тип Не выбрано
Жанр Не выбрано
Материал Не выбрано

FIND CLEAR

Editor

Name *
Author *
DESCRIPTION
Museum name & address *
Start year * End year *
Genre *
Materials *
Type *
URL *
ADD PREVIEW

VIEW AS A TABLE ANALYZE EXPORT

The Three Graces CARRACCI, Agostino
Apollo and the Muses FLORIS, Frans
Jupiter and Mercury in the House of Philemon and Baucis GOUDT, Hendrick
Costume of the allegorical figure "Astrology" ARCIIMBOLDO, Giuseppe

The Rape of Europa DOLENDI, Zacharias
Triumph of Poverty VÖRSTERMAN, Lucas
Apelles Painting Campaspe in the Presence of Alexander HOEFNAGEL, Jacob
Stellion Insulting Ceres HOLLAR, Wenceslaus

1

Рис. 23. Результат фильтрации

5.5. Анализ

- `analysis_by_filter(field: str)` - функция на сервере, принимающая перечень полей, которые необходимо отфильтровать, выполняющая запрос к БД, возвращает значение функции `draw_diagram_get_png()`.
- `draw_diagram_get_png(first_seven_count, other_count, field)` - принимает 2 списка, один из которых - 7 названий наиболее часто встречающихся элементов с их количеством относительно остальных элементов, второй - список из одного элемента с количеством всех остальных, не вошедших в

топ элементов. На основании этих списков строится круговая диаграмма, и на клиент приходит её png-изображение.

- AnalyzeComp - компонента на клиенте, отвечающая за анализ. Отправляет запрос на анализ по отфильтрованным или по всем данным, содержит кнопку, по которой можно скачать результат анализа, а также выводит представление самого изображения анализа.

Filters

Название
s

Автор

Год начала Год заверш...

Музей Staatliche Muse... Тип Не выбрано Жанр Не выбрано Материал Не выбрано

FIND CLEAR

Editor

Name * Author * DESCRIPTION

Museum name & address *

Start year * End year *

Genre *

Materials *

Type *

URL *

ADD PREVIEW

VIEW AS A TABLE ANALYZE EXPORT

| | | | |
|--|---|--|--|
| | Portrait of a Man CONTE, Jacopino del | | Nave, centre aisle, north wall (detail) MOSAIC ARTIST, Italian |
| | The Hand of Christ GALLEN-KALLELA, Akseli | | Venus Presenting Aeneas with Armour Forged by Vulcan BATONI, Pompeo |
| | Annunciation BAROCCI, Federico Fiori | | Interior view: family room MACKINTOSH, Charles Rennie |
| | General view BERNINI, Gian Lorenzo | | The Star - Dancer on the Stage DEGAS, Edgar |
| | Portrait of a Man FABRE, Franois-Xavier | | Children VUILLARD, douard |
| | Three Ages of the Woman and the Death BALDUNG GRIEN, Hans | | Study for an Equestrian Statue of Louis XIV BERNINI, Gian Lorenzo |

1 2 3 4 5 6 ...

Рис. 24. Запрос на фильтрацию

Filters

- Название
- Автор
- Год начала Год заверш...
- Музей Не выбрано
- Тип Не выбрано
- Жанр Не выбрано
- Материал Не выбрано

FIND **CLEAR**

Editor

- Name *
- Author *
- DESCRIPTION**
- Museum name & address *
- Start year * End year *
- Genre *
- Materials *
- Type *

