DuocUC

mail: v.pobletel@profesor.duoc.cl
m.robinson@profesor.duoc.cl
edu.baeza@profesor.duoc.cl

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:

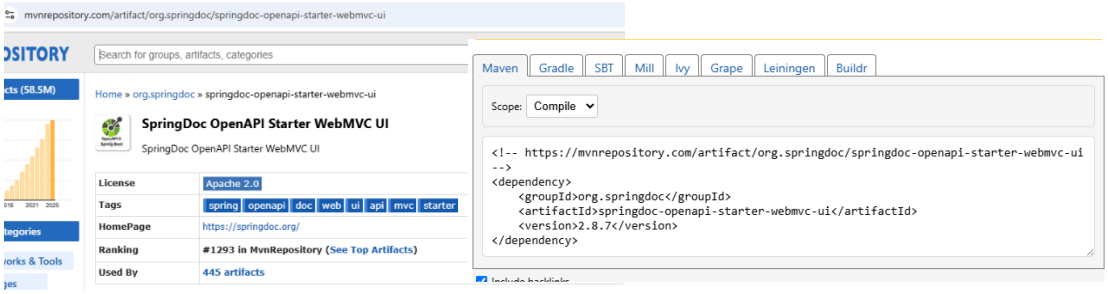
1. Cuando necesitamos documentar nuestra API.
2. Para generar pruebas documentadas.
3. Para facilitar el trabajo frontend y backend.

Ventajas:

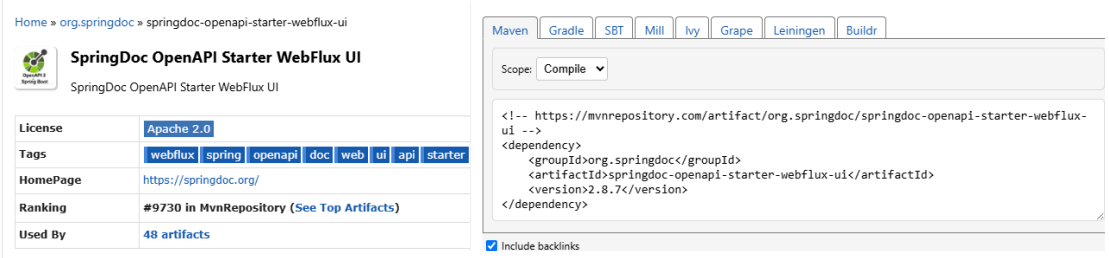
1- Generación automática de documentación por medio de Swagger UI.
2- Compatible con herramientas como Postman, Insomnia, etc.
3- Facilita el testing y validación de API.

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.



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



mail: v.pobletel@profesor.duoc.cl
m.robinson@profesor.duoc.cl
edu.baeza@profesor.duoc.cl

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:

mail: v.pobletel@profesor.duoc.cl
m.robinson@profesor.duoc.cl
edu.baeza@profesor.duoc.cl

```
@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:

mail: v.pobletel@profesor.duoc.cl
m.robinson@profesor.duoc.cl
edu.baeza@profesor.duoc.cl

```java
@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);
}
```

mail: v.pobletel@profesor.duoc.cl
m.robinson@profesor.duoc.cl
edu.baeza@profesor.duoc.cl

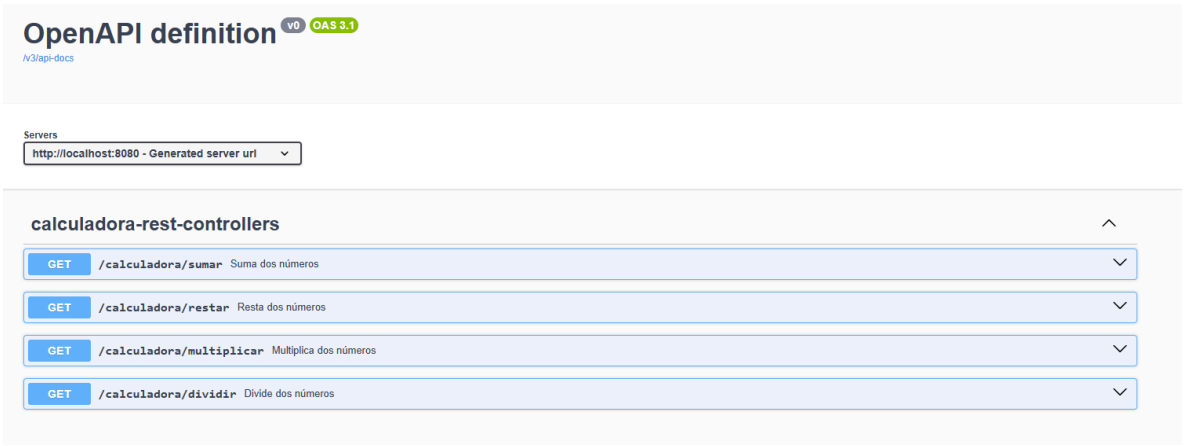
Método dividir:

