Contenido

[1. Introducción 1](#_Toc133423276)

[2. Funcionamiento del servicio 1](#_Toc133423277)

[2.1. Añadir 2](#_Toc133423278)

[2.2. Modificar 3](#_Toc133423279)

[2.3. Lista completa 4](#_Toc133423280)

[2.4. Obtener por id 5](#_Toc133423281)

[2.5. Borrar 6](#_Toc133423282)

[2.6. Buscar por patrón 8](#_Toc133423283)

[3. Test 9](#_Toc133423284)

[4. Comentarios 9](#_Toc133423285)

# Introducción

El API está hecha JDK 17 y con Spring Boot 3.

Dispone de un servicio rest (controlador) que recibe peticiones relativas a héroes.

Cada controlador hace las llamadas a la clase de “servicio” que se encarga de la lógica de negocio e invoca al repositorio Crud de las clases mapeadas en las entidades o a otros servicios.

Las entidades de los héroes tienen 3 datos disponibles cada uno (id, nombre y otros) y

El servicio rest (controlador), siempre retornará un único tipo de objeto que contendrá el resultado de la consulta o los errores si los hubiera.

Los datos se almacenan en la BBDD H2 local.

# Entradas del servicio

Se ha creado una entrada para cada una de las opciones disponibles:

Consultar todos los súper héroes.

Consultar un único súper héroe por id.

Consultar héroes por patrón en el nombre.

Crear un súper héroe.

Modificar un súper héroe.

Eliminar un súper héroe.

A continuación se detalla cómo usar cada una de las llamadas con ayuda de postman.

## Añadir

Dado unos datos de héroe en formato Json, se añadirá al registro si no existe préviamente.

Desde postman (POST) http://localhost:8080/he/heroe

JSON ejemplo:

{

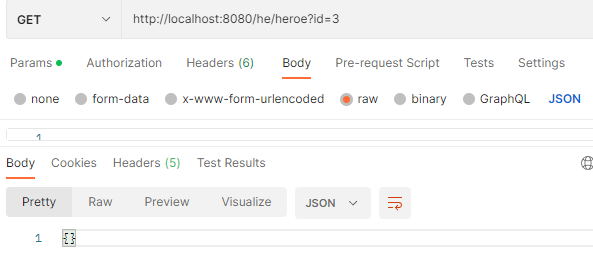
   "id":"3",

   "nombre":"añadido",

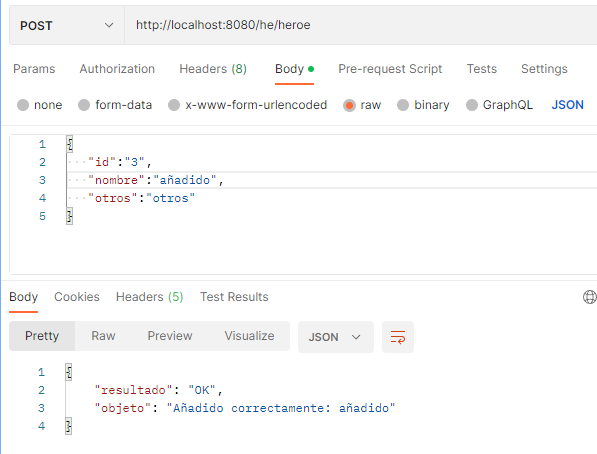
   "otros":"otros"

}

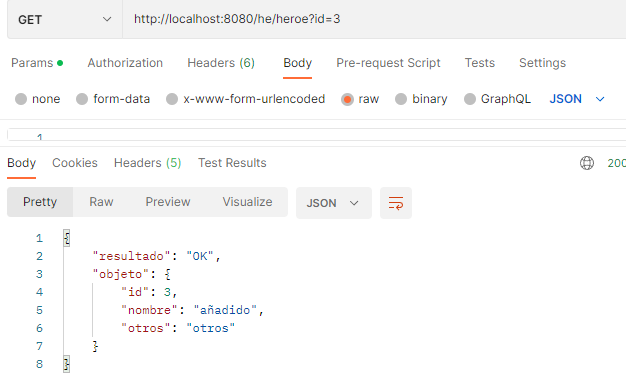
Validamos que no existe nadie con el id 3:



Enviamos para añadir el nuevo héroe:



Posteriormente ya se puede recuperar correctamente:



## Modificar

Dados unos datos de héroe enviado en formato Json y a partir de su id, se modifica el registro existente con los nuevos datos.