Рис. 25. Результат фильтрации

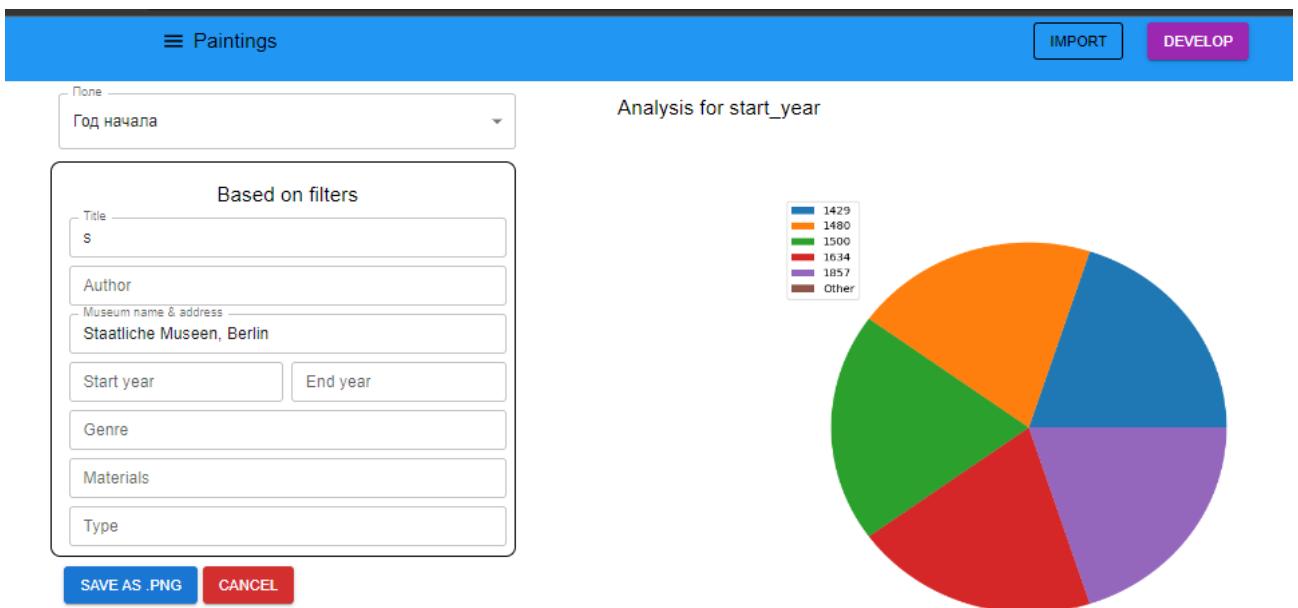


Рис. 26. Результат анализа по отфильтрованным данным

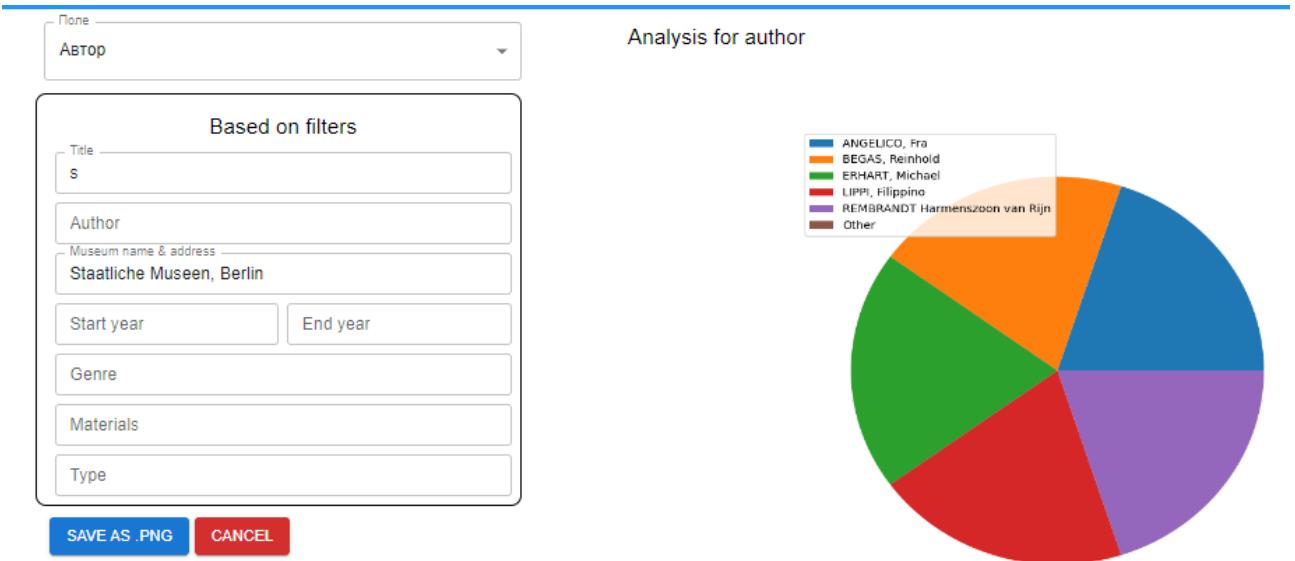


Рис. 27. Результат анализа по отфильтрованным данным

