САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ПЕТРА ВЕЛИКОГО

Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Отчет
Курсовая работа
Дисциплина
«Базы данных»

выполнила: Сабо М.А
группа: 3530901/60202
преподаватель: Мяснов А.В

Санкт-Петербург
2019

Оглавление

1. Цель работы

Систематизация и углубление полученных знаний, самостоятельное изучение REST.

2. Программа работы

1. Выбор темы курсовой работы;
2. Написание и согласование технического задания по курсовой работе;
3. Реализация требуемой функциональности;
4. Демонстрация результатов преподавателю;
5. Оформление отчета по курсовой работе.

3. Техническое задание

Разработать Rest API сервис, работающий с базой данных с помощью HTTP-запросов.
В проекте используются:

1. Spring Framework (Spring MVC)
2. JDBC - платформенно независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД
3. Язык программирования Java
4. Postman

Данная реализация будет поддерживать следующие HTTP-запросы: GET, POST, PUT, DELETE.

Добавление и обновление записей происходит в формате JSON.

Удалять и просматривать записи можно с помощью запроса по конкретному id. (пример DELETE запроса: http://localhost:8080/api/author/24)

Пример GET-запроса: http://localhost:8080/api/auction/51


```
 {
     "id": 51,
     "name": "Сотбис",
     "picture": {
         "id": 16,
         "name": "Дора Маар с кошкой",
         "author": {
             "id": 17,
             "name": "Пабло Пикассо"
         }
     },
     "seller": {
         "id": 15,
         "name": "Уиллард и Адель Джидвиц"
     },
     "buyer": {
         "id": 16,
         "name": "Бидзина Иванишвили"
```

```
        },
        "cost": 100,
        "date": "2009-06-10",
        "time_start": "2009-06-10 06:42:44 AM UTC",
        "time_end": "2009-06-10 09:42:44 AM UTC"
    }
```

Также в проекте предусмотрен просмотр записей с использованием параметров. (пример http://localhost:8080/api/author/search?name=Айвазовский).

Рассмотрим модель проведения аукциона с помощью HTTP-запросов:

1. Посылаем POST-запрос. (В файле .json объявляем аукцион, не заполняя поле покупатель(buyer) и окончание аукциона(time_end))
2. Далее посылаем POST-запросы. (Добавляем предложения о покупке картины, каждый «покупатель» делает ставку).
   Для того, чтобы покупатель не делал ставку меньше предыдущей, и чтобы не оказалось, что аукцион закончен аукцион реализован триггер before insert.
3. По прошествии фиксированного времени закрываем аукцион PUT-запросом. Устанавливаем время окончания аукциона(time_end) и покупателя (buyer).
   После обновления с помощью триггера изменяются поля таблицы picturemuseum (устанавливается новый владелец картины).