Desde postman (PUT) http://localhost:8080/he/heroe

JSON ejemplo:

{

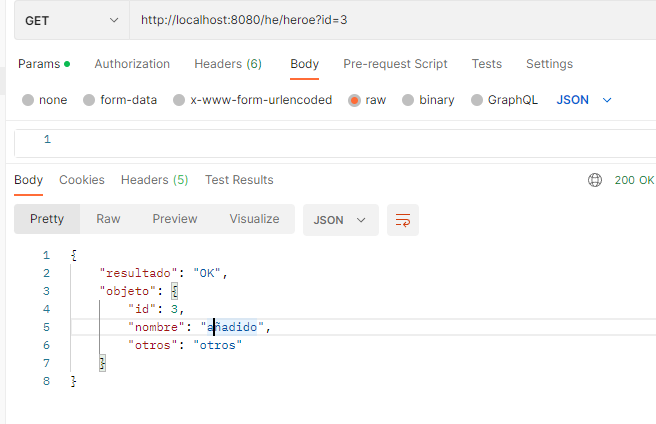
    "id":"3",

    "nombre":"supermodifi",

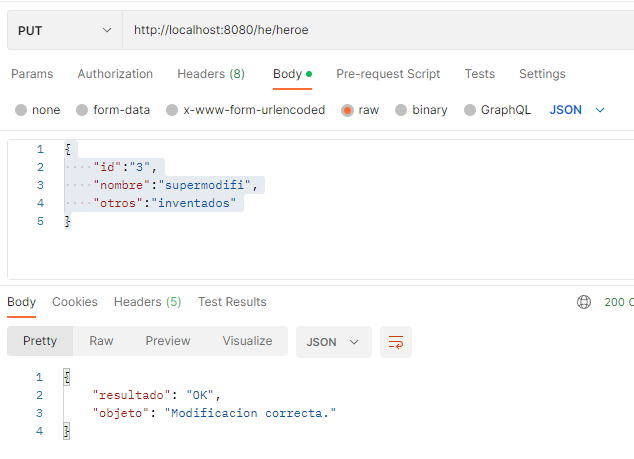
    "otros":"inventados"

}

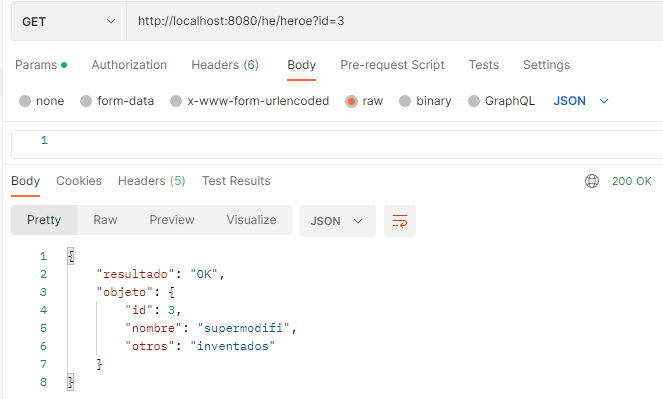
Inicialmente el héroe con id 3 dispone de los siguientes datos:



Se ejecuta el modificar:



