

Guía de Swagger

ÍNDICE

<i>Dependencia</i>	<u>3</u>
<i>Definición</i>	<u>3</u>
<i>Visualización</i>	<u>4</u>
<i>Pruebas</i>	<u>5</u>

Dependencia

Antes de todo es importante añadir la dependencia de Swagger en el pom

```
<dependency>

    <groupId>org.springdoc</groupId>

    <artifactId>springdoc-openapi-ui</artifactId>

    <version>1.6.9</version>

</dependency>
```

Definición

En las clases **@RestController**, justo encima de la definición de la clase debemos poner la etiqueta **@Tag** como se indica en la imagen con una descripción y un nombre.

```
import org.springframework.web.bind.annotation.*;

no usages  Willy

@RestController
@Tag(description = "Provee los datos de los Productos", name = "Controlador de Productos")
public class ProductoController {
```

A continuación, encima del método y debajo del **@GetMapping**/**@PostMapping** se debe colocar la etiqueta **@Operation** con un summary que corresponderá con una breve explicación, y una descripción que aquí si puedes detallar un poco más.

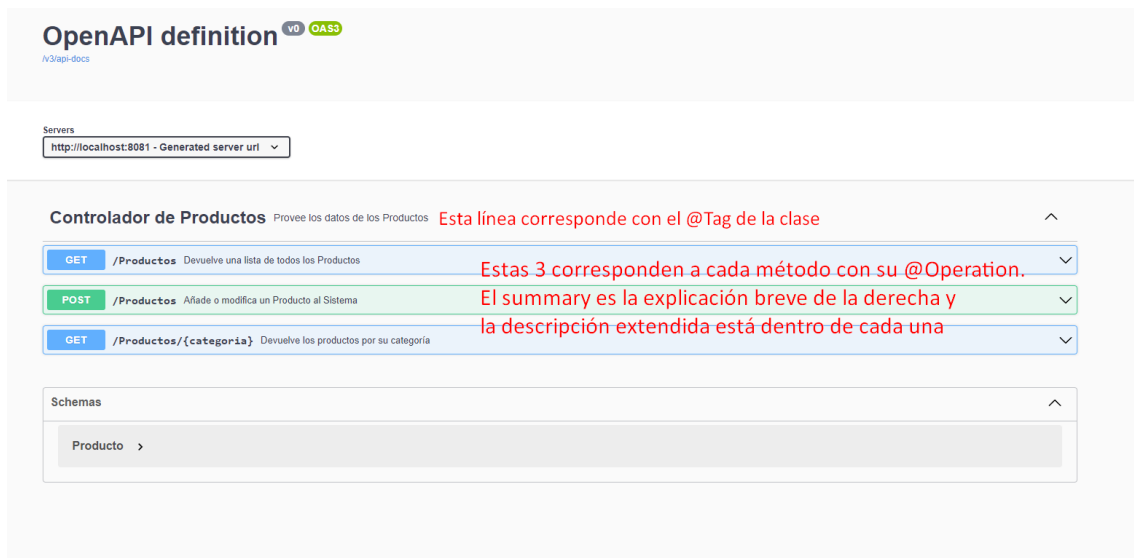
```
// GET
no usages  Willy

@GetMapping("/Productos")
@Operation(summary = "Devuelve una lista de todos los Productos",
           description = "Devuelve los datos pertinentes de los productos.")
public List<Producto> productos_GET() { return jsonDAO.leerJsonProductos(); }
```

