

## Proyecto 3 — API REST con Spring Boot

- Justificación de la nueva tabla.  
Creo la tabla Entrada para añadir más sentido a la base de datos, ya que es necesario para este proyecto.  
Así realizaba el filtro con los precios.
- Tabla de EndPoints

### ArtistaController

| artista-controller |                        |
|--------------------|------------------------|
| GET                | /artista/{id}          |
| PUT                | /artista/{id}          |
| DELETE             | /artista/{id}          |
| GET                | /artista               |
| POST               | /artista               |
| GET                | /artista/nombre/{name} |

### LugarController

| lugar-controller |                      |
|------------------|----------------------|
| GET              | /lugar               |
| GET              | /lugar/nombre/{name} |
| GET              | /lugar/aforo-desc    |

### EventoController

| evento-controller |                       |
|-------------------|-----------------------|
| GET               | /evento               |
| GET               | /evento/{id}/artistas |
| GET               | /evento/fechas        |

### EntradaController

| entrada-controller |                        |
|--------------------|------------------------|
| GET                | /entrada               |
| GET                | /entrada/filtro-precio |
| GET                | /entrada/buscar-tipo   |

## CategoriaController

**categoria-controller**

**GET** /categorias

**GET** /categorias/nombre/{name}

- Ejemplos de peticiones/respuestas.

**DELETE** /artista/{id}

Parameters

| Name                                | Description                  |
|-------------------------------------|------------------------------|
| <b>id</b> <small>* required</small> | integer(\$int32) 3<br>(path) |

Execute Clear

Responses

Curl

```
curl -X 'DELETE' \
'http://localhost:8081/artista/3' \
-H 'accept: */*'
```

Request URL

```
http://localhost:8081/artista/3
```

Server response

Code Details

204 Undocumented Response headers

```
connection: keep-alive
date: Wed, 14 Jan 2026 22:05:29 GMT
keep-alive: timeout=60
```

|  | <b>id_artista</b> | <b>nombre</b> | <b>genero_musical</b> | <b>pais</b> |  |
|--|-------------------|---------------|-----------------------|-------------|--|
|  | 1                 | C. Tangana    | Urbano/Flamenco       | España      |  |
|  | 2                 | Dua Lipa      | Pop                   | Reino Unido |  |
|  | 4                 | The Grizzly   | Indie Rock            | Canadá      |  |
|  | 5                 | Bizarrap      | EDM/Trap              | Argentina   |  |
|  | NULL              | NULL          | NULL                  | NULL        |  |

**POST /artista**

**Parameters**

No parameters

**Request body required**

```
{
  "id": 0,
  "name": "Papa Roach",
  "generoMusical": "Rock",
  "pais": "Estados Unidos"
}
```

**Responses**

**Curl**

```
curl -X 'POST' \
'http://localhost:8081/artista' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{
  "id": 0,
  "name": "Papa Roach",
  "generoMusical": "Rock",
  "pais": "Estados Unidos"
}'
```

**Request URL**

**http://localhost:8081/artista**

**Server response**

| Code                | Details   |
|---------------------|---|
| 201<br>Undocumented | <b>Response body</b>  |
|                     | <pre>{   "id": 6,   "name": "Papa Roach",   "generoMusical": "Estados Unidos",   "pais": "Rock" }</pre> |

|  | id_artista | nombre      | genero_musical  | pais           |  |
|--|------------|-------------|-----------------|----------------|--|
|  | 1          | C. Tangana  | Urbano/Flamenco | España         |  |
|  | 2          | Dua Lipa    | Pop             | Reino Unido    |  |
|  | 4          | The Grizzly | Indie Rock      | Canadá         |  |
|  | 5          | Bizarrap    | EDM/Trap        | Argentina      |  |
|  | 6          | Papa Roach  | Rock            | Estados Unidos |  |
|  | NULL       | NULL        | NULL            | NULL           |  |

GET /lugar/nombre/{name}

Parameters

| Name   | Description   |
|--|---|
| name <small>* required</small><br>string<br>(path) | Wanda   |
| page<br>integer<br>(query)                         | 0   |
| size<br>integer<br>(query)                         | 20  |
| sort<br>array[string]<br>(query)                   | Sorting criteria in the format: property,(asc desc). Default sort order is ascending. Multiple sort criteria are supported.<br><small>Add string item</small> |

Execute Clear

Responses

Curl

```
curl -X 'GET' \
'http://localhost:8081/lugar/nombre/Wanda?page=0&size=20' \
-H 'accept: */*'
```

Request URL

```
http://localhost:8081/lugar/nombre/Wanda?page=0&size=20
```

Server response

| Code | Details       |
|------|---------------|
| 200  | Response body |

```
{
  "content": [
    {
      "id": 1,
      "name": "Wanda Metropolitano",
      "ciudad": "Madrid",
      "aforo": 68000
    }
  ],
}
```

