

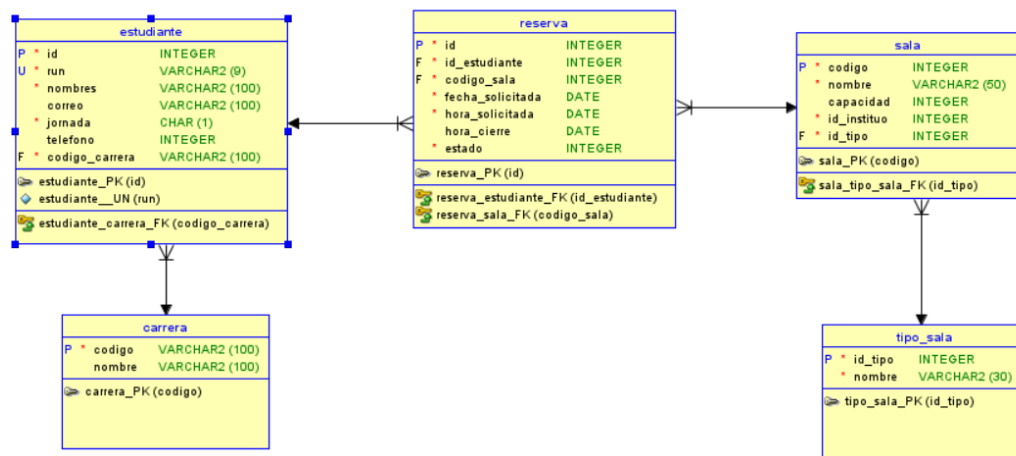
## Guía práctica

### Proyecto reserva de salas biblioteca

### Implementar Swagger



En esta guía, detallaremos paso a paso el desarrollo de un proyecto en Spring llamado "Salas Bibliotecas".



### Contexto del caso

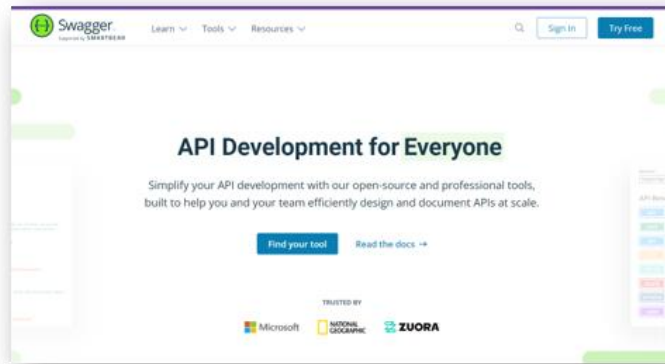
Este proyecto consiste en un sistema de reserva de salas desarrollado en Spring. El sistema permite que los estudiantes reserven salas disponibles.

### Estructura de la Base de Datos

La estructura de la base de datos incluye las siguientes tablas:

- Carrera, Estudiante, Reserva, Sala, Tipo de Sala

### Implementación de Swagger al proyecto



## Paso 1: Añadir Dependencias de Swagger

<https://springdoc.org/>


Primero, necesitas añadir las dependencias necesarias en tu archivo **pom.xml** (para proyectos Maven) o build.gradle (para proyectos Gradle).

```
<dependency>
  <groupId>org.springdoc</groupId>
  <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
  <version>2.6.0</version>
</dependency>
```

```
<dependency>
<groupId>org.springdoc</groupId>
<artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
<version>2.6.0</version>
</dependency>
```

Añadir las siguientes propiedades en **application.properties**.

