

Documentando Testing en Spring Boot

OAS es un estándar para definir la estructura de una API REST. Ha sido desarrollado por Swagger, pero hoy se conoce como Open API.

Cuando usar OAS:

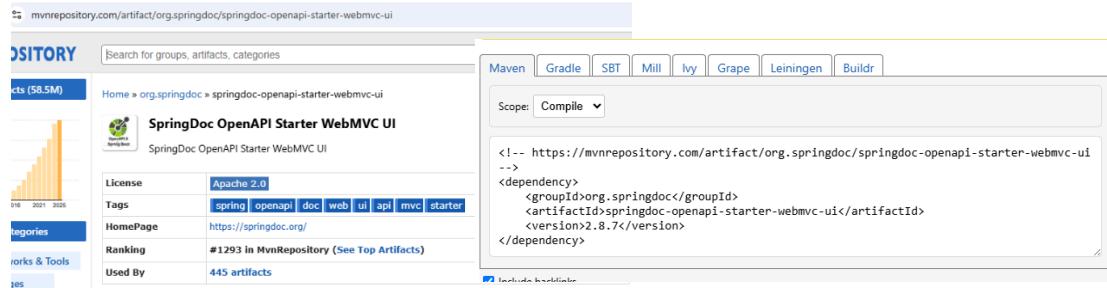
1. Cuando necesitamos documentar nuestra API.
2. Para generar pruebas documentadas.
3. Para facilitar el trabajo frontend y backend.

Ventajas:

- 1- Generación automática de documentación por medio de Swagger UI.
- 2- Compatible con herramientas como Postman, Insomnia, etc.
- 3- Facilita el testing y validación de API.

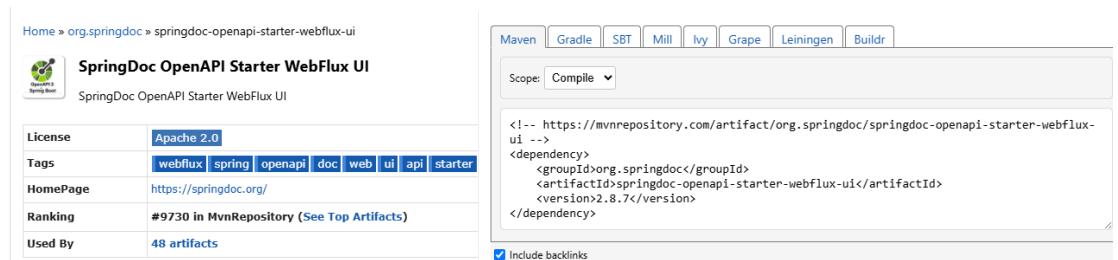
Implementaremos la documentación para nuestro ejemplo de services y restcontrollers del proyecto calculadora:

1. Accedemos a: <https://mvnrepository.com/> y procedemos a buscar las siguientes dependencias:
 - a. springdoc-openapi-starter-webmvc-ui: la dependencia de la biblioteca la añadimos al archivo pom.xml.



```
<!-- https://mvnrepository.com/artifact/org.springdoc/springdoc-openapi-starter-webmvc-ui
-->
<dependency>
  <groupId>org.springdoc</groupId>
  <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
  <version>2.8.7</version>
</dependency>
```

- b. springdoc-openapi-starter-webflux-ui, esta dependencia la añadimos también al archivo pom.



```
<!-- https://mvnrepository.com/artifact/org.springdoc/springdoc-openapi-starter-webflux-
ui
-->
<dependency>
  <groupId>org.springdoc</groupId>
  <artifactId>springdoc-openapi-starter-webflux-ui</artifactId>
  <version>2.8.7</version>
</dependency>
```

2. Añadir dependencias en pom:

```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
    <version>2.8.7</version>
</dependency>
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-starter-webflux-ui</artifactId>
    <version>2.8.7</version>
</dependency>
```

Recuerde que swagger permite documentar peticiones de tipo http, controller, restcontroller.

3. Añadimos los siguientes import y anotaciones en CalculadoraRestControllers:

```
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.media.Content;
import io.swagger.v3.oas.annotations.media.ExampleObject;
import io.swagger.v3.oas.annotations.responses.ApiResponse;
import io.swagger.v3.oas.annotations.responses.ApiResponses;
```

Con el fin de que swagger documente las peticiones de nuestros métodos implementados en el componente rest, debemos añadir sobre cada método lo siguiente:

Método sumar:

```

@Operation(summary = "Suma dos números")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "Suma exitosa",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = "8"))
    ),
    @ApiResponse(responseCode = "400", description = "Parámetros inválidos")
})

@GetMapping("/sumar")
public int sumar(@RequestParam int a, @RequestParam int b){
    return calculadoraservice.sumar(a, b);
}

```

Método multiplicar:

```

@Operation(summary = "Multiplica dos números")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "Multiplicación exitosa",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = "20"))
    ),
    @ApiResponse(responseCode = "400", description = "Parámetros inválidos")
})

@GetMapping("/multiplicar")
public int multiplicar(@RequestParam int a, @RequestParam int b){
    return calculadoraservice.multiplicar(a, b);
}