## Aforo descendente

Parameters

| Name                             | Description  |
|----------------------------------|--|
| page<br>integer<br>(query)       | Zero-based page index (0..N)<br>0  |
| size<br>integer<br>(query)       | The size of the page to be returned<br>20  |
| sort<br>array(string)<br>(query) | Sorting criteria in the format: property,(asc desc). Default sort order is ascending. Multiple sort criteria are supported.<br>Add string item |

Responses

Curl

```
curl -X 'GET' \
'http://localhost:8081/lugar/aforo-desc?page=0&size=20' \
-H 'accept: */*'
```

Request URL

```
http://localhost:8081/lugar/aforo-desc?page=0&size=20
```

Server response

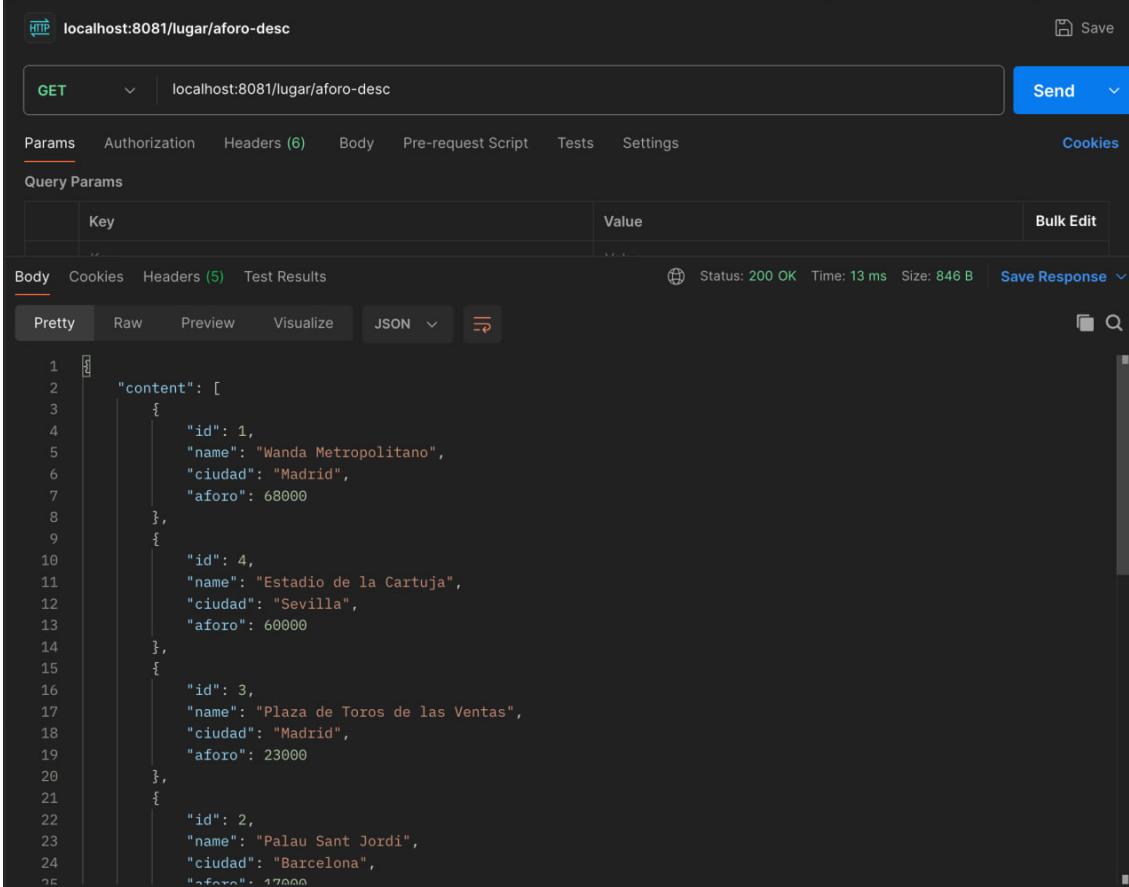
| Code | Details       |
|------|---------------|
| 200  | Response body |

```
{
  "content": [
    {
      "id": 1,
      "name": "Wanda Metropolitano",
      "ciudad": "Madrid",
      "aforo": 68000
    },
    {
      "id": 4,
      "name": "Estadio de la Cartuja",
      "ciudad": "Sevilla",
      "aforo": 60000
    },
    {
      "id": 3,
      "name": "Plaza de Toros de las Ventas",
      "ciudad": "Madrid",
      "aforo": 23000
    },
    {
      "id": 2,
      "name": "Palau Sant Jordi",
      "ciudad": "Barcelona",
      "aforo": 17000
    }
  ]
}
```

