

# **FUNDACIÓN UNIVERSITARIA INTERNACIONAL DE LA RIOJA**

**Desarrollo de aplicaciones web (COLINSO) Octubre 2024 PER  
12153**

**Actividad 1: Laboratorio: Desarrollo de un Back-End utilizando  
Java y Spring**

**DANIEL ESTEBAN SANCHEZ ALVAREZ**

**DICIEMBRE 13 DE 2024**

**Introducción**

**Objetivos**

**Desarrollo**

**Conclusiones**

**Referencias**

## Introducción

La presente actividad se realiza en pro de poner en práctica todos conocimientos adquiridos en el proceso de aprendizaje en la lengua de programación Java y el framework Spring. A través de la materia se ha revisado la creación de microservicios y el enrutamiento de los mismos a través de un Gateway, el cual está a subes registrado en un servidor Eureka, el cual alberga los registros para los microservicios cliente.

Esta actividad tiene como objeto principal demostrar la habilidad de crear dichos microservicios, su enrutamiento y funcionalidad a través de consultas en cliente (Postman)

El código para esta actividad se puede encontrar en el siguiente [link](#), en un repositorio público de mi autoría, el cual contiene el servicio de api-gateway, el discovery-service y dos microservicios con CRUD completo y métodos de búsqueda. También, adjunto en este repositorio un archivo tipo 'workspace' de VS Code para facilitar la revisión del proyecto, el cual fue testeado con esta IDE.

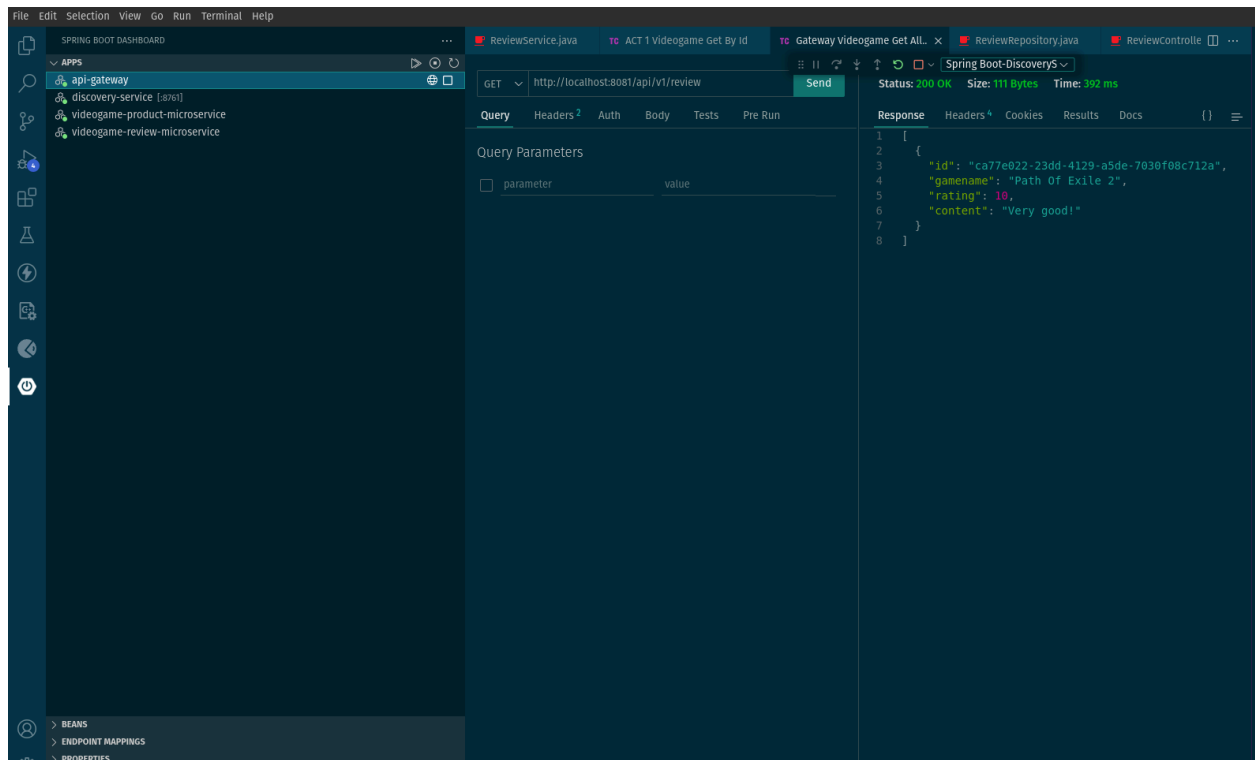
El código también se encuentra adjunto como un comprimido en esta entrega.