```

Método restar:

```
@Operation(summary = "Resta dos números")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "Resta exitosa",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = "2"))
    ),
    @ApiResponse(responseCode = "400", description = "Parámetros inválidos")
})

@GetMapping("/restar")
public int restar(@RequestParam int a, @RequestParam int b){
    return calculadoraservice.restar(a, b);
}
```

Método dividir:

```

@Operation(summary = "Divide dos números")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "División exitosa",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = "2")),
    ),
    @ApiResponse(responseCode = "400", description = "División por cero no permitida",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = "No se puede dividir por cero"))
)
})
@GetMapping("/dividir")
public int dividir(@RequestParam int a, @RequestParam int b){
    return calculadoraservice.dividir(a, b);
}

```

- 4.** En el archivo application-properties, configuramos la ruta de nuestro archivo html para la ejecución de swagger:

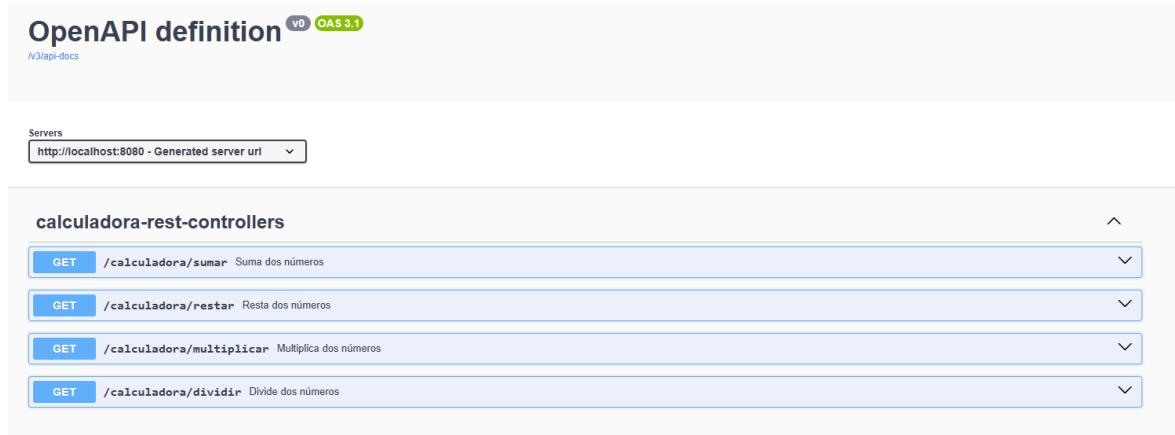
```

main > resources > application.properties
spring.application.name=testingservices
springdoc.swagger-ui.path=/swagger-ui.html

```

- 5.** Ejecutamos la aplicación y accedemos a la siguiente ruta:

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



The screenshot shows the OpenAPI definition interface. At the top, it says "OpenAPI definition v0 OAS 3.1". Below that is a "Servers" dropdown menu containing the URL "http://localhost:8080 - Generated server url". The main area displays the "calculadora-rest-controllers" section with four listed endpoints:

- GET /calculadora/sumar Suma dos números
- GET /calculadora/restar Resta dos números
- GET /calculadora/multiplicar Multiplica dos números
- GET /calculadora/dividir Divide dos números

/calculadora/sumar:

calculadora-rest-controllers

GET /calculadora/sumar Suma dos números

Parameters

Name	Description
a * required	integer(\$int32) a (query)
b * required	integer(\$int32) b (query)

Responses

Code	Description	Links
200	Suma exitosa	No links
400	Parámetros inválidos	No links

Media type: text/plain

Example Value |

8

Media type: */*

Example Value | Schema

0

/calculadora/restar:

GET /calculadora/restar Resta dos números

Parameters

Name	Description
a * required	integer(\$int32) a (query)
b * required	integer(\$int32) b (query)

Responses

Code	Description	Links
200	Resta exitosa	No links
400	Parámetros inválidos	No links

Media type: text/plain

Example Value |

2

Media type: */*

Example Value | Schema

0

/calculadora/multiplicar

GET /calculadora/multiplicar Multiplica dos números

Parameters

Name	Description
a * required	integer(\$int32) (query)
b * required	integer(\$int32) (query)

Responses

Code	Description	Links
200	Multiplicación exitosa Media type <code>text/plain</code> Controls Accept header. Example Value 28	No links
400	Parámetros inválidos Media type <code>*/*</code> Example Value Schema 0	No links

/calculadora/dividir

GET /calculadora/dividir Divide dos números

Parameters

Name	Description
a * required	integer(\$int32) (query)
b * required	integer(\$int32) (query)

Responses

Code	Description	Links
200	División exitosa Media type <code>text/plain</code> Controls Accept header. Example Value 2	No links
400	División por cero no permitida Media type <code>text/plain</code> Example Value No se puede dividir por cero	No links