4. Описание проекта

4.1 Структура базы данных

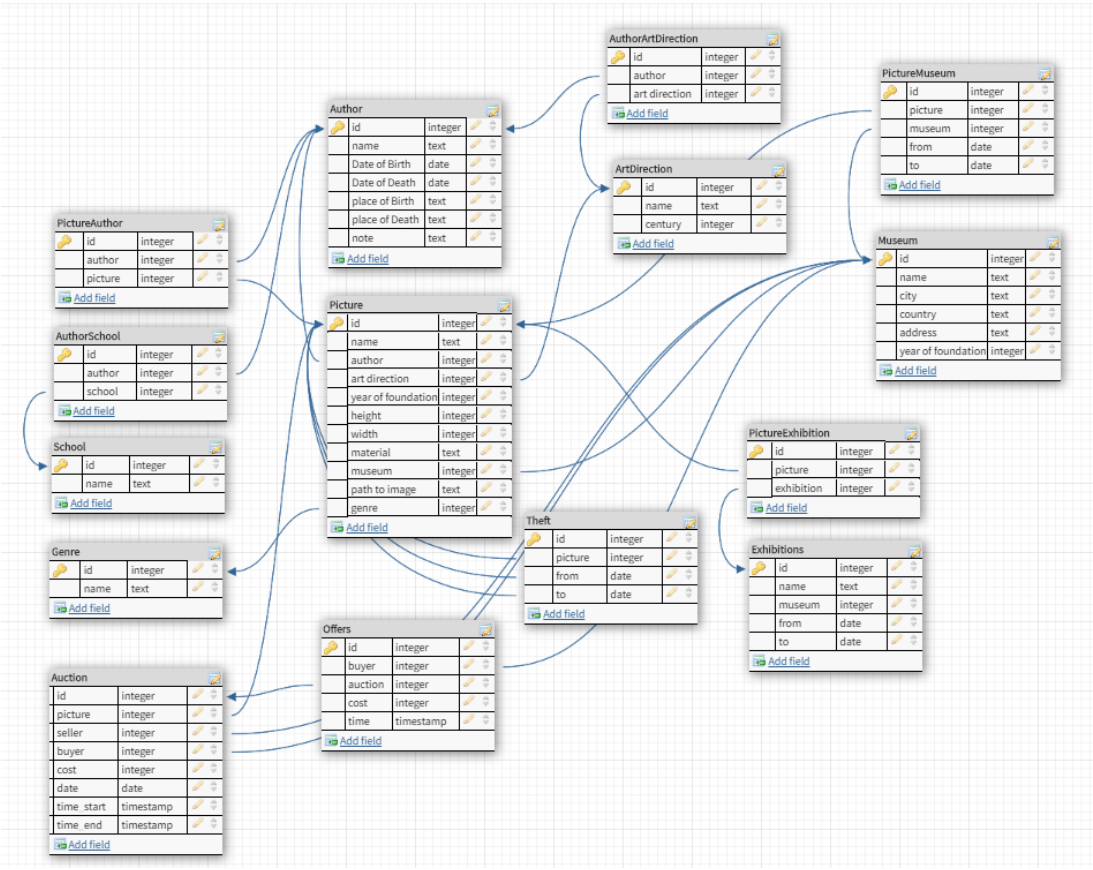

Рис 4.1.1 Структура БД

4.2 Структура приложения

Структура приложения представлена ниже. В model описываются поля базы данных. В rest напишем контроллеры.

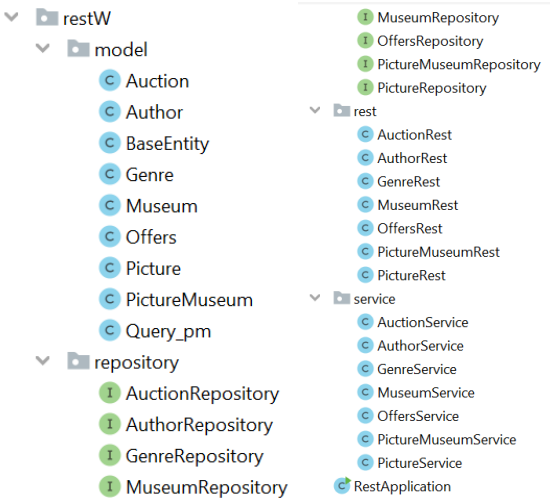

Рис 4.2.1 Структура приложения

5 Тестирование HTTP-запросов

Будем тестировать запросы в приложении Postman.

5.1 GET-запрос

Пример 1: http://localhost:8080/api/author/search?name=айв


```
{
        "id": 2,
        "name": "Айвазовский Иван
Константинович"
}
```


Пример 2: http://localhost:8080/api/author/search?name=ко


```
{
        "id": 2,
        "name": "Айвазовский Иван
Константинович"
    },
    {
        "id": 4,
        "name": "Саврасов Алексей
Кондратьевич"
    },
    {
        "id": 7,
        "name": "Кончаловский Петр
Петрович"
    },
    {
        "id": 9,
        "name": "Машков Илья Иванович"
    },
    {
        "id": 10,
        "name": "Савицкий Константин
Аполлонович"}
```


Пример 3: http://localhost:8080/api/auction/51


```
{
    "id": 51,
    "name": "Сотбис",
    "picture": {
        "id": 16,
        "name": "Дора Маар с кошкой",
        "author": {
            "id": 17,
            "name": "Пабло Пикассо"
        }
    },
```

```
    "seller": {
        "id": 15,
        "name": "Уиллард и Адель Джидвиц"
    },
    "buyer": {
        "id": 16,
        "name": "Бидзина Иванишвили"
    },
    "cost": 100,
    "date": "2009-06-10",
    "time_start": "2009-06-10 06:42:44 AM
UTC",
    "time_end": "2009-06-10 09:42:44 AM
UTC"
}
```

5.2 POST-запрос

http://localhost:8080/api/author

```
    {
        "name": "Сальвадор Дали"
    }
```

В результате в таблицу добавится новый автор.

5.3 PUT-запрос

http://localhost:8080/api/author

```
    {
        "id": 26,
        "name": "Сальвадор Дали2"
    }
```

В результате обновится запись по id = 26.

5.4 DELETE-запрос

В результате http://localhost:8080/api/author/26 будет удален автор по id = 26.