## **Objetivos**

- Crear dos microservicios con persistencia en base de datos en el framework Spring.
- Enrutar y conectar estos servicios a través de un servidor Eureka.
- Demostrar persistencia, búsqueda, actualización y borrado de los datos a través de estos microservicios y el servidor de Eureka, a través de un Gateway.

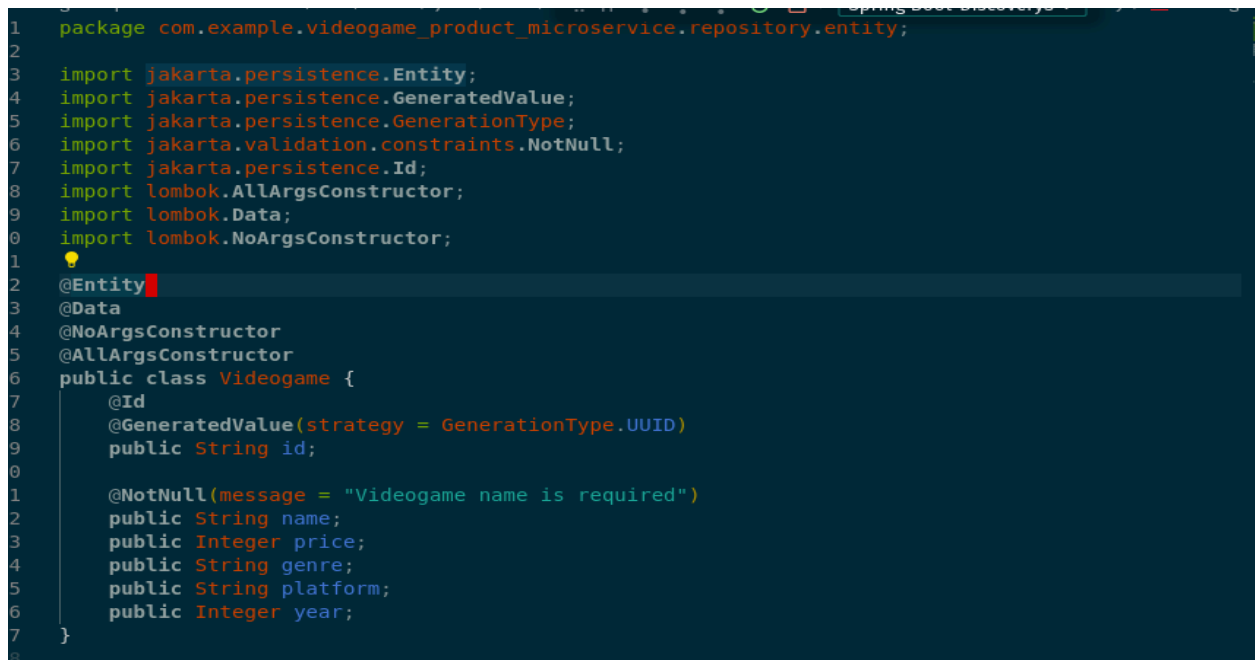
# Desarrollo

## 1. Vista general del proyecto y sus partes



## 2. Microservicio “Videogame Product”

### a. Entidad



## b. Repository

```
1 package com.example.videogame_product_microservice.repository;
2
3 import org.springframework.data.repository.CrudRepository;
4 import org.springframework.stereotype.Repository;
5
6 import com.example.videogame_product_microservice.repository.entity.Videogame;
7
8 @Repository
9 public interface VideogameRepository extends CrudRepository<Videogame, String> {
10     ...
11 }
12
```

