

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

ОТЧЕТ
по Лабораторной работе № 2

Специальность:
09.03.03 Мобильные и сетевые технологии

Проверил:
Добряков Д. И. _____
Дата: «__» _____ 202__ г.
Оценка _____

Выполнил:
студенты группы К33401
Ковалев В. М.

Санкт-Петербург
2023

ЦЕЛЬ РАБОТЫ

Реализовать свое RESTful API.

ВЫПОЛНЕНИЕ

Мое приложение представляет собой базу данных игр Steam (~8 тысяч записей), по которой можно делать поиск по категориям, жанрам, разработчикам и сортировкой по цене. Сервис доступен только зарегистрированным пользователям. Игры можно добавлять в избранное. Приложение разделено на микросервисы.

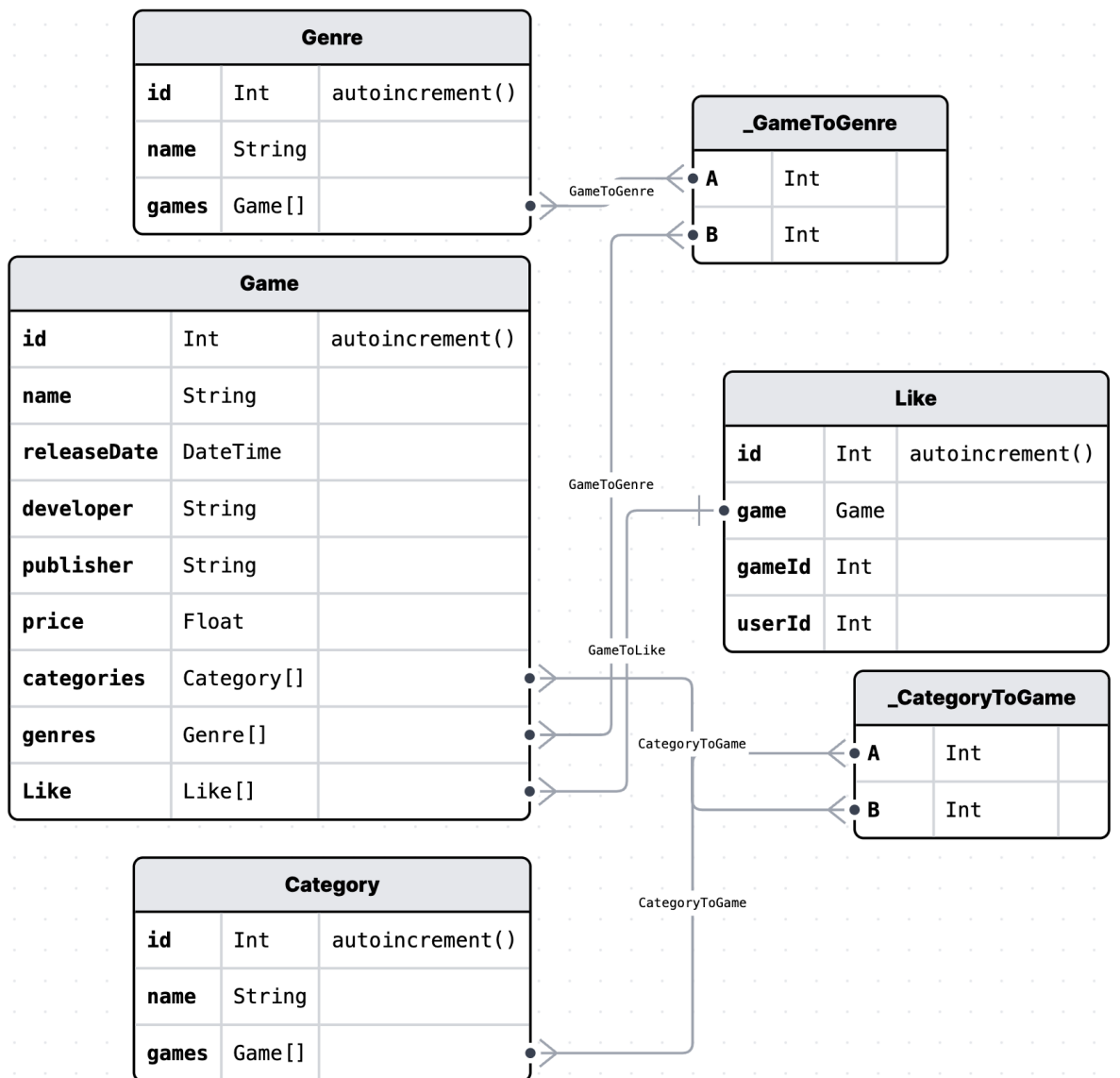
Схема базы данных auth-сервиса

User		
id	Int	autoincrement()
email	String	
password	String	
name	String?	
• JWTAccess	JWTAccess?	

JWTAccess		
id	String	uuid()
createTime	DateTime	now()
user	User	•
userId	Int	
• JWTRefresh	JWTRefresh?	

