

Capturas de pantalla del funcionamiento.

AsientoRestController

- Obtener Asientos Disponibles por Zona y Evento

```
GET http://localhost:8080/api/asientos?zonalId=1&eventoId=2

Params • Authorization Headers (7) Body Scripts Settings
Body Cookies Headers (5) Test Results Visualize
{} JSON Preview Visualize

1 [
2   {
3     "id": {
4       "idEvento": 2,
5       "idAsiento": 1
6     },
7     "estado": "DISPONIBLE",
8     "evento": {
9       "idEvento": 2,
10      "nombreEvento": "Jazz Night con Herbie Hancock",
11      "fecha": "2025-07-22",
12      "duracion": 150,
13      "boletos": []
14    },
15    "asiento": {
16      "idAsiento": 1,
17      "numeroAsiento": "A-001",
18      "zona": {
19        "idZona": 1,
20        "nombreZona": "VIP",
21        "precio": 2500.0,
22        "capacidad": 20
23      },
24      "boleto": null,
25      "boletos": []
26    }
27  },
28  {
29    "id": {
30      "idEvento": 2,
31      "idAsiento": 2
32    },
```

- Obtener un AsientoEvento por idAsiento

```
GET http://localhost:8080/api/asientos/5

Params Authorization Headers (7) Body Scripts Setting
Body Cookies Headers (5) Test Results Visualize

{} JSON Preview Visualize

1 {
2   "id": {
3     "idEvento": 1,
4     "idAsiento": 5
5   },
6   "estado": "DISPONIBLE",
7   "evento": {
8     "idEvento": 1,
9     "nombreEvento": "Rock Fest 2025",
10    "fecha": "2025-06-10",
11    "duracion": 180,
12    "boletos": []
13  },
14  "asiento": {
15    "idAsiento": 5,
16    "numeroAsiento": "A-005",
17    "zona": {
18      "idZona": 1,
19      "nombreZona": "VIP",
20      "precio": 2500.0,
21      "capacidad": 20
22    },
23    "boleto": null,
24    "boletos": []
25  }
26 }
```

EventoRestController

- Obtener todos los eventos

HTTP <http://localhost:8080/api/eventos>

GET <http://localhost:8080/api/eventos>

Params Auth Headers (7) Body Scripts Settings

Query Params

	Key	Value	De
	Key	Value	De

Body [200 OK](#)

{ } JSON [Preview](#) [Visualize](#)

```
1  [
2      {
3          "idEvento": 1,
4          "nombreEvento": "Rock Fest 2025",
5          "fecha": "2025-06-10",
6          "duracion": 180,
7          "boletos": []
8      },
9      {
10         "idEvento": 2,
11         "nombreEvento": "Jazz Night con Herbie Hancock",
12         "fecha": "2025-07-22",
13         "duracion": 150,
14         "boletos": []
15     },
16     {
17         "idEvento": 3,
18         "nombreEvento": "Taylor Swift Eras Tour",
19         "fecha": "2025-08-15",
20         "duracion": 210,
21         "boletos": []
22     }
23 ]
```

- Obtener un evento por ID

GET

⌵

http://localhost:8080/api/eventos/1

ParamsAuthHeaders (7)BodyScriptsSettings

Query Params

	Key	Value	Desc
	Key	Value	Desc

Body⌵🕒200 OK

{ } JSON⌵▶ Preview🔗 Visualize⌵

```
1 {
2   "idEvento": 1,
3   "nombreEvento": "Rock Fest 2025",
4   "fecha": "2025-06-10",
5   "duracion": 180,
6   "boletos": []
7 }
```

- Crear un evento

The screenshot displays a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:8080/api/eventos
- Body Tab:** Selected, showing the request body in JSON format:

```
1 {
2   "nombreEvento": "Gorillaz",
3   "fecha": "2025-06-20",
4   "duracion": 120,
5   "tipoEvento": {
6     "idTipoEvento": 2
7   }
8 }
```
- Status:** 200 OK
- Response Body Tab:** Selected, showing the response body in JSON format:

```
1 {
2   "idEvento": 13,
3   "nombreEvento": "Gorillaz",
4   "fecha": "2025-06-20T00:00:00.000+00:00",
5   "duracion": 120,
6   "boletos": null
7 }
```

