

# DOCUMENTACIÓN API SUPERHEROES

Diego Cantos Escribano

<b>Superhéroe</b>	<b>1</b>
<b>Seguridad</b>	<b>2</b>
<b>Endpoints</b>	<b>2</b>
<b>Dockerfile</b>	<b>7</b>

## Superhéroe

Se ha realizado el desarrollo de una api que permita el mantenimiento de superhéroes. Sobre estos superhéroes se van a realizar varias acciones:

- Consultar a todos los superhéroes.
- Consultar un único súper héroe por id.
- Consultar todos los súper héroes que contienen, en su nombre, el valor de un parámetro enviado en la petición. Por ejemplo, si enviamos “man” devolverá “Spiderman”, “Superman”, “Manolito el fuerte”, etc.
- Modificar un súper héroe.
- Eliminar un súper héroe.
- Test unitarios de algún servicio.

Los datos se guardan en una base de datos h2 que estará en memoria. Junto a la api hay 2 archivos sql, schema.sql y data.sql que servirán para inicializar la base de datos y rellenarla con información.

La base de datos tiene 2 tablas: la tabla de usuarios y la tabla de superhéroes. La tabla de usuarios guarda el nombre del usuario, su contraseña, sus roles y su id y sirve para poder loguearse con usuarios que tengan permisos para hacer las peticiones que modifiquen la base de datos. La tabla de superhéroes guarda la información sobre los superhéroes con su id y nombre.

La api se levanta en el puerto 8080.

# Seguridad

La aplicación tiene una seguridad básica en los accesos a los endpoints que puedan modificar la base de datos. Estos son el endpoint para eliminar superhéroes y el que se utilizará para modificar los datos de un superhéroe. Los datos del usuario que puede entrar son nombre de usuario: usuario y contraseña: 1234. En base de datos la contraseña se guarda encriptada mediante Bcrypt.

## Endpoints

Los endpoints de la api son:

### Get Superheroes

Devuelve la lista completa de superhéroes.

<http://localhost:8080/superheroe/>

```
curl --location --request GET 'http://localhost:8080/superheroe/' \
--header 'Cookie: JSESSIONID=7BF7F721D1487F83D7858983C31EB2B7' \
--data-raw ''
```

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:8080/superheroe/
- Params:** Authorization, Headers (12), Body, Pre-request Script, Test Results
- Query Params:** A table with two columns: KEY and Value. The first row has 'Key' in the KEY column and an empty Value column.
- Body:** Cookies (1), Headers (11), Test Results
- Response Format:** Pretty, Raw, Preview, Visualize, JSON (selected)
- Response Content:** A JSON array of three objects, each representing a superhero with an 'id' and a 'nombre'.

```
1 [
2   {
3     "id": 1,
4     "nombre": "Spiderman"
5   },
6   {
7     "id": 2,
8     "nombre": "Manin el mago"
9   },
10  {
11    "id": 3,
12    "nombre": "Superprueba"
13  }
14 ]
```

## Get Superheroe

Devuelve el superhéroe indicado en la petición

<http://localhost:8080/superheroe/{id}/>

```
curl --location --request GET 'http://localhost:8080/superheroe/1/' \
--header 'Cookie: JSESSIONID=7BF7F721D1487F83D7858983C31EB2B7' \
--data-raw ''
```

GET

⌵

http://localhost:8080/superheroe/1/

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Query Params

	KEY	VALUE
--	-----	-------

Body

Cookies (1)

Headers (10)

Test Results

Pretty

Raw

Preview

Visualize

JSON ⌵

⌵

1

2

3

4

{

"id": 1,


"nombre": "Spiderman"

}


## Get Superheroe filtrado

Devuelve los superhéroes que contengan la palabra indicado en el nombre.

<http://localhost:8080/superheroe?nombre={nombre}>

PruebaTecnica / Get Superheroes Filtro 

---

GET  http://localhost:8080/superheroe?nombre=man



Params • Authorization Headers (8) Body Pre-request Script

Query Params

KEY
-----

---

Body Cookies (1) Headers (10) Test Results

Pretty Raw Preview Visualize JSON  


```
1  []
2  {
3    "id": 1,
4    "nombre": "Spiderman"
5  },
6  {
7    "id": 2,
8    "nombre": "Manin el mago"
9  }
10 []
```

## Patch Superheroe modificar


Modifica el nombre del superhéroe que corresponda a la id. Para hacerlo el usuario tiene que autenticarse