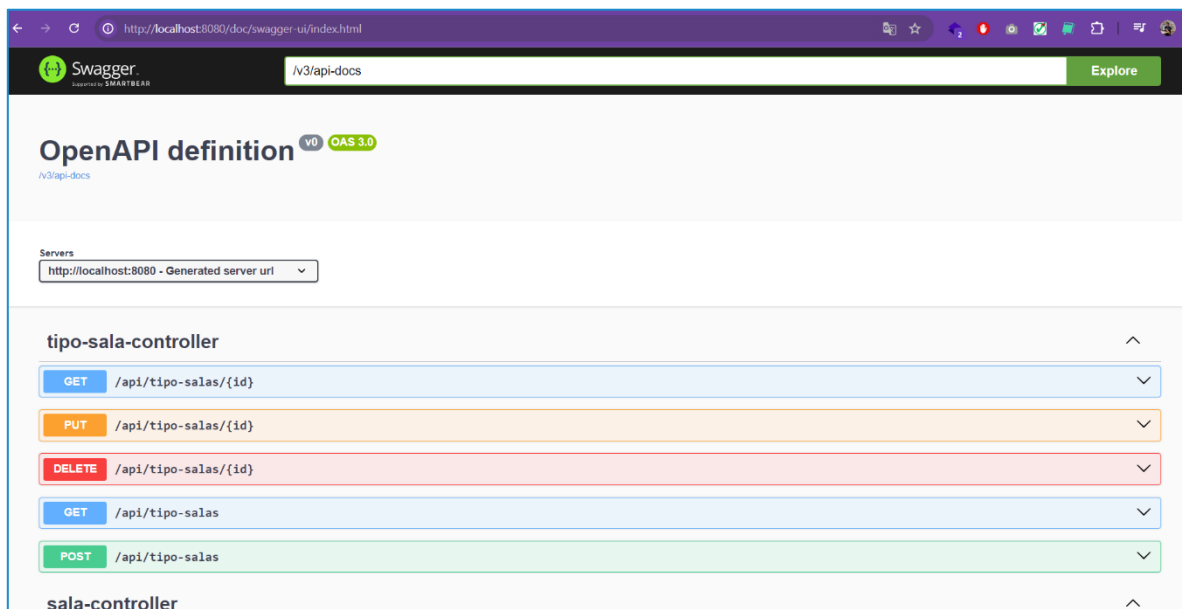
- springdoc.api-docs.enabled=true
- springdoc.swagger-ui.enabled=true
- springdoc.swagger-ui.path=/doc/swagger-ui.html



```
m pom.xml (biblioteca.salas.duoc) application.properties x
1  spring.application.name=biblioteca.salas.duoc
2  spring.profiles.active=dev
3
4  springdoc.api-docs.enabled=true
5  springdoc.swagger-ui.enabled=true
6  springdoc.swagger-ui.path=/doc/swagger-ui.html
```

### Ejecutamos el programa

- Ir a la siguiente ruta definida en las propiedades
  - <http://localhost:8080/doc/swagger-ui/index.html>



### Probemos la navegación de Swagger

- Ir a la api de las /api/salas, metodo GET
- Retornar todas las salas

sala-controller

GET /api/salas/{id}

PUT /api/salas/{id}

DELETE /api/salas/{id}

GET /api/salas

Parameters

No parameters

Execute

Clear

Responses

Curl

```

curl -X 'GET' \
  'http://localhost:8080/api/salas' \
  -H 'accept: */*'

```

Funciona perfectamente

Request URL

http://localhost:8080/api/salas

Server response

Code
Details

200

Response body

```

{
  "codigo": 2,
  "nombre": "South Olson",
  "capacidad": 46,
  "idInstituto": 8,
  "tipoSala": {
    "idTipo": 3,
    "nombre": "Horror"
  }
},
{
  "codigo": 3,
  "nombre": "Waters Institute",
  "capacidad": 97,
  "idInstituto": 8,
  "tipoSala": {
    "idTipo": 1,
    "nombre": "Tall tale"
  }
},
{
  "codigo": 4,
  "nombre": "Southern Barton College",
  "capacidad": 89,
  "idInstituto": 1,
  "tipoSala": {
    "idTipo": 3,
    "nombre": "Horror"
  }
}

```

Response headers

```

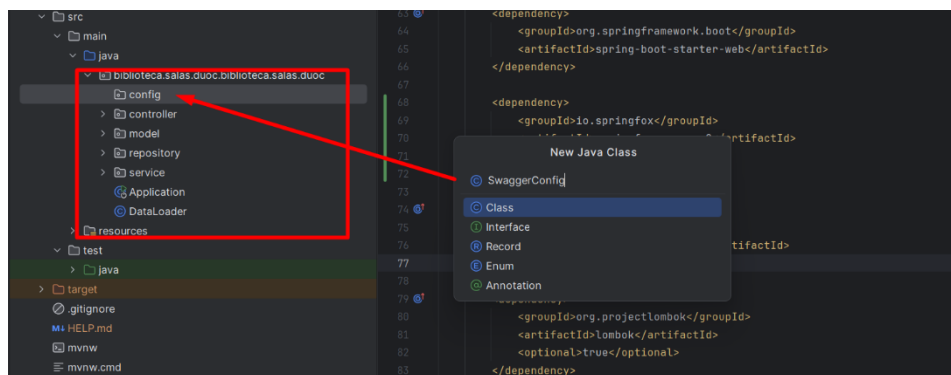
connection: keep-alive
content-type: application/json
date: Thu, 11 Jul 2024 14:31:59 GMT
keep-alive: timeout=60
transfer-encoding: chunked