- Actualizar un evento

PUT http://localhost:8080/api/eventos/1

Params Auth Headers (10) Body **Scripts** Settings

raw JSON

```

1 {
2   "nombreEvento": "Luis Miguel",
3   "fecha": "2025-06-25",
4   "duracion": 180,
5   "tipoEvento": {
6     "idTipoEvento": 3
7   }
8 }
9
10

```

Body 200 OK

JSON Preview Visualize

```

1 {
2   "idEvento": 1,
3   "nombreEvento": "Luis Miguel",
4   "fecha": "2025-06-24",
5   "duracion": 180,
6   "boletos": []

```

- Eliminar un evento

DELETE http://localhost:8080/api/eventos/13

Params Auth Headers (10) Body **Scripts** Settings

Headers 9 hidden

	Key	Value	De...
<input type="checkbox"/>			
	Key	Value	Desi

Body 204 No Content

Raw Preview Visualize

```

1

```

TipoEventoRestController

- Obtener todos

GET ⌵ http://localhost:8080/api/tipo-evento

Params Auth Headers (10) Body ● Scripts Settings

Headers ⌵ 9 hidden

	Key	Value	
<input type="checkbox"/>			
	Key	Value	

Body ⌵ 🕒 200 OK

{} JSON ⌵ ▶ Preview 🔍 Visualize ⌵

```
1  [
2      {
3          "idTipoEvento": 10,
4          "tipo": "Banda",
5          "eventos": [
6              {
7                  "idEvento": 10,
8                  "nombreEvento": "Banda MS en Vivo",
9                  "fecha": "2025-07-18",
10                 "duracion": 180,
11                 "boletos": []
12             }
13         ]
14     },
15     {
16         "idTipoEvento": 7,
17         "tipo": "Blues",
18         "eventos": [
19             {
20                 "idEvento": 7,
21                 "nombreEvento": "BB King Tributo",
```

- Obtener por ID

GET http://localhost:8080/api/tipo-evento/1

Params Auth Headers (10) Body ● Scripts Settings

Headers 9 hidden

	Key	Value
<input type="checkbox"/>		
	Key	Value

Body

{ } JSON Preview Visualize

```
1 {
2   "idTipoEvento": 1,
3   "tipo": "Rock",
4   "eventos": []
5 }
```

- Crear nuevo

POST http://localhost:8080/api/tipo-evento

Params Auth Headers (10) Body ● Scripts Settings

raw JSON

```
1 {
2   "tipo": "Comedia"
3 }
4
5
6
```

Body

200 OK

{ } JSON Preview Visualize

```
1 {
2   "idTipoEvento": 11,
3   "tipo": "Comedia",
4   "eventos": null
5 }
```


- Actualizar

PUT http://localhost:8080/api/tipo-evento/8

Params Auth Headers (10) **Body** Scripts Settings

raw JSON

```

1 {
2   "tipo": "Comedia Musical"
3 }
4
5
6

```

Body

{ } JSON Preview Visualize

```

1 {
2   "idTipoEvento": 8,
3   "tipo": "Comedia Musical",
4   "eventos": [
5     {
6       "idEvento": 8,
7       "nombreEvento": "Tomorrowland Festival",
8       "fecha": "2025-12-25",
9       "duracion": 300,
10      "boletos": []
11    },
12    {
13      "idEvento": 11,
14      "nombreEvento": "Humbe",
15      "fecha": "2025-04-26",
16      "duracion": 120,
17      "boletos": []
18    }
19  ]
20 }

```

- Eliminar

DELETE http://localhost:8080/api/tipo-evento/9

Params Auth **Headers (10)** Body Scripts Settings

Headers 9 hidden

	Key	Value	De...
<input type="checkbox"/>			
	Key	Value	Desc

Body

204 No Content

Raw Preview Visualize

1

ZonaRestController

- Obtener todas las zonas

GET

Params Auth Headers (10) Body **●** Scripts Settings

Headers ☐ 9 hidden

	Key	Value
<input type="checkbox"/>		
	Key	Value

Body

{} JSON

```
1  [  
2      {  
3          "idZona": 1,  
4          "nombreZona": "VIP",  
5          "precio": 2500.0,  
6          "capacidad": 20  
7      },  
8      {  
9          "idZona": 2,  
10         "nombreZona": "General 1",  
11         "precio": 800.0,  
12         "capacidad": 50  
13     },  
14     {  
15         "idZona": 3,  
16         "nombreZona": "Preferente",  
17         "precio": 1500.0,  
18         "capacidad": 30  
19     },  
20 ]
```