Download

Response headers

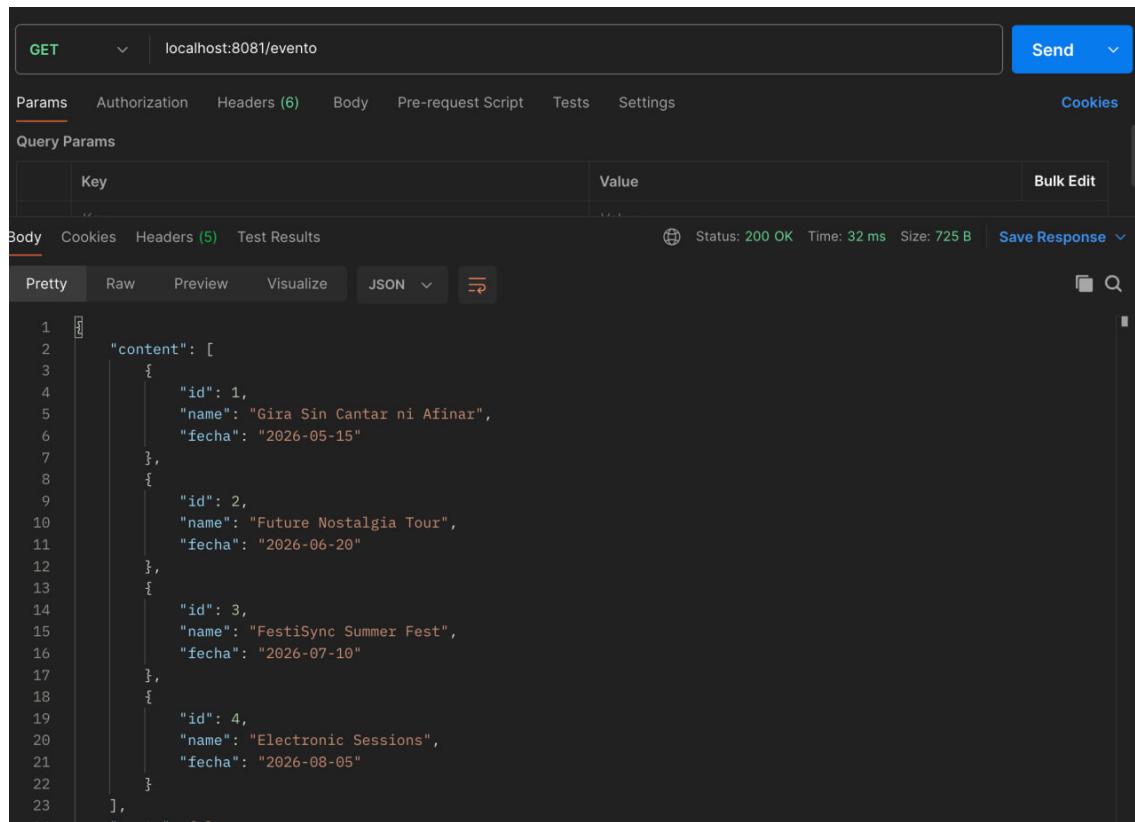
## Comprobación en Postman.



The screenshot shows the Postman application interface. At the top, the URL is set to `localhost:8081/lugar/aforo-desc`. Below the URL, there's a `GET` button and a `Send` button. The `Params` tab is selected, showing a table for `Query Params` with one entry: `Key` and `Value`. The `Body` tab is selected, displaying the JSON response. The response is a pretty-printed JSON array with four elements:

```
1 "content": [
2   {
3     "id": 1,
4     "name": "Wanda Metropolitano",
5     "ciudad": "Madrid",
6     "aforo": 68000
7   },
8   {
9     "id": 4,
10    "name": "Estadio de la Cartuja",
11    "ciudad": "Sevilla",
12    "aforo": 60000
13  },
14  {
15    "id": 3,
16    "name": "Plaza de Toros de las Ventas",
17    "ciudad": "Madrid",
18    "aforo": 23000
19  },
20  {
21    "id": 2,
22    "name": "Palau Sant Jordi",
23    "ciudad": "Barcelona",
24    "aforo": 17000
25  }
]
```