```

Responses

## Configurar Swagger

Crea una clase de configuración para Swagger en tu proyecto Spring Boot. Por ejemplo, puedes crear una clase llamada SwaggerConfig.



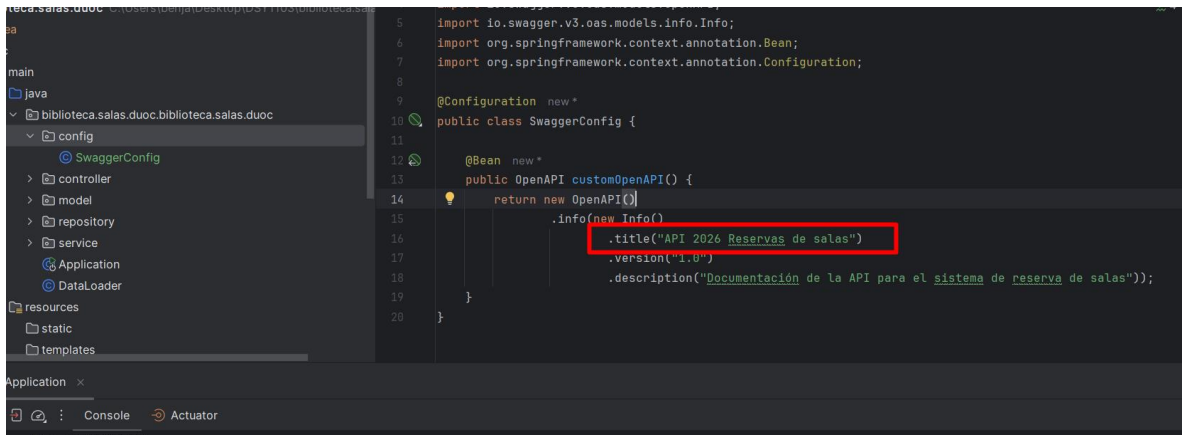
### Creación de la Clase de Configuración de Swagger

- Ubicación de la Clase: La clase `SwaggerConfig` debe estar en el paquete `config` o en un paquete que agrupe las configuraciones de la aplicación. Esto mantiene una estructura ordenada y facilita la gestión de configuraciones.

```

177 pom.xml (biblioteca.salas.duoc) application.properties SwaggerConfig.java
4   import io.swagger.v3.oas.models.OpenAPI;
5   import io.swagger.v3.oas.models.info.Info;
6   import org.springframework.context.annotation.Bean;
7   import org.springframework.context.annotation.Configuration;
8
9   @Configuration new *
10  public class SwaggerConfig {
11
12      @Bean new *
13      public OpenAPI customOpenAPI() {
14          return new OpenAPI()
15              .info(new Info()
16                  .title("API 2026 Reservas de salas")
17                  .version("1.0")
18                  .description("Documentación de la API para el sistema de reserva de salas"));
19      }
20  }
    
```