## c. Service

```
1 package com.example.videogame_product_microservice.service;
2
3 import java.util.List;
4 import java.util.Optional;
5
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Repository;
8
9 import com.example.videogame_product_microservice.repository.VideogameRepository;
10 import com.example.videogame_product_microservice.repository.entity.Videogame;
11
12 @Repository
13 public class VideogameService {
14     @Autowired
15     VideogameRepository videogameRepository;
16     public List<Videogame> getAll() {
17         return (List<Videogame>) videogameRepository.findAll();
18     }
19
20     public Optional<Videogame> getById(String id) {
21         return videogameRepository.findById(id);
22     }
23
24     public Videogame save(Videogame videogame) {
25         return videogameRepository.save(videogame);
26     }
27
28     public String delete(String id) {
29         videogameRepository.deleteById(id);
30
31         return "Videogame deleted!";
32     }
33 }
34
```

#### d. Controller

```
4 import org.springframework.web.bind.annotation.RestController;
5 import com.example.videogame_product_microservice.repository.entity.Videogame;
6 import com.example.videogame_product_microservice.service.VideogameService;
7
8 import jakarta.validation.Valid;
9
10 @RestController
11 @RequestMapping("api/v1/videogame")
12 public class VideogameController {
13     @Autowired
14     VideogameService videogameService;
15
16     // Get all videogames
17     @GetMapping()
18     public List<Videogame> getAll() {
19         return videogameService.getAll();
20     }
21
22     // Get videogame by id
23     @GetMapping("/{id}")
24     public Optional<Videogame> getAll(@PathVariable(name = "id") String id) {
25         return videogameService.getById(id);
26     }
27
28     // Create videogame
29     @PostMapping
30     public Videogame saveVideogame(@Valid @RequestBody Videogame videogame) {
31         return videogameService.save(videogame);
32     }
33
34     // Update videogame
35     @PutMapping
36     public Videogame updateVideogame(@RequestBody Videogame videogame) {
37         return videogameService.save(videogame);
38     }
39
40     // Delete videogame
41     @DeleteMapping("/{id}")
42     public String deleteVideogame(@PathVariable(name = "id") String id) {
43         return videogameService.delete(id);
44     }
45 }
```

### 3. Microservicio “Videogame Review”

#### a. Entidad

```
1 package com.example.videogame_review_microservice.repository.entity;
2
3 import jakarta.persistence.Entity;
4 import jakarta.persistence.GeneratedValue;
5 import jakarta.persistence.GenerationType;
6 import jakarta.validation.constraints.NotNull;
7 import jakarta.persistence.Id;
8 import lombok.AllArgsConstructor;
9 import lombok.Data;
10 import lombok.NoArgsConstructor;
11
12 @Entity
13 @Data
14 @NoArgsConstructor
15 @AllArgsConstructor
16 public class Review {
17     @Id
18     @GeneratedValue(strategy = GenerationType.UUID)
19     public String id;
20
21     @NotNull(message = "Videogame name is required")
22     public String gamename;
23     ...public Integer rating;
24     ...public String content;
25 }
```

#### b. Repository

```
1 package com.example.videogame_review_microservice.repository;
2
3
4 import org.springframework.data.repository.CrudRepository;
5 import org.springframework.stereotype.Repository;
6
7 import com.example.videogame_review_microservice.repository.entity.Review;
8
9 @Repository
10 public interface ReviewRepository extends CrudRepository<Review, String> {
11
12 }
13
```

#### c. Service

```
13
14 @Repository
15 public class ReviewService {
16
17     @Autowired
18     ReviewRepository reviewRepository;
19     public List<Review> getAll() {
20         return (List<Review>) reviewRepository.findAll();
21     }
22
23     public Optional<Review> getById(String id) {
24         return reviewRepository.findById(id);
25     }
26
27     public Review save(Review review) {
28         return reviewRepository.save(review);
29     }
30
31     public String delete(String id) {
32         reviewRepository.deleteById(id);
33
34         return "Review deleted!";
35     }
36
37 }
```