## EVENTO Postman



GET localhost:8081/evento

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

| Key | Value |
|-----|-------|
|-----|-------|

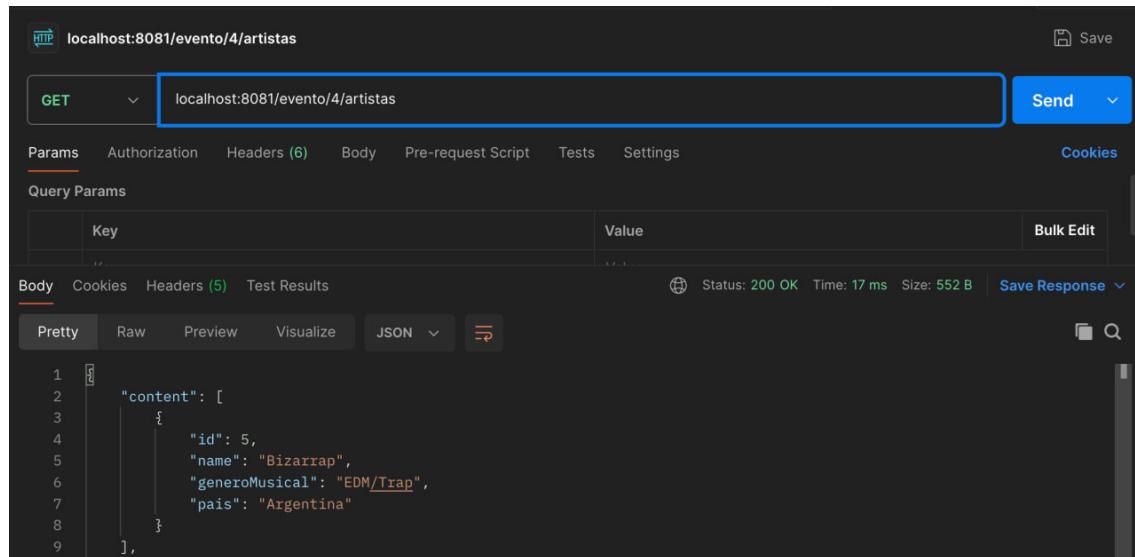
Body Cookies Headers (5) Test Results

Status: 200 OK Time: 32 ms Size: 725 B Save Response

Pretty Raw Preview Visualize JSON

```
1 "content": [
2   {
3     "id": 1,
4     "name": "Gira Sin Cantar ni Afinar",
5     "fecha": "2026-05-15"
6   },
7   {
8     "id": 2,
9     "name": "Future Nostalgia Tour",
10    "fecha": "2026-06-20"
11  },
12  {
13    "id": 3,
14    "name": "FestiSync Summer Fest",
15    "fecha": "2026-07-10"
16  },
17  {
18    "id": 4,
19    "name": "Electronic Sessions",
20    "fecha": "2026-08-05"
21  }
22 ],
23 ]
```

“Artistas que estén en ese IDEvento”



localhost:8081/evento/4/artistas

GET localhost:8081/evento/4/artistas

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

| Key | Value |
|-----|-------|
|-----|-------|

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 17 ms Size: 552 B Save Response

Pretty Raw Preview Visualize JSON

```
1 "content": [
2   {
3     "id": 5,
4     "name": "Bizarrap",
5     "generoMusical": "EDM/Trap",
6     "pais": "Argentina"
7   }
8 ],
9 ]
```

## Filtrado entre fechas

Name Description

|   |   |
|---|---|
| fechaInicio * required<br>string(\$date)<br>(query) | 2026-06-01  |
| fechaFin * required<br>string(\$date)<br>(query)    | 2026-08-10  |
| page<br>integer<br>(query)                          | Zero-based page index (0..N)<br>0   |
| size<br>integer<br>(query)                          | The size of the page to be returned<br>20   |
| sort<br>array[string]<br>(query)                    | Sorting criteria in the format: property,(asc desc). Default sort order is ascending. Multiple sort criteria are supported.<br><input type="button" value="Add string item"/> |

Responses

Curl

```
curl -X 'GET' \
'http://localhost:8081/evento/fechas?fechaInicio=2026-06-01&fechaFin=2026-08-10&page=0&size=20' \
-H 'accept: */*'
```

Request URL

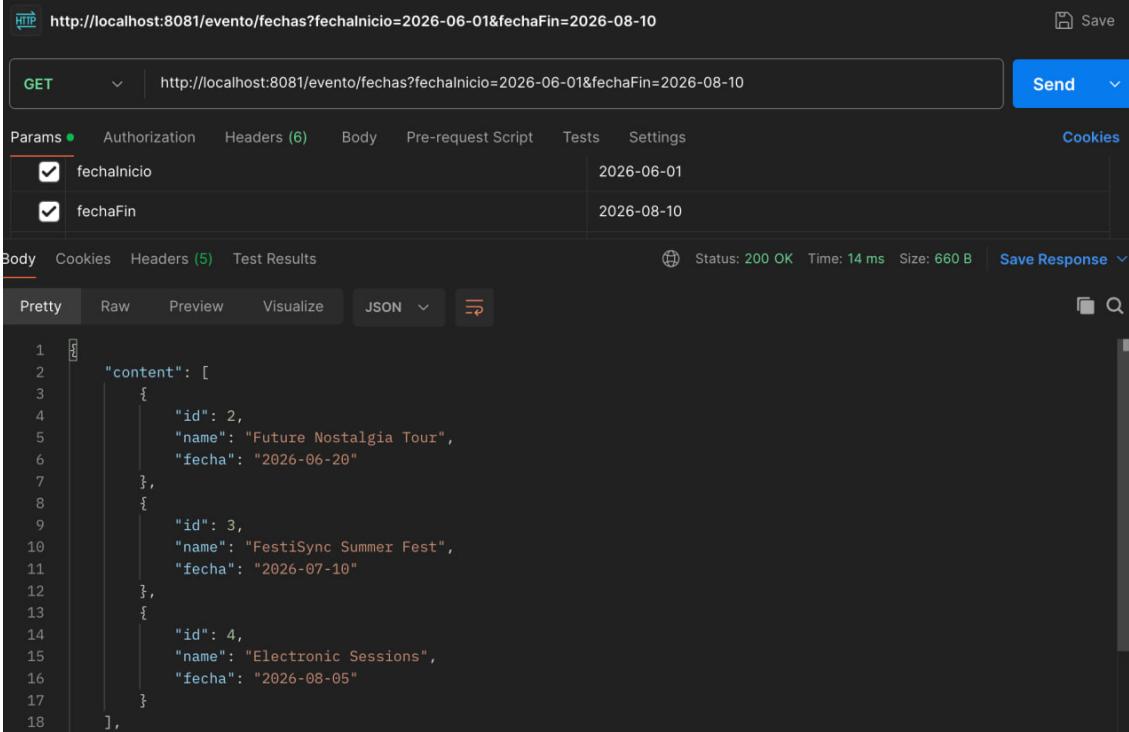
```
http://localhost:8081/evento/fechas?fechaInicio=2026-06-01&fechaFin=2026-08-10&page=0&size=20
```