Vamos a parar la aplicación y volver a lanzarla

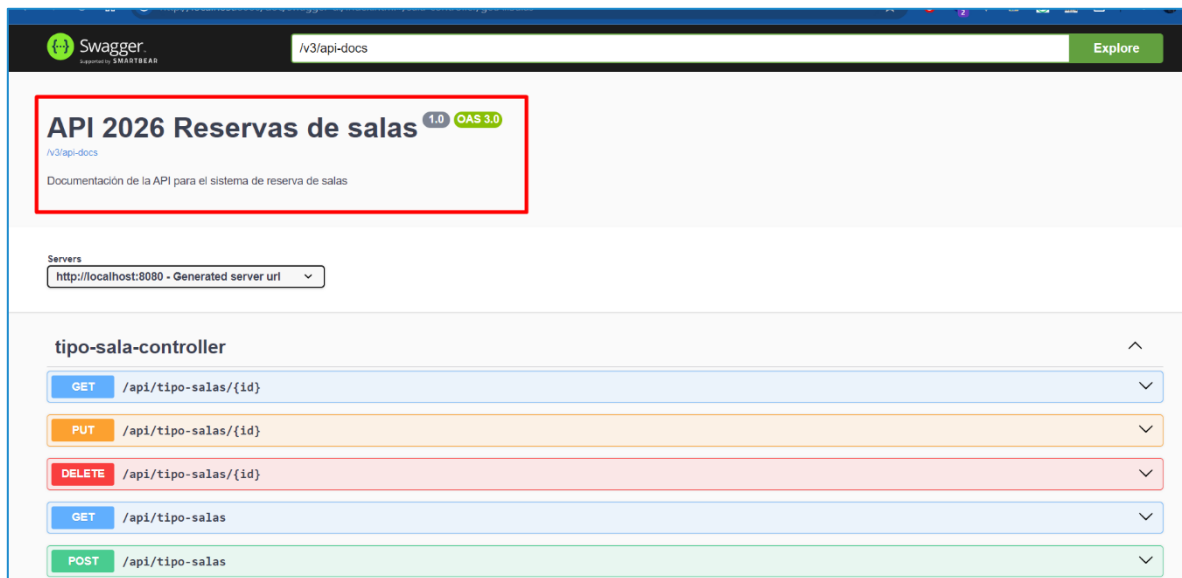


```

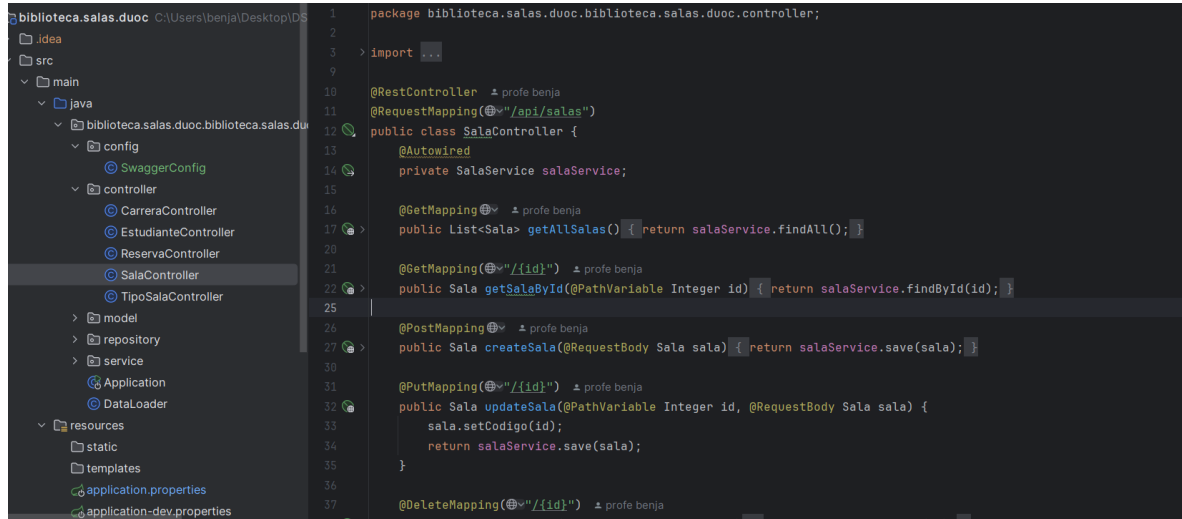
5 import io.swagger.v3.oas.models.info.Info;
6 import org.springframework.context.annotation.Bean;
7 import org.springframework.context.annotation.Configuration;
8
9 @Configuration
10 public class SwaggerConfig {
11
12     @Bean
13     public OpenAPI customOpenAPI() {
14         return new OpenAPI()
15             .info(new Info()
16                 .title("API 2026 Reservas de salas")
17                 .version("1.0")
18                 .description("Documentación de la API para el sistema de reserva de salas"));
19     }
20 }

```

Los cambios han sido reflejados.