#### d. Controller

```
17
18 import jakarta.validation.Valid;
19
20 @RestController
21 @RequestMapping("api/v1/review")
22 public class ReviewController {
23     @Autowired
24     ReviewService reviewService;
25
26     // Get all reviews
27     @GetMapping()
28     public List<Review> getAll() {
29         return reviewService.getAll();
30     }
31
32     // Get review by id
33     @GetMapping("{id}")
34     public Optional<Review> getAll(@PathVariable(name = "id") String id) {
35         return reviewService.getById(id);
36     }
37
38     // Create review
39     @PostMapping
40     public Review saveReview(@Valid @RequestBody Review review) {
41         return reviewService.save(review);
42     }
43
44     // Update review
45     @PutMapping
46     public Review updateReview(@RequestBody Review review) {
47         return reviewService.save(review);
48     }
49
50     // Delete review
51     @DeleteMapping("{id}")
52     public String deleteReview(@PathVariable(name = "id") String id) {
53         return reviewService.delete(id);
54     }
55 }
56
```

4. Servicio Eureka funcional y con registros de api-gateway, videogame-product-microservice y videogame-review-microservice:

localhost:8761

spring Eureka

HOME LAST 1000 SINCE STARTUP

System Status

Environmenttest

Data centerdefault

Current time2024-12-13T19:52:50 -0500

Uptime02:46

Lease expiration enabledtrue

Renews threshold5

Renews (last min)12

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - api-gateway:b9837fde-6eb7-4bde-8038-30c315c411e7
VIDEOGAME-PRODUCT-MICROSERVICE	n/a (1)	(1)	UP (1) - videogame-product-microservice:94a05207-e1b7-47d2-a7ac-3606ecb33404
VIDEOGAME-REVIEW-MICROSERVICE	n/a (1)	(1)	UP (1) - videogame-review-microservice:1a374861-b565-4243-ad73-a97b4f5885f7

General Info

Name	Value
total-avail-memory	120mb
num-of-cpus	12
current-memory-usage	52mb (43%)

5. Consultas

a. “Get All Videogames”

ReviewController.java

TC ACT 1 Videogame Get All

Spring Boot-DiscoveryS

GET

http://localhost:8081/api/v1/videogame

Send

Status: 200 OK

Size: 132 Bytes

Time: 28 ms

Query

Headers2

Auth

Body1

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

1 {

2 {

3 "id": "25eaddfb-7414-4cb9-bd37-c1a1a4eb40bf",

4 "name": "Path Of Exile 2",

5 "price": 30,

6 "genre": "ARPG",

7 "platform": "PC, XBOX",

8 "year": 2024

9 }

10 }

Response

Headers4

Cookies

Results

Docs

{}

Copy

## b. “Get Videogame by ID”

The screenshot shows a REST client interface with a GET request to `http://localhost:8081/api/v1/videogame/20607d62-11ac-44cd-814c-f9c26d3459ec`. The response status is 200 OK, size is 124 Bytes, and time is 27 ms. The response body is a JSON object:

```
1 {
2   "id": "20607d62-11ac-44cd-814c-f9c26d3459ec",
3   "name": "Path Of Exile 2",
4   "price": 30,
5   "genre": "ARPG",
6   "platform": "PC",
7   "year": 2024
8 }
```

## c. “Update Videogame”

The screenshot shows a REST client interface with a PUT request to `http://localhost:8081/api/v1/videogame`. The request body is a JSON object:

```
1 {
2   "id": "107ed189-e6b2-4bcb-960c-f01590e0deb7",
3   "name": "STALKER 2",
4   "price": null,
5   "genre": "Action",
6   "platform": "Xbox",
7   "year": 2024
8 }
```