Server response

| Code | Details       |
|------|---------------|
| 200  | Response body |

```
{
  "content": [
    {
      "id": 2,
      "name": "Future Nostalgia Tour",
      "fecha": "2026-06-20"
    },
    {
      "id": 3,
      "name": "FestiSync Summer Fest",
      "fecha": "2026-07-10"
    },
    {
      "id": 4,
      "name": "Electronic Sessions",
      "fecha": "2026-08-05"
    }
  ],
  "empty": false,
  "first": true,
  "last": true,
  "number": 0,
  "numberOfElements": 3,
  "paged": {
    "size": 0
  }
}
```

## Filtrado en Postman



The screenshot shows a Postman request configuration for a GET request to `http://localhost:8081/evento/fechas?fechalinicio=2026-06-01&fechaFin=2026-08-10`. The 'Params' tab is selected, showing two parameters: `fechalinicio` with value `2026-06-01` and `fechaFin` with value `2026-08-10`. The 'Body' tab shows a JSON response with three events:

```
1 "content": [
2   {
3     "id": 2,
4     "name": "Future Nostalgia Tour",
5     "fecha": "2026-06-20"
6   },
7   {
8     "id": 3,
9     "name": "FestiSync Summer Fest",
10    "fecha": "2026-07-10"
11  },
12  {
13    "id": 4,
14    "name": "Electronic Sessions",
15    "fecha": "2026-08-05"
16  }
17 ],
18 ]
```

## GET Entradas

GET /entrada

Parameters

| Name   | Description  |
|--|--|
| page<br><small>integer (query)</small>       | Zero-based page index (0..N)<br>0  |
| size<br><small>integer (query)</small>       | The size of the page to be returned<br>20  |
| sort<br><small>array[string] (query)</small> | Sorting criteria in the format: property,(asc desc). Default sort order is ascending. Multiple sort criteria are supported.<br>Add string item |

Execute Clear

Responses

Curl

```
curl -X 'GET' \
'http://localhost:8081/entrada?page=0&size=20' \
-H 'accept: */*' 
```

Request URL

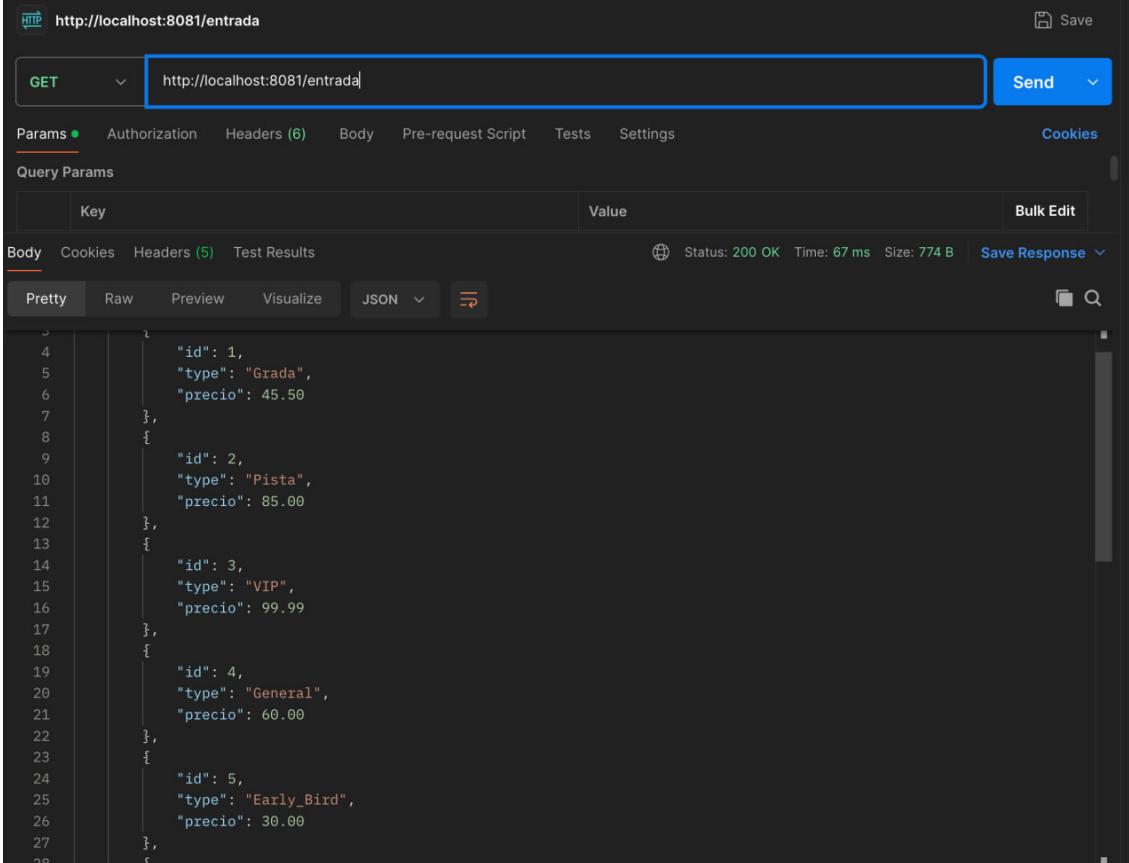
```
http://localhost:8081/entrada?page=0&size=20
```