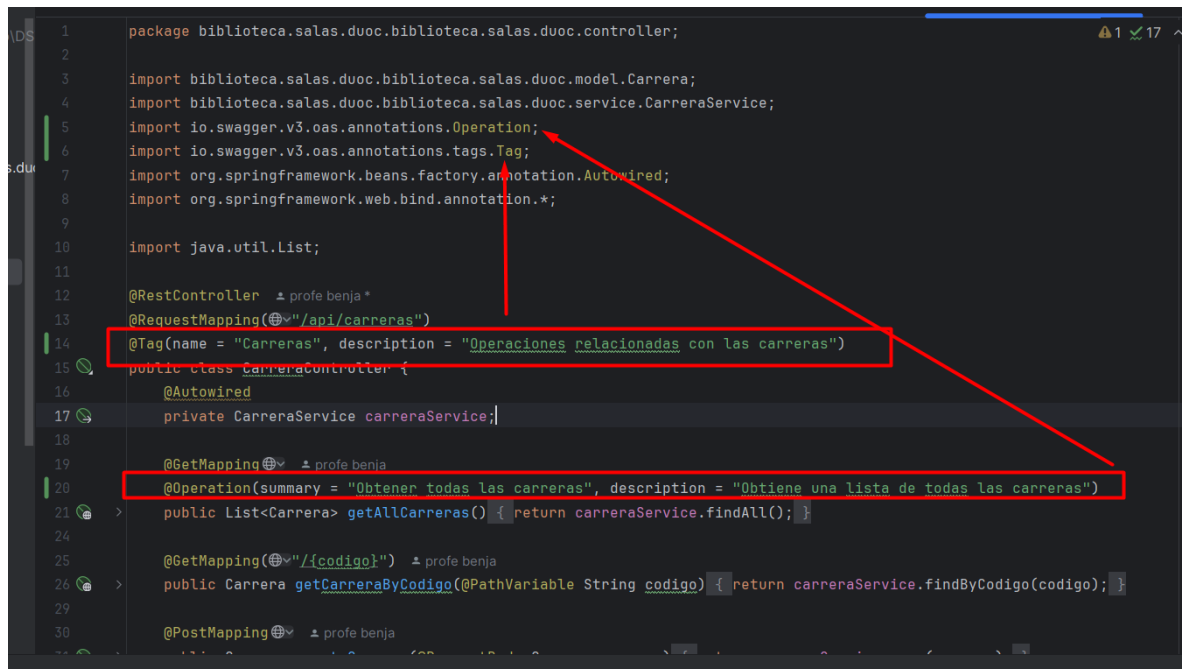


## Documentar controladores



```

1 package biblioteca.salas.duoc.biblioteca.salas.duoc.controller;
2
3 import ...
4
5 @RestController
6 @RequestMapping("/api/salas")
7 public class SalaController {
8     @Autowired
9     private SalaService salaService;
10
11     @GetMapping
12     public List<Sala> getAllSalas() { return salaService.findAll(); }
13
14     @GetMapping("/{id}")
15     public Sala getSalaById(@PathVariable Integer id) { return salaService.findById(id); }
16
17     @PostMapping
18     public Sala createSala(@RequestBody Sala sala) { return salaService.save(sala); }
19
20     @PutMapping("/{id}")
21     public Sala updateSala(@PathVariable Integer id, @RequestBody Sala sala) {
22         sala.setCodigo(id);
23         return salaService.save(sala);
24     }
25
26     @DeleteMapping("/{id}")
27 }
    
```



```

1 package biblioteca.salas.duoc.biblioteca.salas.duoc.controller;
2
3 import biblioteca.salas.duoc.biblioteca.salas.duoc.model.Carrera;
4 import biblioteca.salas.duoc.biblioteca.salas.duoc.service.CarreraService;
5 import io.swagger.v3.oas.annotations.Operation;
6 import io.swagger.v3.oas.annotations.tags.Tag;
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.web.bind.annotation.*;
9
10 import java.util.List;
11
12 @RestController
13 @RequestMapping("/api/carreras")
14 @Tag(name = "Carreras", description = "Operaciones relacionadas con las carreras")
15 public class CarreraController {
16     @Autowired
17     private CarreraService carreraService;
18
19     @GetMapping
20     @Operation(summary = "Obtener todas las carreras", description = "Obtiene una lista de todas las carreras")
21     public List<Carrera> getAllCarreras() { return carreraService.findAll(); }
22
23     @GetMapping("/{codigo}")
24     public Carrera getCarreraByCodigo(@PathVariable String codigo) { return carreraService.findByCodigo(codigo); }
25
26     @PostMapping
27     public Carrera createCarrera(@RequestBody Carrera carrera) { return carreraService.save(carrera); }
28
29     @DeleteMapping("/{codigo}")
30     public void deleteCarrera(@PathVariable String codigo) { carreraService.delete(codigo); }
31 }
    
```