The response status is 200 OK, size is 124 Bytes, and time is 47 ms. The response body is a JSON object:

```
1 {
2   "id": "107ed189-e6b2-4bcb-960c-f01590e0deb7",
3   "name": "STALKER 2",
4   "price": null,
5   "genre": "Action",
6   "platform": "Xbox",
7   "year": 2024
8 }
```

## d. “Create Videogame”

The screenshot shows a REST client interface with a POST request to `http://localhost:8081/api/v1/videogame`. The request body is a JSON object:

```
1 {
2   "name": "Elden Ring",
3   "genre": "Action",
4   "platform": "Xbox, PC",
5   "year": 2022
6 }
```

The response status is 200 OK, size is 129 Bytes, and time is 35 ms. The response body is a JSON object:

```
1 {
2   "id": "880a13a3-c590-4dd3-8c3b-092782ee240b",
3   "name": "Elden Ring",
4   "price": null,
5   "genre": "Action",
6   "platform": "Xbox, PC",
7   "year": 2022
8 }
```

## e. “Delete Videogame”

The screenshot shows a REST client interface with a DELETE request to `http://localhost:8081/api/v1/videogame/63ae9a41-1bda-495f-bd59-3d11ba18c289`. The response status is 200 OK, size is 18 Bytes, and time is 30 ms. The response body is a plain text message:

```
1 Videogame deleted!
```

## f. “Get All Reviews”

GET <http://localhost:8081/api/v1/review> Send

Query Headers<sup>2</sup> Auth Body Tests Pre Run

Query Parameters

☐ parameter value

Status: 200 OK Size: 111 Bytes Time: 423 ms

Response Headers<sup>4</sup> Cookies Results Docs

```
1 [
2 {
3   "id": "ca77e022-23dd-4129-a5de-7030f08c712a",
4   "gamename": "Path Of Exile 2",
5   "rating": 10,
6   "content": "Very good!"
7 }
8 ]
```

## g. “Get Review By ID”

GET <http://localhost:8081/api/v1/review/ca77e022-23dd-4129-a5de-7030f08c712a> Send

Query Headers<sup>2</sup> Auth Body Tests Pre Run

Query Parameters

☐ parameter value

Status: 200 OK Size: 109 Bytes Time: 65 ms

Response Headers<sup>4</sup> Cookies Results Docs

```
1 {
2   "id": "ca77e022-23dd-4129-a5de-7030f08c712a",
3   "gamename": "Path Of Exile 2",
4   "rating": 10,
5   "content": "Very good!"
6 }
```

## h. “Create Review”

POST <http://localhost:8081/api/v1/review> Send

Query Headers<sup>2</sup> Auth Body<sup>1</sup> Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "gamename": "Elden Ring",
3   "rating": 10,
4   "content": "Perfect!"
5 }
```

Status: 200 OK Size: 102 Bytes Time: 109 ms

Response Headers<sup>4</sup> Cookies Results Docs

```
1 {
2   "id": "2b6a50c7-13f3-470d-8698-c5c4a83cce92",
3   "gamename": "Elden Ring",
4   "rating": 10,
5   "content": "Perfect!"
6 }
```

## i. “Update Review”

PUT <http://localhost:8081/api/v1/review> Send

Query Headers<sup>2</sup> Auth Body<sup>1</sup> Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "id": "2b6a50c7-13f3-470d-8698-c5c4a83cce92",
3   "gamename": "Elden Ring",
4   "rating": 10,
5   "content": "Perfect Game!"
6 }
```

Status: 200 OK Size: 107 Bytes Time: 45 ms

Response Headers<sup>4</sup> Cookies Results Docs

```
1 {
2   "id": "2b6a50c7-13f3-470d-8698-c5c4a83cce92",
3   "gamename": "Elden Ring",
4   "rating": 10,
5   "content": "Perfect Game!"
6 }
```

## j. “Delete Review”

DELETE <http://localhost:8081/api/v1/review/2b6a50c7-13f3-470d-8698-c5c4a83cce92> Send