Server response

| Code | Details       |
|------|---------------|
| 200  | Response body |

```
{
  "content": [
    {
      "id": 1,
      "type": "Grada",
      "precio": 45.5
    },
    {
      "id": 2,
      "type": "Pista",
      "precio": 85
    },
    {
      "id": 3,
      "type": "VIP",
      "precio": 99.99
    },
    {
      "id": 4,
      "type": "General",
      "precio": 60
    },
    {
      "id": 5,
      "type": "Early Bird"
    }
  ]
}  
```

## GET Entradas en Postman



The screenshot shows a Postman interface with a GET request to `http://localhost:8081/entrada`. The response body is a JSON array of five objects, each representing an entrance with fields `id`, `type`, and `precio`.

```
[{"id": 1, "type": "Grada", "precio": 45.5}, {"id": 2, "type": "Pista", "precio": 85.0}, {"id": 3, "type": "VIP", "precio": 99.99}, {"id": 4, "type": "General", "precio": 60.0}, {"id": 5, "type": "Early_Bird", "precio": 30.0}]
```

## GET Entrada/buscartipo

GET /entrada/buscar-tipo

Parameters

Name Description

tipo **required** General  
string  
(query)

page Zero-based page index (0..N)  
integer  
(query)  
0

size The size of the page to be returned  
integer  
(query)  
20

sort Sorting criteria in the format: property,(asc|desc). Default sort order is ascending. Multiple sort criteria are supported.  
array[string]  
(query)  
Add string item

Execute Clear

Responses

Curl

```
curl -X 'GET' \
'http://localhost:8081/entrada/buscar-tipo?tipo=General&page=0&size=20' \
-H 'accept: */*' 
```

Request URL

```
http://localhost:8081/entrada/buscar-tipo?tipo=General&page=0&size=20
```

Server response

| Code | Details  |
|------|--|
| 200  | Response body  |
|      | { "content": [ { "id": 4, "type": "General", "precio": 60 } ]} |

## GET entrada/filtroprecio

GET /entrada/filtro-precio

Parameters

Name Description

precioMax \* required  
number  
(query)  
50.00

page  
integer  
(query)  
0

size  
integer  
(query)  
20

sort  
array[string]  
(query)  
Sorting criteria in the format: property,(asc|desc). Default sort order is ascending. Multiple sort criteria are supported.  
Add string item

Execute Clear

Responses

Curl

```
curl -X 'GET' \
'http://localhost:8081/entrada/filtro-precio?precioMax=50.00&page=0&size=20' \
-H 'accept: */*' 
```

Request URL

```
http://localhost:8081/entrada/filtro-precio?precioMax=50.00&page=0&size=20
```

Server response

Code Details

200 Response body

```
content : [ { "id": 1, "type": "Grada", "precio": 45.5 }, { "id": 5, "type": "Early_Bird", "precio": 30 }, { "id": 7, "type": "Entrada_Unica", "precio": 40 } ], "empty": false
```

## GET categorias/nombre

The screenshot shows the Postman interface for a GET request to `/categorias/nombre/{name}`. The parameters section includes:

- name** (required, string, path): Concierto
- page** (integer, query): 0
- size** (integer, query): 20
- sort** (array[string], query): Add string item

Below the parameters are buttons for **Execute** and **Clear**. The **Responses** section contains a **Curl** command and a **Request URL** field. The **Server response** section shows a **Code** of 200 and a **Details** section with a JSON response body:

```
{ "content": [ { "id": 1, "name": "Concierto" } ] }
```

En Postman.