```
@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:

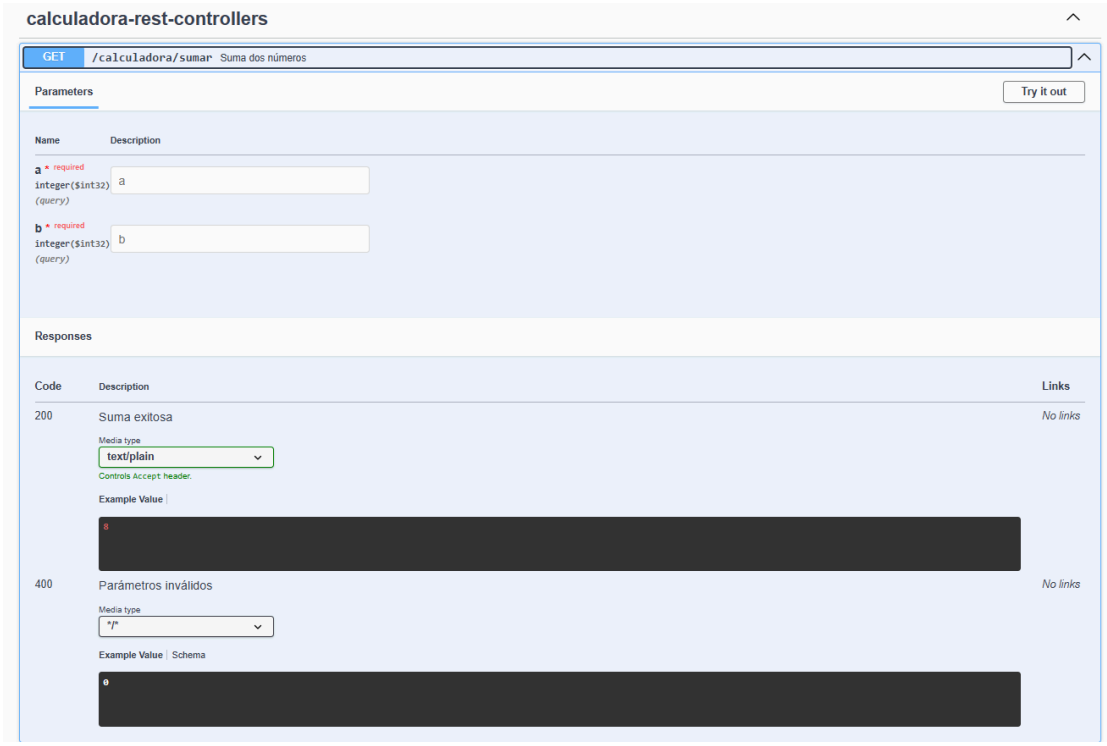
```
main > resources > application.properties
spring.application.name=testingservices
springdoc.swagger-ui.path=/swagger-ui.html
```