Query Headers<sup>2</sup> Auth Body Tests Pre Run

Query Parameters

☐ parameter value

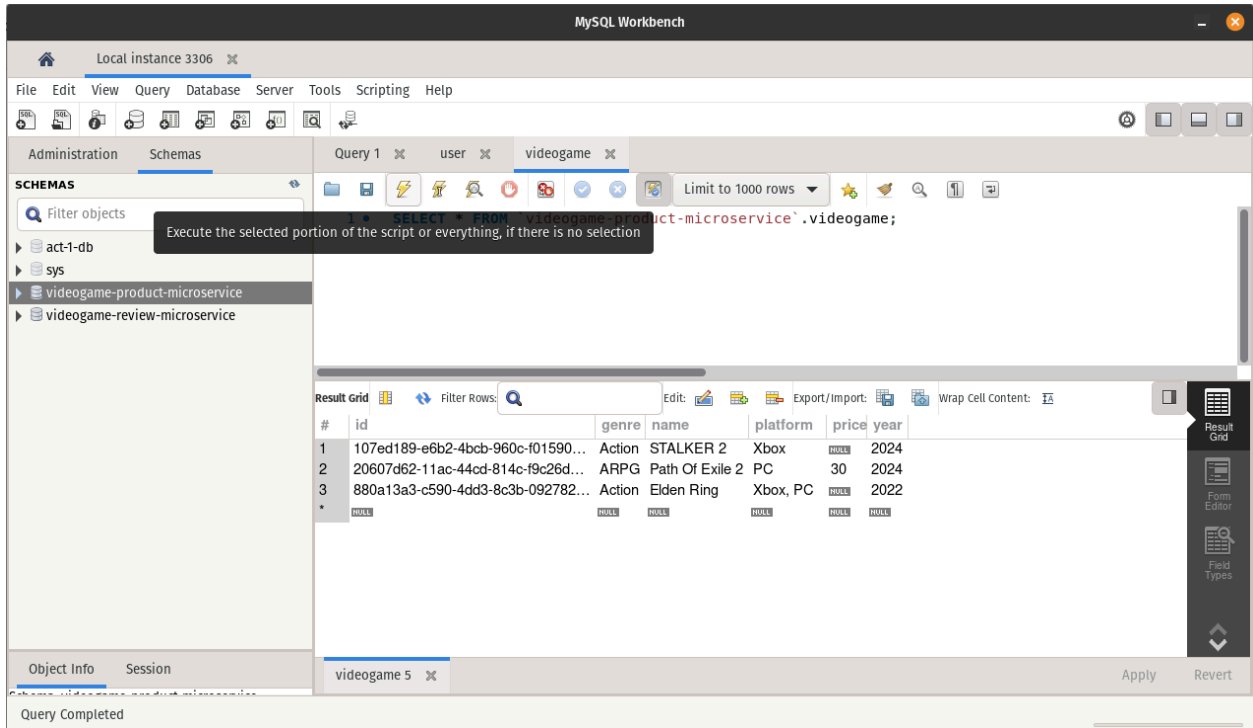
Status: 200 OK Size: 15 Bytes Time: 45 ms

