

¿Qué es Swagger?

Desarrollo de un API REST con Spring Boot

Documentación de nuestra API

- Una API que no está documentada, posiblemente sea difícil de utilizar.
 - No todo el mundo entiende lo mismo por REST.
 - En ocasiones, se implementan reglas de validación que nos obligan a utilizar tipos de datos concretos.
- ¿Cómo crear esta documentación?
 - Spring REST Docs vs Swagger + Swagger UI

Spring Rest Docs

- Combina documentación escrita a mano con Asciidoctor y fragmentos autogenerados producidos con Spring MVC Test.
 - Positivo
 - Proyecto dentro del paraguas de Spring
 - Nos *obliga* a escribir tests y tenerlos actualizados
 - Negativo
 - Nos obliga a escribir tests.

Swagger

- Swagger es un framework de código abierto respaldado por un gran ecosistema de herramientas que ayuda a los desarrolladores a diseñar, construir, documentar y consumir servicios web RESTful.
- Una de las más utilizadas es *Swagger UI tool*.
- Para nosotros, Swagger es una serie de reglas, especificaciones y herramientas que nos ayudan a documentar nuestras APIs.

Swagger

- *Utiliza* un json que incluye toda la documentación de nuestra API

```
"paths": {  
  "/pet": {  
    "post": {  
      "tags": [  
        "pet"  
      ],  
      "summary": "Add a new pet to the store",  
      "description": "",  
      "operationId": "addPet",  
      "consumes": [  
        "application/json",  
        "application/xml"  
      ],  
      "produces": [  
        "application/xml",  
        "application/json"  
      ],  
    },  
  },  
}
```

Swagger UI

- Es capaz de transformar ese JSON y hacerlo *interactivo*

pet Everything about your Pets

POST **/pet** Add a new pet to the store

PUT **/pet** Update an existing pet

GET **/pet/findByStatus** Finds Pets by status

Swagger UI

- Nos permite probar las peticiones, incluso con nuestros propios datos

The screenshot shows the Swagger UI for a **POST /pet** endpoint. The header bar is green and contains the text "POST /pet Add a new pet to the store". Below this is a "Parameters" section, which is currently empty. The main body of the interface is a light green box. On the left side of this box, the word "body" is followed by a red asterisk and the word "required". Below "body" are the labels "object" and "(body)". To the right of these labels is the text "Pet object that needs to be added to the store". Below this text are two links: "Edit Value" and "Model". The central area of the body is a white box containing a JSON object:

```
{  "id": 0,  "category": {    "id": 0,    "name": "string"  },  "name": "doggie",  "photoUrls": [    "string"  ],  "tags": [    {      "id": 0,      "name": "string"    }  ],  "status": "available"}
```

 At the bottom left of the body is a red "Cancel" button. At the bottom right is a dropdown menu labeled "Parameter content type" with "application/json" selected.

POST /pet Add a new pet to the store

Parameters

Name Description

body * required
object
(body)
Pet object that needs to be added to the store
Edit Value Model

```
{  "id": 0,  "category": {    "id": 0,    "name": "string"  },  "name": "doggie",  "photoUrls": [    "string"  ],  "tags": [    {      "id": 0,      "name": "string"    }  ],  "status": "available"}
```

Cancel

Parameter content type
application/json

**Ahora, vamos a integrar Swagger en
nuestro proyecto**