<http://localhost:8080/superheroe/{id}?nombre={nombre}>

```
curl --location --request PATCH 'http://localhost:8080/superheroe/1/?nombre=superman' \  
--header 'Authorization: Basic dXN1YXJpbzoxMjM0' \  
--header 'Cookie: JSESSIONID=1190F76F4D486E6B1981EC7057E1B106' \  
--data-raw "
```



PruebaTécnica / Patch Superheroe Modificar 

---

PATCH 


http://localhost:8080/superheroe/1/?nombre=superman


---


Params  Authorization  Headers (10) Body Pre-request Script Tests Settings

---

Type 

Basic Auth 

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#) 

 Heads up! These parameters hold sensitive data. To keep this data secure while

Username 

usuario

Password 

....

☐ Show Password

Body Cookies (1) Headers (11) Test Results

Pretty Raw Preview Visualize JSON  

1

2

3

4

1

2

3

4

"id": 1,

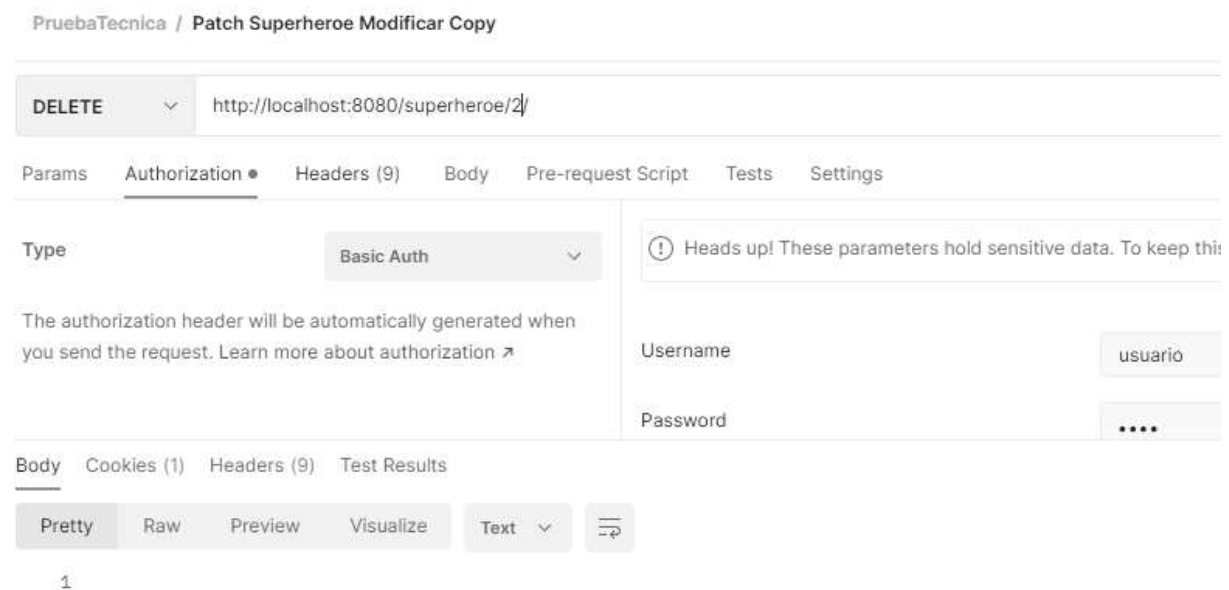
"nombre": "superman"

## Delete Superheroe modificar

Elimina el superheroe al que pertenece la id. Para hacerlo el usuario tiene que autenticarse.

<http://localhost:8080/superheroe/{id}/>

```
curl --location --request DELETE 'http://localhost:8080/superheroe/2' \  
--header 'Authorization: Basic dXN1YXJpbzoxMjM0' \  
--header 'Cookie: JSESSIONID=1190F76F4D486E6B1981EC7057E1B106' \  
--data-raw ''
```



Se entrega un json con la colección de los endpoints para postman

## Dockerfile

Además de la api, se entrega un dockerfile con la imagen del jar de la api. Ejecutando la imagen la api se levanta y ya se puede utilizar.

Para construir la imagen y ejecutarlo se han utilizado dos comandos.

Para construirlo

```
docker build -t spring-boot-docker:spring-docker .
```

Para ejecutarlo

```
docker run -p 8080:8080 spring-boot-docker:spring-docker .
```