- Obtener zona por ID

GET

⌵

http://localhost:8080/api/zonas/4

ParamsAuthHeaders (10)Body ●ScriptsSettings

Headers

👁 9 hidden

	Key	Value
<input type="checkbox"/>		
	Key	Value

Body

⌵🕒

{ } JSON ⌵

▶ Preview

🔍 Visualize ⌵

```
1  {
2    "idZona": 4,
3    "nombreZona": "Grada Baja",
4    "precio": 1000.0,
5    "capacidad": 55
6  }
```

- Crear nueva zona

POST

http://localhost:8080/api/zonas

ParamsAuthHeaders (10)BodyScriptsSettings

rawJSON

1

{

2

"nombreZona": "General",

3

"precio": 1000.0,

4

"capacidad": 200

5

}

6

7

8

Body

{}

JSON

Preview

Visualize

1

{

2

"idZona": 10,

3

"nombreZona": "General",

4

"precio": 1000.0,

5

"capacidad": 200

6

}

- Actualizar zona

The screenshot shows a REST client interface with a PUT request to `http://localhost:8080/api/zonas/1`. The request body is a JSON object with the following fields: `nombreZona` (VIP Modificada), `precio` (3000.0), and `capacidad` (25). The response is a 200 OK status with a JSON body containing the updated zone details: `idZona` (1), `nombreZona` (VIP Modificada), `precio` (3000.0), and `capacidad` (20).

```
PUT http://localhost:8080/api/zonas/1
```

Params Auth Headers (10) **Body** Scripts Settings

raw JSON

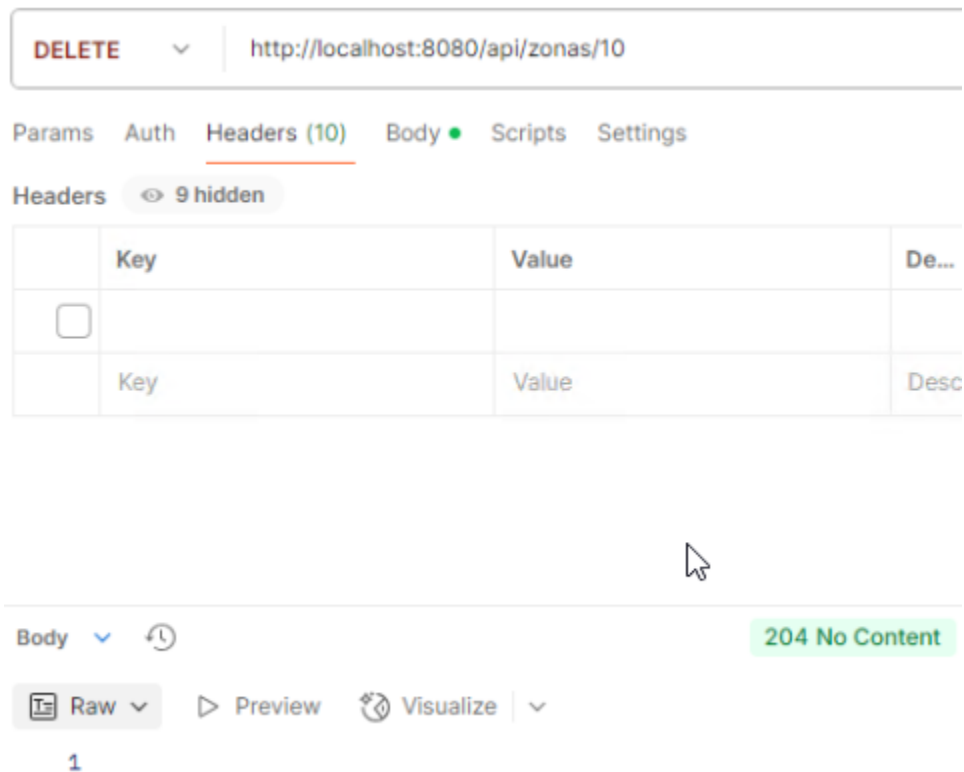
```
1 {
2   "nombreZona": "VIP Modificada",
3   "precio": 3000.0,
4   "capacidad": 25
5 }
```

Body 200 OK

{ JSON Preview Visualize

```
1 {
2   "idZona": 1,
3   "nombreZona": "VIP Modificada",
4   "precio": 3000.0,
5   "capacidad": 20
6 }
```

- Eliminar zona



Configuraciones necesarias para la aplicación de consumo.