The screenshot shows the Postman UI with the following details:

- HTTP Method:** GET
- URL:** `http://localhost:8081/categorias/nombre/Concierto`
- Params:** size (Value: 20)
- Headers:** (6 items)
- Body:** (Pretty, Raw, Preview, Visualize, JSON)
- Status:** 200 OK
- Response Body:** (JSON)

```
1 [ { "content": [ { "id": 1, "name": "Concierto" } ] } ]
```

## GET Categorias

**categoria-controller**

**GET /categorias**

**Parameters**

| Name                             | Description  |
|----------------------------------|--|
| page<br>integer<br>(query)       | Zero-based page index (0..N)<br>0  |
| size<br>integer<br>(query)       | The size of the page to be returned<br>20  |
| sort<br>array(string)<br>(query) | Sorting criteria in the format: property,(asc desc). Default sort order is ascending. Multiple sort criteria are supported.<br>Add string item |

**Responses**

**Curl**

```
curl -X 'GET' \
'http://localhost:8081/categorias?page=0&size=20' \
-H 'accept: */*'
```

**Request URL**

```
http://localhost:8081/categorias?page=0&size=20
```

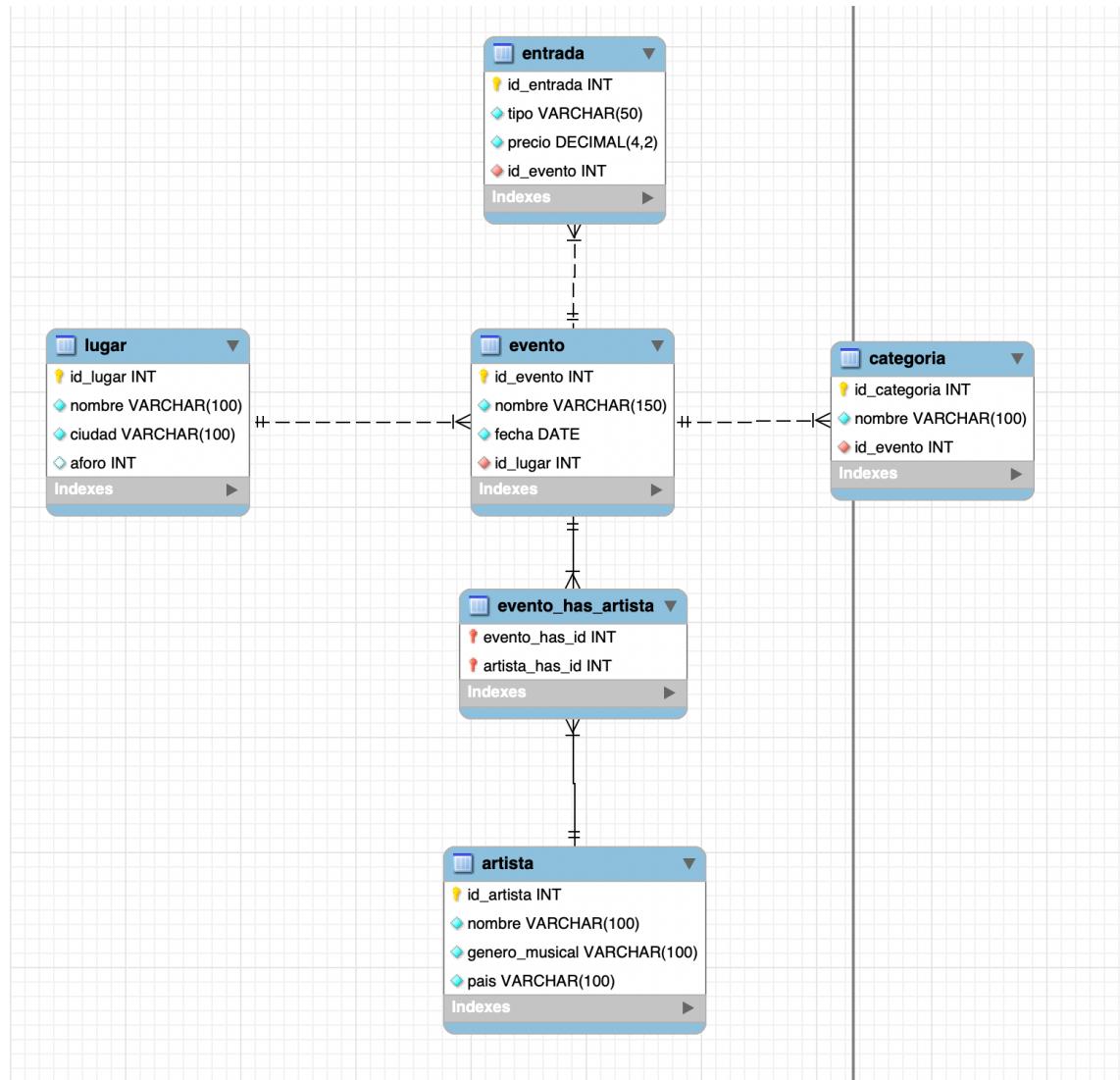
**Server response**

**Code** **Details**

200 Response body

```
{
  "content": [
    {
      "id": 1,
      "name": "Concierto"
    },
    {
      "id": 2,
      "name": "Gira Mundial"
    },
    {
      "id": 3,
      "name": "Festival"
    },
    {
      "id": 4,
      "name": "Clubbing"
    }
  ],
  "pageable": {
    "page": 0,
    "size": 20,
    "totalPages": 1,
    "totalElements": 4
  },
  "last": true,
  "number": 0
}
```