Visualización

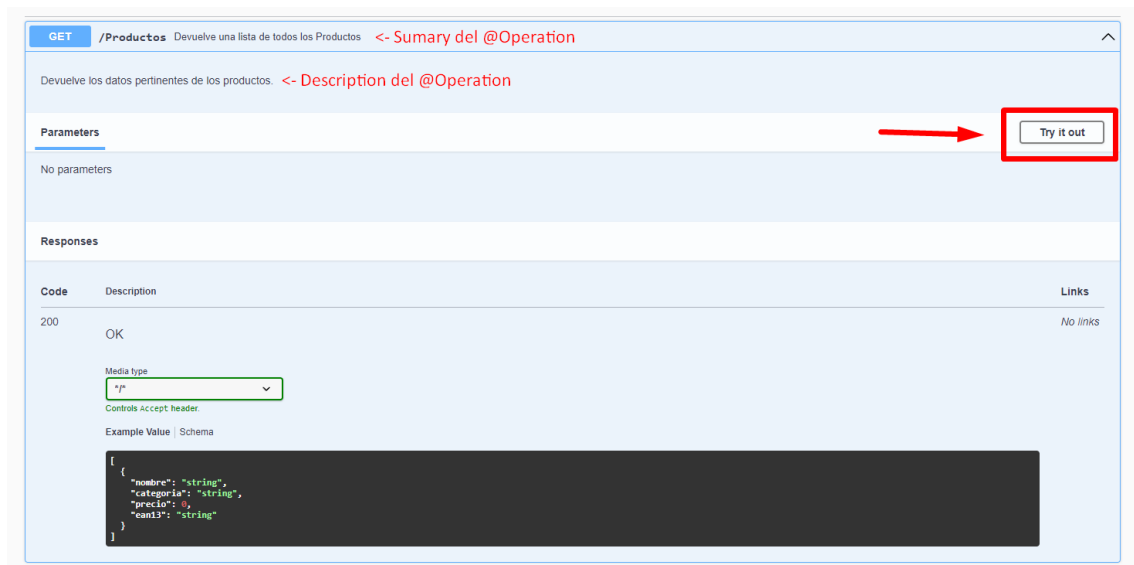
Para visualizar el resultado y hacer las pruebas del back, debéis ir al siguiente enlace: <http://localhost:8081/swagger-ui/index.html> cambiando el “8081” por el puerto correspondiente en el que ejecutéis el back.

Y como podéis ver en la imagen, hay una sección llamada en este caso **Controlador de Productos** que corresponde a lo que hayáis puesto en el **@Tag** de la clase, y debajo una sección por cada método mapeado con **@GetMapping/@PostMapping** que corresponde a lo que hayáis puesto en la etiqueta **@Operation**

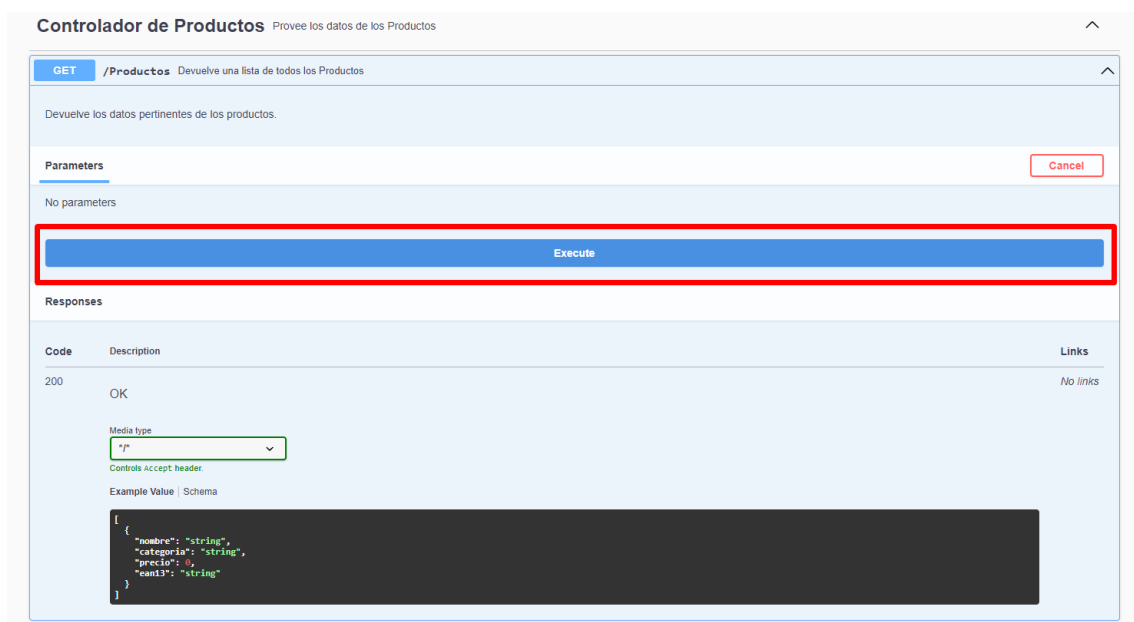


Pruebas

Para hacer las respectivas pruebas del back, simplemente despliegas el método que quieras probar, en este caso el **GET** general, que corresponde al primero y vemos lo siguiente:



Para probar el **GET** le damos primero al botón de "Try it out" de la derecha, nos saldrá un botón grande que pone **Execute**, le damos y podremos ver el resultado



En el caso de que funcione correctamente nos devolverá el código 200 que corresponde con el **HttpStatus.OK**, y visualizaremos el resultado como se muestra a continuación:

No parameters

Execute

Clear

Responses

Curl

curl -X 'GET' \n'http://localhost:8081/Productos' \n-H 'accept: */*'

Request URL

http://localhost:8081/Productos

Server response

Code

Details

200

Response body

```
{\n  {\n    "nombre": "producto",\n    "categoria": "categoria",\n    "precio": 0,\n    "ean13": "322500532677"\n  },\n  {\n    "nombre": "madera",\n    "categoria": "mueble",\n    "precio": 0,\n    "ean13": "0185388275285"\n  }\n}
```

Download

Response headers

```
connection: keep-alive\ncontent-type: application/json\ndate: Tue,04 Jul 2023 16:13:07 GMT\nkeep-alive: timeout=60\ntransfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	No links

En el caso del **POST** es similar, solo que tendremos que introducir un “Schema”, es decir como si agregáramos a mano un nuevo elemento en el json, teniendo en cuenta que si hay un elemento que se introduce automáticamente como una ID, no hay que ponerlo.

POST

/Productos

Añade o modifica un Producto al Sistema

Añade un nuevo producto si "nuevoCampo" es true (se dio al botón de crear nuevo en el front) en el caso de que "nuevoCampo" sea false significa que simplemente estamos modificando un elemento existente

Parameters

Cancel

No parameters

Request body

required

application/json

```
{\n  "nombre": "string",\n  "categoria": "string",\n  "precio": 0,\n  "ean13": "string"\n}
```

Execute

Responses

Aquí un ejemplo de un nuevo elemento introducido:

The screenshot shows a REST client interface with a green header bar. The method is **POST** and the URL is **/Productos**. Below the header, there is a description: "Añade o modifica un Producto al Sistema". A note explains: "Añade un nuevo producto si 'nuevoCampo' es true (se dio al botón de crear nuevo en el front) en el caso de que 'nuevoCampo' sea false significa que simplemente estamos modificando un elemento existente".

The **Parameters** tab is selected, showing "No parameters". There are **Cancel** and **Reset** buttons.

The **Request body** tab is selected, showing a JSON object:

```
{  "nombre": "silla",  "categoria": "mueble",  "precio": 30,  "ean13": "3545188274292"}
```

. The content type is set to **application/json**.

A large blue **Execute** button is at the bottom.

The **Responses** section is currently empty.

Al darle a ejecutar, si nuestro método funciona correctamente, nos mostrará el código 200, correspondiente al **HttpStatus.CREATE** y nos saldrá la lista con el nuevo elemento, como se muestra a continuación.

The screenshot shows the same REST client interface after the request has been executed. The **Execute** button is now greyed out, and a **Clear** button is visible.

The **Responses** section is expanded, showing the **Response body** tab. It displays a JSON array of three products. The third product, which is a chair, is highlighted with a red box:

```
{  "nombre": "silla",  "categoria": "mueble",  "precio": 30,  "ean13": "3545188274292"}
```

. A **Download** button is next to the response body.

The **Response headers** section shows:

```
connection: keep-alive  content-type: application/json
```

Y listo, aquí tienes una forma rápida y sencilla para probar el back a tu disposición, espero que os resulte útil. Suerte en el examen.