## Anotaciones Comunes de Swagger (OpenAPI)

### @Tag

**Descripción:** Se usa para agrupar operaciones relacionadas y describir una sección de la API.

**Uso:** Colocado a nivel de clase del controlador para definir un grupo de endpoints.

**Ejemplo:**

```
@Tag(name = "Carreras", description = "Operaciones relacionadas con las carreras")
```

### @Operation

**Descripción:** Describe una operación o endpoint específico de la API.

**Uso:** Colocado a nivel de método en el controlador para documentar un endpoint específico.

**Ejemplo:**

```
@Operation(summary = "Obtener todas las carreras", description =  
"Obtiene una lista de todas las carreras")
```

### @ApiResponse

**Descripción:** Documenta una posible respuesta de un endpoint.

**Uso:** Usado dentro de @Operation para describir respuestas específicas.

**Ejemplo:**

```
@ApiResponse(responseCode = "200", description = "Operación exitosa")
```

### @ApiResponses

**Descripción:** Agrupa múltiples anotaciones @ApiResponse.

**Uso:** Usado dentro de @Operation para documentar múltiples respuestas posibles de un endpoint.

**Ejemplo:**

```
@ApiResponses(value = {  
    @ApiResponse(responseCode = "200", description = "Operación exitosa"),  
    @ApiResponse(responseCode = "404", description = "Carrera no encontrada")  
})
```

### @Parameter

**Descripción:** Describe un parámetro de un endpoint.

**Uso:** Colocado en los parámetros del método en el controlador.

**Ejemplo:**

```
@Parameter(description = "Código de la carrera", required = true)
```



## @RequestBody

**Descripción:** Documenta el cuerpo de una solicitud.

**Uso:** Colocado en los parámetros del método en el controlador para describir el cuerpo de la solicitud.

```
@RequestBody(description = "Carrera a crear", required = true)
```

## @Schema

**Descripción:** Describe un modelo o esquema utilizado en la API.

**Uso:** Colocado en las clases de modelo para describir sus propiedades.

**Ejemplo:**

```
@Schema(description = "Entidad que representa una carrera")
public class Carrera {
    @Schema(description = "Código de la carrera", example = "ING")
    private String codigo;

    @Schema(description = "Nombre de la carrera", example = "Ingeniería")
    private String nombre;
}
```

## @Content

**Descripción:** Define el contenido de una respuesta o un parámetro.

**Uso:** Usado dentro de @ApiResponse o @RequestBody para describir el contenido específico.

**Ejemplo:**

```
@ApiResponse(responseCode = "200", description = "Operación exitosa",
              content = @Content(mediaType = "application/json",
                                  schema = @Schema(implementation =
Carrera.class)))
```

## @ExampleObject

**Descripción:** Proporciona un ejemplo de un objeto o una respuesta.

**Uso:** Usado dentro de @Content para definir ejemplos.

**Ejemplo:**

```
@ExampleObject(name = "EjemploCarrera", value = "{\"codigo\": \"ING\",
\"nombre\": \"Ingeniería\"}")
```