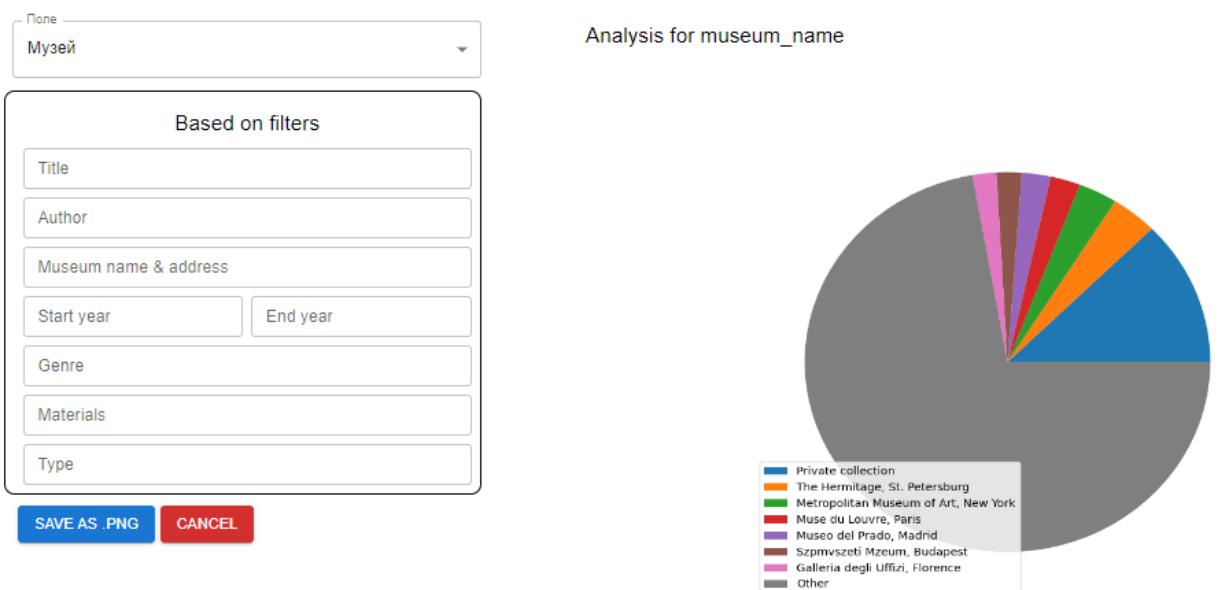


Рис. 28. Результат анализа по всем данным

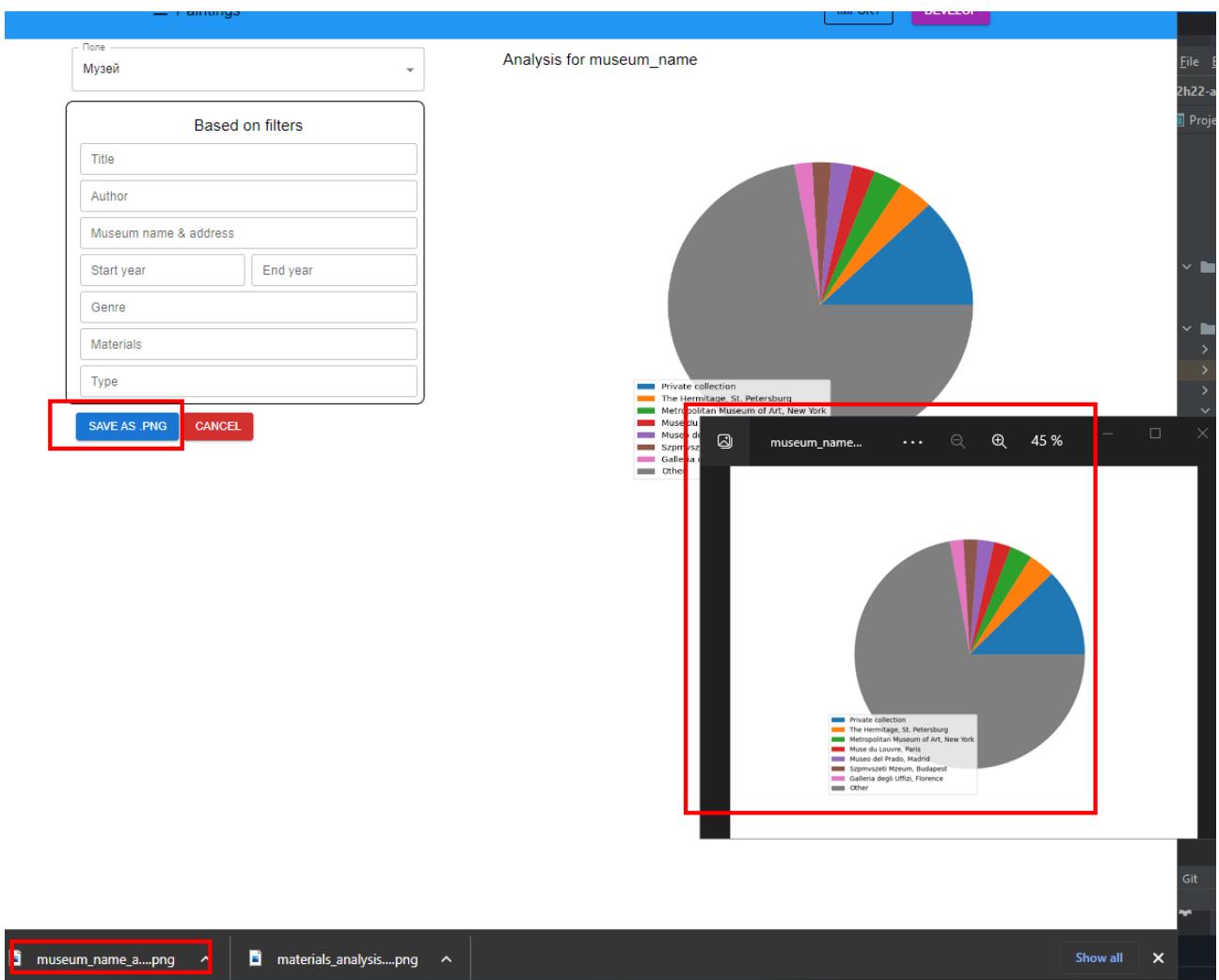


Рис. 29. Сохранение картинки

5.6. Импорт

- `reimport_arts()` (сервер) - импортирует произведения путем очистки базы и memcached и добавления в базу новых из поступившего с клиента json. Также все значения добавляются в memcached.
- AppBarComp - содержит в том числе кнопку import, файл с данными можно загрузить при помощи `onFileChange()`.