5. Ejecutamos la aplicación y accedemos a la siguiente ruta:
http://localhost:8080/swagger-ui/index.html

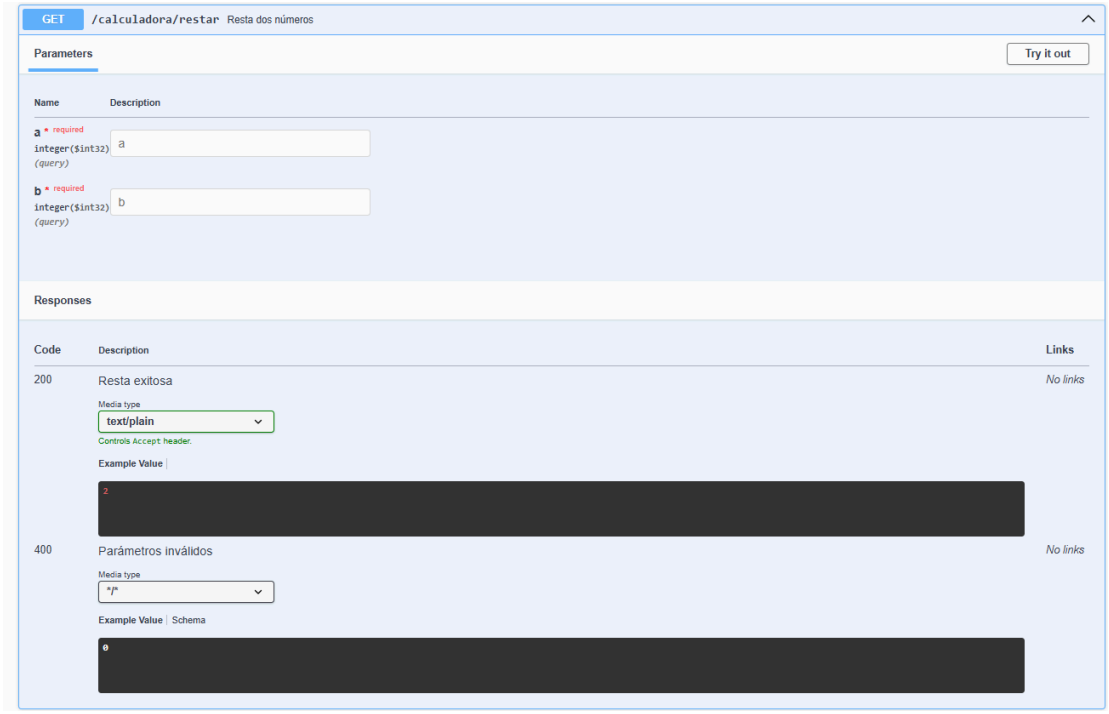


mail: v.pobletel@profesor.duoc.cl
m.robinson@profesor.duoc.cl
edu.baeza@profesor.duoc.cl

/calculadora/sumar:

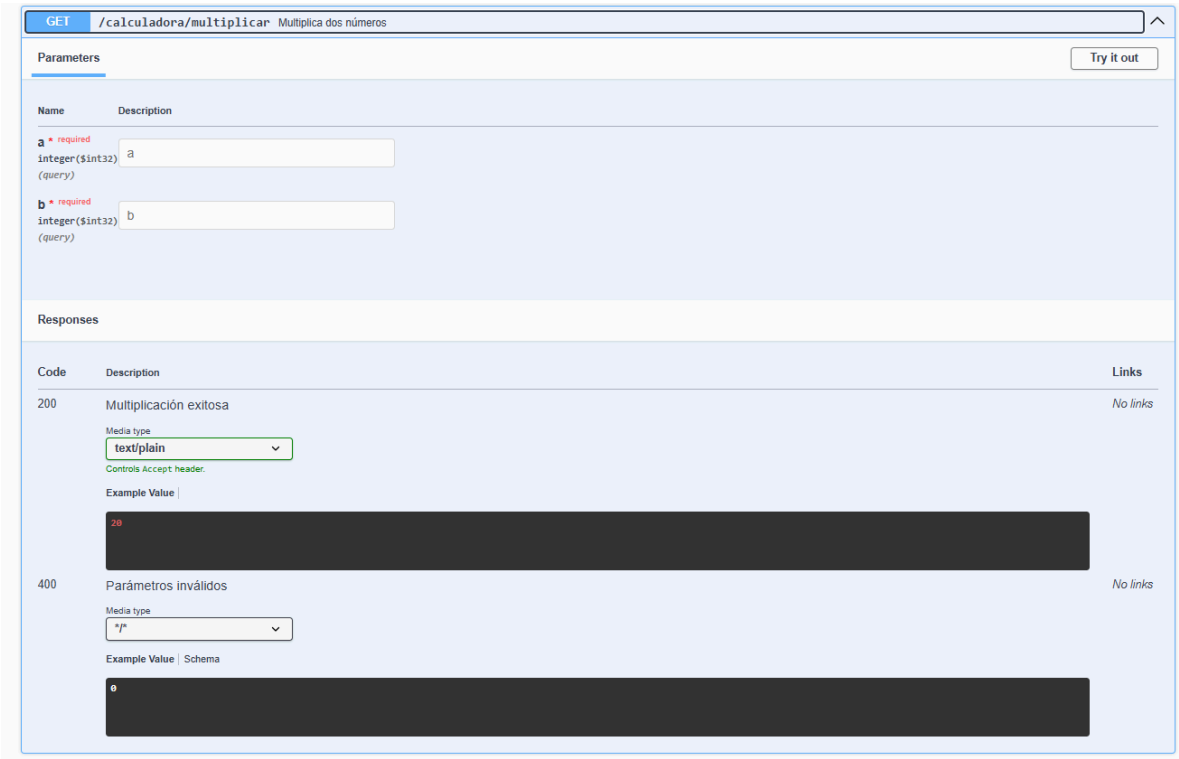


/calculadora/restar:



mail: v.pobletel@profesor.duoc.cl
m.robinson@profesor.duoc.cl
edu.baeza@profesor.duoc.cl

/calculadora/multiplicar



/calculadora/dividir