- Diagrama E/R final.



```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS,
UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN
_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SU
BSTITUTION';

-----
-- Schema FestiSync
-----

-----
-- Schema FestiSync
-----

CREATE SCHEMA IF NOT EXISTS FestiSync ;
USE FestiSync ;

-----
-- Table FestiSync.lugar
-----

DROP TABLE IF EXISTS FestiSync.lugar ;

CREATE TABLE IF NOT EXISTS FestiSync.lugar (
    id_lugar INT NOT NULL AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
    ciudad VARCHAR(100) NOT NULL,
    aforo INT NULL,
    PRIMARY KEY (id_lugar)
) ENGINE = InnoDB;

-----
-- Table FestiSync.evento
-----

DROP TABLE IF EXISTS FestiSync.evento ;

CREATE TABLE IF NOT EXISTS FestiSync.evento (
    id_evento INT NOT NULL AUTO_INCREMENT,
    nombre VARCHAR(150) NOT NULL,
    fecha DATE NOT NULL,
    id_lugar INT NOT NULL,
    PRIMARY KEY (id_evento),
    INDEX fk_evento_lugar_idx (id_lugar ASC) VISIBLE,
```

```
CONSTRAINT fk_evento_lugar
FOREIGN KEY (id_lugar)
REFERENCES FestiSync.lugar (id_lugar)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- Table FestiSync.categoría
```

```
DROP TABLE IF EXISTS FestiSync.categoría ;
```

```
CREATE TABLE IF NOT EXISTS FestiSync.categoría (
    id_categoria INT NOT NULL AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
    id_evento INT NOT NULL,
    PRIMARY KEY (id_categoria),
    INDEX fk_categoria_evento1_idx (id_evento ASC) VISIBLE,
    CONSTRAINT fk_categoria_evento1
        FOREIGN KEY (id_evento)
        REFERENCES FestiSync.evento (id_evento)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
    ENGINE = InnoDB;
```

```
-- Table FestiSync.artista
```

```
DROP TABLE IF EXISTS FestiSync.artista ;
```

```
CREATE TABLE IF NOT EXISTS FestiSync.artista (
    id_artista INT NOT NULL AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
    genero_musical VARCHAR(100) NOT NULL,
    pais VARCHAR(100) NOT NULL,
    PRIMARY KEY (id_artista))
    ENGINE = InnoDB;
```

```
-- Table FestiSync.evento_has_artista
```

```
DROP TABLE IF EXISTS FestiSync.evento_has_artista ;
```

```
CREATE TABLE IF NOT EXISTS FestiSync.evento_has_artista (
```

```
evento_has_id INT NOT NULL,  
artista_has_id INT NOT NULL,  
PRIMARY KEY (evento_has_id, artista_has_id),  
INDEX fk_evento_has_artista_artista1_idx (artista_has_id ASC) VISIBLE,  
INDEX fk_evento_has_artista_evento1_idx (evento_has_id ASC) VISIBLE,  
CONSTRAINT fk_evento_has_artista_evento1  
    FOREIGN KEY (evento_has_id)  
        REFERENCES FestiSync.evento (id_evento)  
        ON DELETE NO ACTION  
        ON UPDATE NO ACTION,  
CONSTRAINT fk_evento_has_artista_artista1  
    FOREIGN KEY (artista_has_id)  
        REFERENCES FestiSync.artista (id_artista)  
        ON DELETE CASCADE  
        ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

```
--  
-- Table FestiSync.entrada  
--  
DROP TABLE IF EXISTS FestiSync.entrada ;
```

```
CREATE TABLE IF NOT EXISTS FestiSync.entrada (  
id_entrada INT NOT NULL AUTO_INCREMENT,  
tipo VARCHAR(50) NOT NULL,  
precio DECIMAL(4,2) NOT NULL,  
id_evento INT NOT NULL,  
PRIMARY KEY (id_entrada),  
INDEX fk_entrada_evento1_idx (id_evento ASC) VISIBLE,  
CONSTRAINT fk_entrada_evento1  
    FOREIGN KEY (id_evento)  
        REFERENCES FestiSync.evento (id_evento)  
        ON DELETE NO ACTION  
        ON UPDATE NO ACTION)  
ENGINE = InnoDB;
```

```
SET SQL_MODE=@OLD_SQL_MODE;  
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;  
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```