Paintings

IMPORT DEVELOP

VIEW AS A TABLE ANALYZE EXPORT

Filters

Название
Автор
Год начала Год завершения
Музей Не выбрано
Тип Не выбрано
Жанр Не выбрано
Материал Не выбрано

FIND CLEAR

Editor

Name *
Author *
DESCRIPTION
Museum name & address *
Start year * End year *
Genre *
Materials *
Type *
URL *
ADD PREVIEW

Рис. 30. Пустая база до импорта

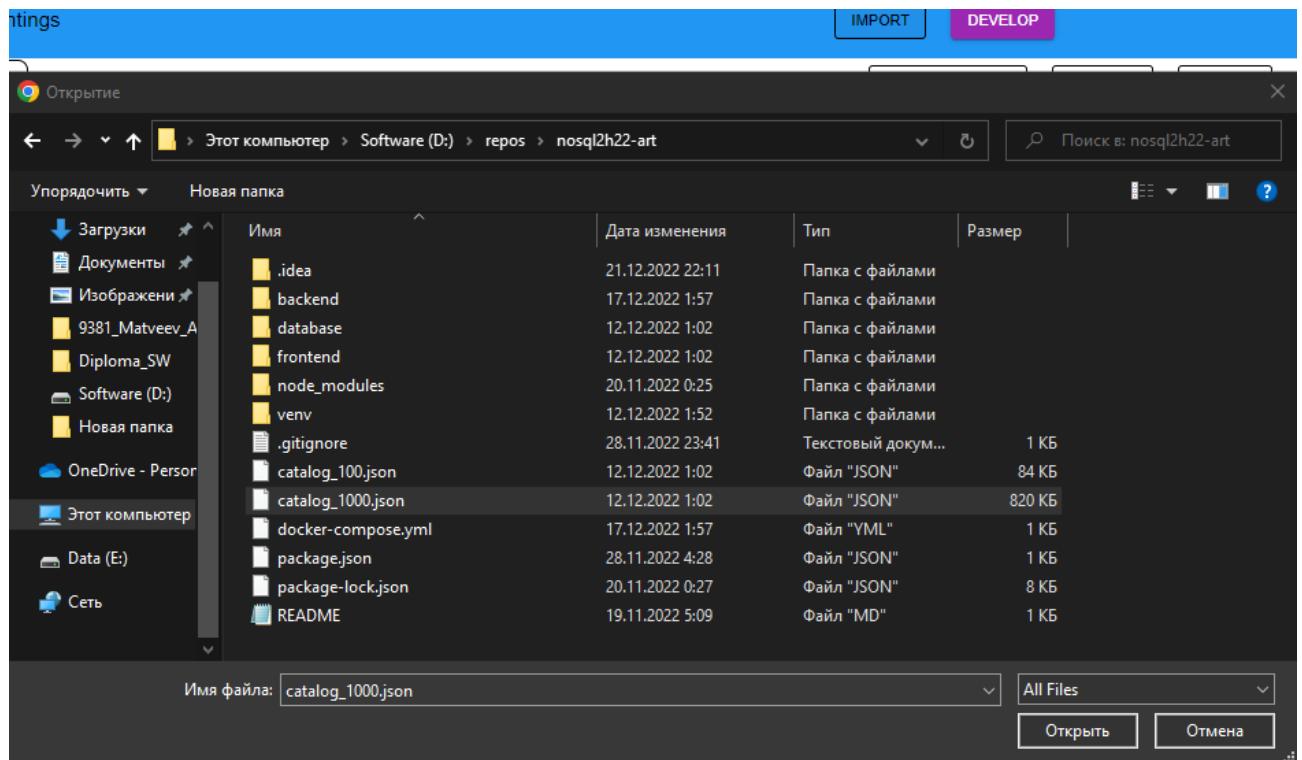


Рис. 31. Импорт

The screenshot shows a digital interface for managing art collections, specifically focusing on paintings. The top navigation bar includes tabs for 'Paintings' (selected), 'IMPORT', 'ANALYZE', and 'EXPORT'. On the left, there's a sidebar titled 'Filters' with fields for 'Название' (Name), 'Автор' (Author), 'Год начала' (Start year) and 'Год заверш...' (End year), 'Музей' (Museum) set to 'Не выбрано' (Not selected), 'Тип' (Type) set to 'Не выбрано' (Not selected), 'Жанр' (Genre) set to 'Не выбрано' (Not selected), and 'Материал' (Material) set to 'Не выбрано' (Not selected). Below these are 'FIND' and 'CLEAR' buttons. Further down is an 'Editor' section with fields for 'Name *', 'Author *', and a 'DESCRIPTION' button. There are also fields for 'Museum name & address *', 'Start year *', 'End year *', 'Genre *', 'Materials *', 'Type *', and 'URL *', each with 'ADD' and 'PREVIEW' buttons.