Para que la aplicación RESTful funcione correctamente y se pueda consumir desde herramientas como Postman, o un cliente front-end, se realizaron las siguientes configuraciones:

- Dependencias en pom.xml
Se añadieron las dependencias necesarias para construir una aplicación con Spring Boot
- ```

<!-- Spring Web para crear controladores REST -->
<dependency>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-web</artifactId>
</dependency>

<!-- Spring Data JPA para el acceso a base de datos -->
<dependency>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

```

```
<!-- Driver de base de datos (MariaDB en este caso) -->
<dependency>
 <groupId>org.mariadb.jdbc</groupId>
 <artifactId>mariadb-java-client</artifactId>
 <version>3.4.1</version>
</dependency>
```

- Configuración en application.properties

```
Puerto del servidor
server.port=8080
```

```
Conexión a la base de datos MariaDB
spring.datasource.url=jdbc:mariadb://localhost:3306/prototype3
spring.datasource.username=tu_usuario
spring.datasource.password=tu_contraseña
```

```
Configuración JPA / Hibernate
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDBDialect
```

- Controladores REST habilitados  
Todos los controladores REST están anotados con @RestController y manejan rutas del tipo /api/  
sto permite consumir los datos desde clientes externos o Postman accediendo a rutas como:
  - GET http://localhost:8080/api/eventos
  - POST http://localhost:8080/api/zonas
- ScriptSQL  
Se pasará el script con las sentencias SQL y todo lo necesario para la conexión que se hará con el POSTMAN y con la información necesaria

### **Como acceder a la sección del consumo del RESTful.**

A continuación, se explica paso a paso cómo consumir la API RESTful desarrollada con Spring Boot utilizando Postman, una herramienta popular para realizar peticiones HTTP y probar servicios web:

#### **1. Iniciar la aplicación Spring Boot**

Antes de consumir cualquier endpoint REST, asegúrate de que la aplicación esté corriendo en el puerto configurado (por defecto 8080). Ejecuta el proyecto desde tu IDE (IntelliJ, Eclipse, etc.)

## 2.- Abrir Postman

## 3.- Acceder a los endpoints REST

En los ejemplos que se realizaron en la primera parte de este documento se pueden ver los diferentes endpoints para cada uno de los estos ejemplos.

## Documentación en Swagger

Una vez ejecutada la aplicación, la interfaz de Swagger se encuentra disponible en la siguiente URL:

<http://localhost:8080/swagger-ui/index.html>

Se aplicaron las siguientes anotaciones para enriquecer la documentación de Swagger:

- `@RestController` y `@RequestMapping`: delimitan los recursos y rutas.
- `@Tag`: describe el propósito del grupo de endpoints.
- `@Operation`: indica el objetivo de cada operación HTTP (GET, POST, PUT, DELETE).
- `@Parameter`: documenta los parámetros de entrada con descripción y ejemplos.

Tipo de Evento		Operaciones para gestionar los tipos de evento	^
GET	/api/tipo-evento/{id}	Obtener un tipo de evento por su ID	▼
PUT	/api/tipo-evento/{id}	Actualizar un tipo de evento existente	▼
DELETE	/api/tipo-evento/{id}	Eliminar un tipo de evento por su ID	▼
GET	/api/tipo-evento	Obtener todos los tipos de evento	▼
POST	/api/tipo-evento	Crear un nuevo tipo de evento	▼
Eventos		Operaciones CRUD para la gestión de eventos	^
GET	/api/eventos/{id}	Obtener un evento por ID	▼
PUT	/api/eventos/{id}	Actualizar un evento existente	▼
DELETE	/api/eventos/{id}	Eliminar un evento por ID	▼
GET	/api/eventos	Obtener todos los eventos	▼
POST	/api/eventos	Crear un nuevo evento	▼
Zonas		Operaciones para gestionar las zonas del evento	^
GET	/api/zonas/{id}	Obtener una zona específica por ID	▼
PUT	/api/zonas/{id}	Actualizar una zona existente por ID	▼
DELETE	/api/zonas/{id}	Eliminar una zona por ID	▼
GET	/api/zonas	Obtener todas las zonas disponibles	▼
POST	/api/zonas	Crear una nueva zona	▼



**Asientos** Operaciones para consultar asientos disponibles por zona y evento



GET	/api/asientos	Obtener asientos disponibles por zona y evento	▼
GET	/api/asientos/{idAsiento}	Obtener información de un asiento por ID	▼