JWTRefresh		
id	String	uuid()
createTime	DateTime	now()
access	JWTAccess	•
jwtAccessId	String	

Схема базы данных games-сервиса



Файл GameController.ts:

```
class GameController {
  private gamesService: GamesService

  constructor() {
    this.gamesService = new GamesService()
  }

  getAll = async (req: Request, res: Response) => {
    let {offset, count, developer, publisher, sortByPrice} = req.query
    offset = offset ? offset : "0"
    count = count ? count : "10"
    sortByPrice = ["desc", "asc"].includes(String(sortByPrice)) ? String(sortByPrice) : undefined
    developer = developer ? String(developer) : undefined
    publisher = publisher ? String(publisher) : undefined
    const {
      total,
      result
    } = await this.gamesService.getAll(Number(count), Number(offset), developer, publisher, sortByPrice)
    res.json({body: {total, offset, count, result}})
  }
}

export default GameController;
```

Файл LikesController.ts

```
class LikesController {
  private LikesService: LikesService

  constructor() {
    this.LikesService = new LikesService()
  }

  get = async (req: Request, res: Response) => {
    try {
      const { "user-id": userId } = req.headers
      if (userId) {
        return res.json(await this.LikesService.get(Number(userId)))
      }
      return res.status( code: 412 ).json( body: {error: "User ID not in headers"})
    } catch (e: any) {
      return res.status( code: 404 ).json( body: {error: e.message})
    }
  }

  post = async (req: Request, res: Response) => {
    try {
      const { "user-id": userId } = req.headers
      const { gameId } = req.body
      if (userId) {
        return res.json(await this.LikesService.post(Number(gameId), Number(userId)))
      }
      return res.status( code: 412 ).json( body: {error: "User ID not in headers"})
    } catch (e: any) {
      return res.status( code: 404 ).json( body: {error: e.message})
    }
  }

  delete = async (req: Request, res: Response) => {
    try {
      const { "user-id": userId } = req.headers
      const { id: likeId } = req.params
      if (userId) {
        return res.json(await this.LikesService.delete(Number(likeId)))
      }
      return res.status( code: 412 ).json( body: {error: "User ID not in headers"})
    } catch (e: any) {
      return res.status( code: 404 ).json( body: {error: e.message})
    }
  }
}

export default LikesController;
```

ДЕМОНСТРАЦИЯ

Регистрация:

POST

localhost:8000/auth/register

Params

Authorization

Headers (9)

Body ●

Pre-requests

● none

● form-data

● x-www-form-urlencoded

● raw

```
1 {
2   ... "email": "admin@admin.com",
3   ... "password": "admin",
4   ... "name": "Valery"
5 }
```

Body

Cookies

Headers (7)

Test Results

Pretty

Raw

Preview

Visualize

JSON

⌵

⌵

```
1 {
2   "id": 1,
3   "email": "admin@admin.com",
4   "name": "Valery"
5 }
```

Получение токена:

POST

{{url}}/auth/jwt/create

Params

Headers

Body ●

Query Params

	Key	Value
--	-----	-------

Body

Headers (7)

Pretty

Raw

Preview

JSON

⌵

⌵

```
1 {
2   "access": "9648ac95-ba5a-4dbd-85b5-2c93110139af",
3   "refresh": "0ef3c82f-2b3a-4536-a0e0-972b95b93f95"
4 }
```

Получение списка игр (доступна пагинация):

GET

{{url}}/games?offset=20&count=20&publisher=Ubisoft&sortByPrice=desc&developer=Ubisoft

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Headers

7 hidden

	Key	Value
--	-----	-------

Body

Cookies

Headers (7)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

{

2

"total": 55,

3

"offset": "20",

4

"count": "20",

5

"result": [

6

{

7

"id": 408250,

8

"name": "Eagle Flight",

9

"releaseDate": "2016-12-20T00:00:00.000Z",

10

"developer": "Ubisoft Montreal Studio",

11

"publisher": "Ubisoft",

12

"price": 16.99,

13

"categories": [

14

{

15

"id": 1,

Поставить лайк игре:

POST

{{url}}/games/likes/create

Params

Authorization

Headers (9)

Boc

none

form-data

x-www-form-urlencoded

1

{

2

"gameId": 70

3

}

Body

Cookies

Headers (7)

Test Results

Pretty

Raw

Preview

Visualize

1

{

2

"id": 1,

3

"gameId": 70,

4

"userId": 1

5

}

Удалить лайк:

The screenshot shows a REST client interface. At the top, a dropdown menu is set to 'DELETE' and the URL is `{{url}}/games/likes/1`. Below the URL bar, there are tabs for 'Params', 'Authorization', 'Headers (7)', and 'Body'. The 'Headers' tab is selected, showing a button that says '7 hidden'. Below the headers, there are tabs for 'Body', 'Cookies', 'Headers (7)', and 'Test Results'. The 'Body' tab is selected, and within it, there are sub-tabs for 'Pretty', 'Raw', 'Preview', and 'Visualize'. The 'Pretty' sub-tab is active, displaying a JSON response in a formatted, color-coded manner. The JSON response is:

```
{
  "id": 1,
  "gameId": 70,
  "userId": 1
}
```

Полная документация с примерами доступна по [ссылке](#)

ВЫВОД

В результате выполнения лабораторной работы я получил RESTfull API.