## @ArraySchema

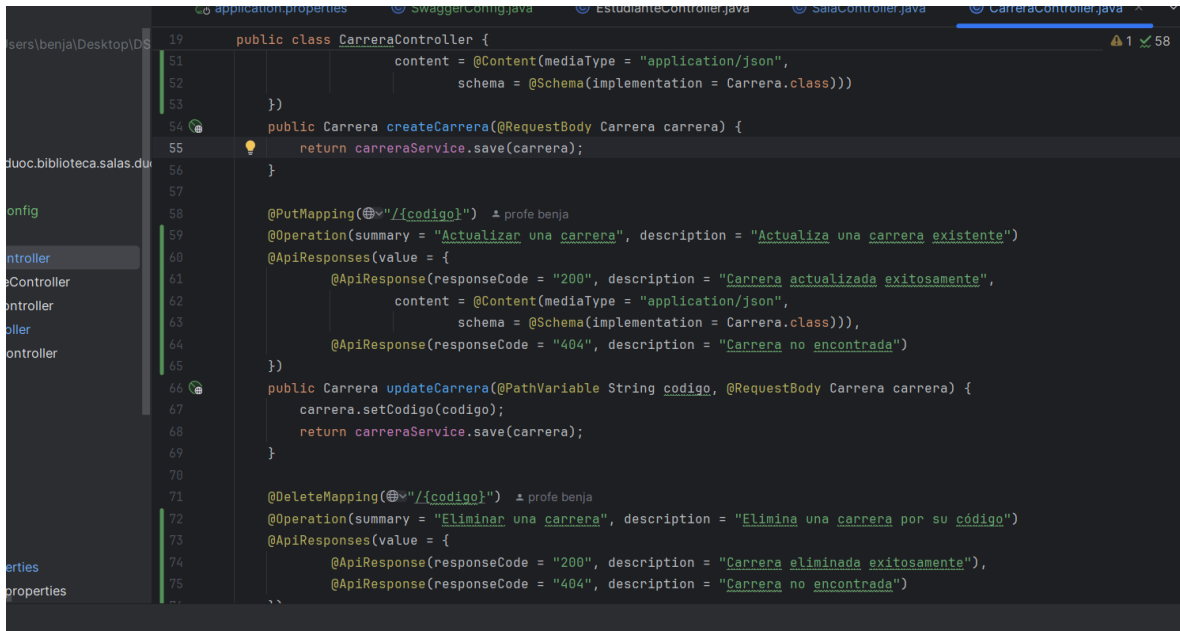
**Descripción:** Describe un esquema de un array.

**Uso:** Usado en una propiedad de tipo array en un modelo.

**Ejemplo:**

```
@ArraySchema(schema = @Schema(implementation = Carrera.class))
private List<Carrera> carreras;
```

## Probando cambios en CarreraController



```
19 public class CarreraController {
20     // ...
21     content = @Content(mediaType = "application/json",
22         schema = @Schema(implementation = Carrera.class)))
23     }
24     public Carrera createCarrera(@RequestBody Carrera carrera) {
25         return carreraService.save(carrera);
26     }
27
28     @PutMapping("/{codigo}")
29     @Operation(summary = "Actualizar una carrera", description = "Actualiza una carrera existente")
30     @ApiResponses(value = {
31         @ApiResponse(responseCode = "200", description = "Carrera actualizada exitosamente",
32             content = @Content(mediaType = "application/json",
33                 schema = @Schema(implementation = Carrera.class))),
34         @ApiResponse(responseCode = "404", description = "Carrera no encontrada")
35     })
36     public Carrera updateCarrera(@PathVariable String codigo, @RequestBody Carrera carrera) {
37         carrera.setCodigo(codigo);
38         return carreraService.save(carrera);
39     }
40
41     @DeleteMapping("/{codigo}")
42     @Operation(summary = "Eliminar una carrera", description = "Elimina una carrera por su código")
43     @ApiResponses(value = {
44         @ApiResponse(responseCode = "200", description = "Carrera eliminada exitosamente"),
45         @ApiResponse(responseCode = "404", description = "Carrera no encontrada")
46     })
47     // ...
48 }
```

## Mejoras reflejadas en carreras

Carreras Operaciones relacionadas con las carreras		
GET	/api/carreras/{codigo}	Obtener una carrera por código
PUT	/api/carreras/{codigo}	Actualizar una carrera
DELETE	/api/carreras/{codigo}	Eliminar una carrera
GET	/api/carreras	Obtener todas las carreras
POST	/api/carreras	Crear una nueva carrera

## Información detallada en el metodo put

Responses		
Code	Description	Links
200	Carrera actualizada exitosamente Media type <input type="text" value="application/json"/> <small>Controls Accept header.</small> Example Value   Schema <pre>{   "codigo": "string",   "nombre": "string" }</pre>	No links
404	Carrera no encontrada Media type <input type="text" value="/*"/> Example Value   Schema <pre>{   "codigo": "string",   "nombre": "string" }</pre>	No links

### **Actividad personalizacion de Swagger ui.**

- Mejora y personaliza swagger.
- Investiga y personaliza swagger bajo OpenIA OAS.
- Comparte con el Docente y tu compañero.