The main area displays a grid of 12 cards, each representing a different artwork:

- Allegory of the Freedom of Trade** by LAIRESSE, Grard de
- The Throne of Saint Peter (detail)** by BERNINI, Gian Lorenzo
- Exterior view** by JUSSOW, Heinrich Christoph
- St Christopher in a Panoramic Landscape** by COCK, Jan Wellens de
- Lamentation over the Dead Christ (detail)** by SIGNORELLI, Luca
- Franciscan Allegories** by Giotto di Bondone
- San Giovanni a Carbonara: Cappella Caracciolo di Vico** by ARCHITECT, Italian
- Miracle of the Resurrected Child (scene 5)** by SIMONE MARTINI
- Casa Barbey: south faade** by RASPALL, Manuel Joaquim
- Adoration of the Magi** by UNKNOWN MASTER, Spanish
- The Throne of Grace** by MASTER of the Holy Kinship
- No. 11 Scenes from the Life of the Virgin: 5. Marriage of the Virgin (detail)** by Giotto di Bondone

At the bottom center of the grid, there are page navigation links: 1 2 3 4 5 ...

Рис. 32. После импорта

5.7. Экспорт

- `onExport()` в компоненте `App` получает элементы, отправляя запрос по тому же маршруту, что и при отображении.

☰ Paintings

IMPORT
DEVELOP

VIEW AS A TABLE
ANALYZE
EXPORT

Filters

Не выбрано

Не выбрано

Не выбрано

Не выбрано

FIND
CLEAR

Editor

DESCRIPTION

Start year *
End year *

ADD
PREVIEW



Allegory of the Freedom of Trade
LAIRÉSSE, Gérard de



The Throne of Saint Peter (detail)
BERNINI, Gian Lorenzo



Exterior view
JUSOW, Heinrich Christoph



St Christopher in a Panoramic Landscape
COCK, Jan Wellens de



Lamentation over the Dead Christ (detail)
SIGNORELLI, Luca



Franciscan Allegories
GIOTTO di Bondone



San Giovanni a Carbonara: Cappella Caracciolo di Vico
ARCHITECT, Italian



Miracle of the Resurrected Child (scene 5)
SIMONE MARTINI



Casa Barbey: south faade
RASPALL, Manuel Joaquim



Adoration of the Magi
UNKNOWN MASTER, Spanish



The Throne of Grace
MASTER of the Holy Kinship



No. 11 Scenes from the Life of the Virgin: 5. Marriage of the Virgin (detail)
GIOTTO di Bondone

1 2 3 4 5 ...

Show all
X

A screenshot of a file manager interface. At the top, there are three tabs: 'exported_data (9).json', 'museum_name_a...png', and 'materials_analysis....png'. The first tab is highlighted with a red box and has an arrow pointing to it from the previous screenshot. The other two tabs are shown in smaller preview boxes.

Рис. 33. Экспорт

35

ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта было разработано приложение, представляющее каталог произведений искусства.

Приложение позволяет:

- Просматривать данные в виде списка или таблицы.
- Получить всю информацию о произведении.
- Отфильтровать данные по заданным характеристикам.
- Представить анализ отфильтрованных данных в виде диаграммы.
- Редактировать данные.
- Удалять данные.
- Добавлять новые данные.
- Импортировать и экспорттировать данные.

ПРИЛОЖЕНИЕ А

Сборка и развёртывание приложения

- Необходимо установить docker и docker-compose
- Собрать проект с помощью docker-compose build --no-cache
- Запустить проект командой docker-compose up из его корневой директории.