6 Модель проведения аукциона

1. Посылаем POST-запрос. (В файле .json объявляем аукцион, не заполняя поле покупатель(buyer) и окончание аукциона(time_end))

```
{
            "name": "Сотбис",
              "time_start":  "2009-06-10  10:42:44  AM
     MSD",
              "picture": {
                  "id": 16,
                  "name": "Дора Маар с кошкой",
                  "author": {
                      "id": 17,
                      "name": "Пабло Пикассо"
                  }
              },
              "seller": {
                  "id": 15,
                  "name": "Уиллард и Адель Джидвиц"
              },
              "buyer": {
                  "id": 16,
                  "name": "Бидзина Иванишвили"
              },
              "cost": 10,
              "date": "2009-06-10"
}
```


2. Далее посылаем POST-запросы. (Добавляем предложения о покупке картины, каждый «покупатель» делает ставку).


```
     {
              "cost": 100,
              "buyer": {
                  "id": 16,
                  "name": "Бидзина Иванишвили"
              },
              "time": "2009-06-10 10:43:44 AM MSD",
              "auction": {
                  "id": 51,
                  "name": "Сотбис",
                  "picture": {
                      "id": 16,
                      "name": "Дора Маар с кошкой",
                      "author": {
                          "id": 17,
                          "name": "Пабло Пикассо"
```

```
                }
            },
                "seller": {
                    "id": 15,
                    "name":   "Уиллард   и   Адель
    Джидвиц"
                },
                "cost": 10,
                "date": "2009-06-10"
            }
    }
```

Для того, чтобы покупатель не делал ставку меньше предыдущей, и чтобы не оказалось, что аукцион закончен аукцион реализован триггер before insert.
Реализация триггера.

```
    drop function IF EXISTS fun_for_auction CASCADE;
    create   function   fun_for_auction()   returns   trigger   as
    $fun_for_auction$
    begin
    if ((select time_end from auction where auction.id = NEW.auction)
    is not null)
         then RAISE EXCEPTION 'This auction has ended.';
    end if;
    if exists (select *
                 from offers, auction
                 where (offers.auction = auction.id)
                 and  (NEW.cost  <  auction.cost)  and  (auction  =
    NEW.auction))
         then RAISE EXCEPTION 'Your bid is lower than the previous.';

    end if;
    UPDATE auction SET cost = NEW.cost, buyer = NEW.buyer where id =
    NEW.auction;
    return NEW;
    end;
    $fun_for_auction$ language plpgsql;

    create trigger fun_for_auction before insert on offers
for each row execute procedure fun_for_auction()
```

3. По прошествии фиксированного времени закрываем аукцион PUT-запросом. Устанавливаем время окончания аукциона(time_end) и покупателя (buyer).

```
    {
        "id": 51,
        "name": "Сотбис",
        "picture": {
            "id": 16,
            "name": "Дора Маар с кошкой",
            "author": {
                "id": 17,
                "name": "Пабло Пикассо"
            }
        },
        "seller": {
            "id": 15,
            "name": "Уиллард и Адель Джидвиц"
        },
        "buyer": {
            "id": 16,
            "name": "Бидзина Иванишвили"
        },
        "cost": 100,
        "date": "2009-06-10",
        "time_start": "2009-06-10 06:42:44 AM UTC",
        "time_end": "2009-06-10 09:42:44 AM UTC"
}
```

4. После обновления с помощью триггера изменяются поля таблицы picturemuseum (устанавливается новый владелец картины).
   Реализация триггера.

```
drop function IF EXISTS fun_for_close_auction CASCADE;
create function fun_for_close_auction() returns trigger as
$fun_for_close_auction$
begin
insert into picturemuseum (picture, museum, "from") values
(NEW.picture, NEW.buyer, NEW.date);
update picturemuseum set "to" = NEW.date
     where NEW.seller = picturemuseum.museum and NEW.picture =
picturemuseum.picture;
return NEW;
end;
$fun_for_close_auction$ language plpgsql;
```

```
    create trigger fun_for_close_auction after update of time_end on
    auction
for each row execute procedure fun_for_close_auction()
```

7 Выводы

В процессе выполнение курсовой работы были закреплены знания и практические навыки, полученные в течение всего курса "Базы Данных".

Был приобретен опыт работы с СУБД – PostgreSQL и получен опыт разработки REST API сервиса.

8 Список источников

1. Spring Data JPA @Query [Электронный ресурс]: https://www.baeldung.com/spring-data-jpa-query
2. Web on Servlet Stack [Электронный ресурс]: https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html
3. REST [Электронный ресурс]: https://ru.wikipedia.org/wiki/REST

9 Приложения

Код проекта: https://github.com/maria-sabo/Pictures