Response Headers<sup>4</sup> Cookies Results Docs

```
1 Review deleted!
```

## 6. MySQL Workbench con bases de datos para cada microservicio

### a. videogame-product-microservice



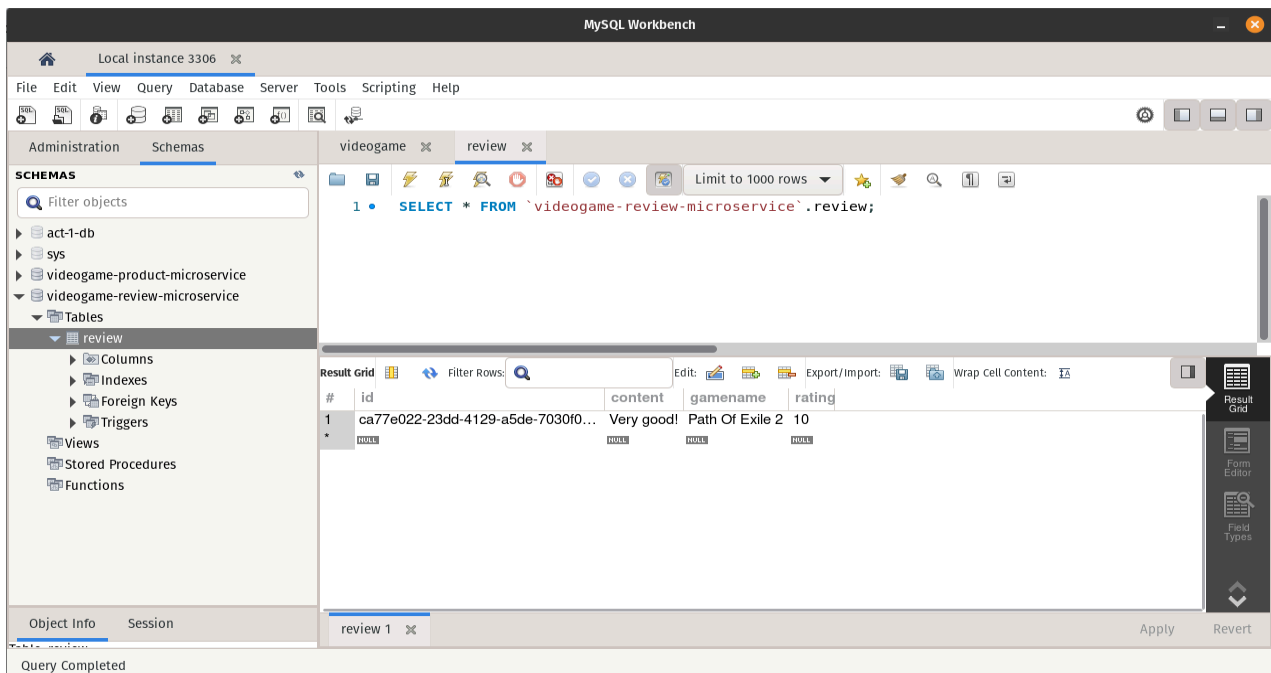
MySQL Workbench interface showing a query executed on the 'videogame-product-microservice' database. The query is:

```
1 • SELECT * FROM `videogame-product-microservice`.videogame;
```

The result grid displays the following data:

#	id	genre	name	platform	price	year
1	107ed189-e6b2-4bcb-960c-f01590...	Action	STALKER 2	Xbox	NULL	2024
2	20607d62-11ac-44cd-814c-f9c26d...	ARPG	Path Of Exile 2	PC	30	2024
3	880a13a3-c590-4dd3-8c3b-092782...	Action	Elden Ring	Xbox, PC	NULL	2022

### b. videogame-review-service



MySQL Workbench interface showing a query executed on the 'videogame-review-microservice' database. The query is:

```
1 • SELECT * FROM `videogame-review-microservice`.review;
```

The result grid displays the following data:

#	id	content	gamename	rating
1	ca77e022-23dd-4129-a5de-7030f0...	Very good!	Path Of Exile 2	10

## Conclusiones

1. Se logra crear los microservicios que generan persistencia en la base de datos.
2. Se logra crear un servicio de Eureka para registrar los microservicios.
3. Se logra crear un api-gateway service para enrutar todos los microservicios a través de un solo lugar.
4. Cabe mencionar que hubo un par de problemas; primero, hubo un par de problemas con el versionamiento de Eureka Client, lo que tomó un buen tiempo para testear y encontrar el problema, ya que la presentación de errores de Spring y Eureka no eran de gran ayuda. Segundo, la velocidad inicial de carga de los microservicios lleva a pensar que no está bien hecho el código, que no están funcionando los servicios, pero solo hay que darle un tiempo considerable a Gateway y Eureka para que inicialice completamente.

## Referencias

- *JDK 21 documentation - home*. Oracle Help Center. (2023, September 19). <https://docs.oracle.com/en/java/javase/21/index.html>
- *Spring Boot*. *Spring Boot :: Spring Boot*. (n.d.). <https://docs.spring.io/spring-boot/index.html>