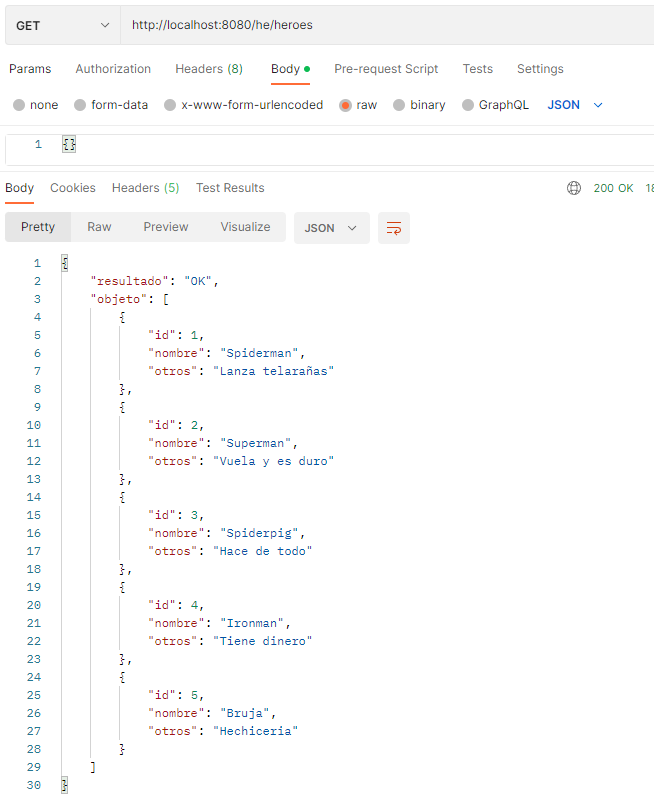
Se valida que los datos se han modificado al recuperar el héroe:



## Lista completa

Retorna la lista de animales existentes en bbdd.

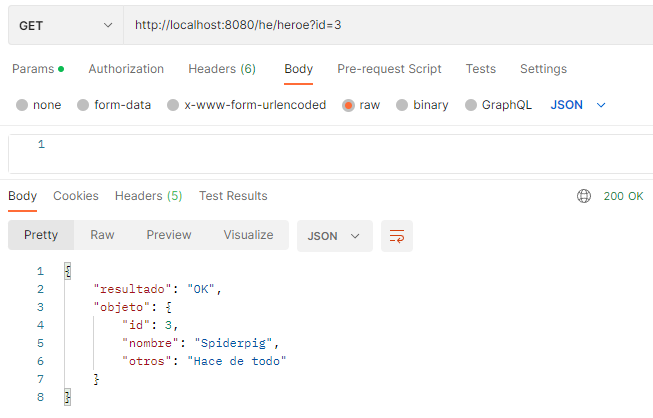
Desde postman (GET) http://localhost:8080/he/heroes



## Obtener por id

Busca un héroe con el id indicado por parámetro.

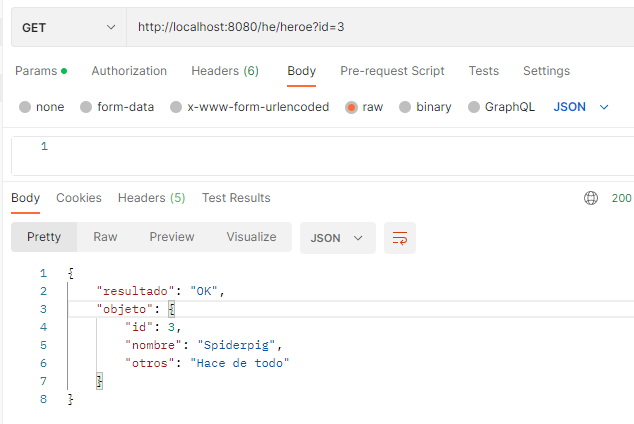
Desde postman (GET) http://localhost:8080/he/heroe?id={id del héroe a buscar}



## Borrar

Permite borrar un héroe por su id, si existe:

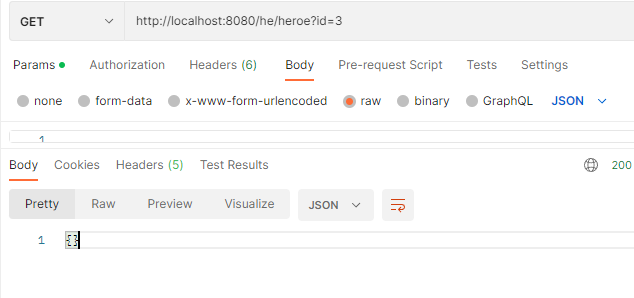
Desde postman (DELETE) http://localhost:8080/he/heroe?id={id del héroe a borrar}

Existiendo inicialmente el héroe: 

Se ejecuta el borrado de ese id.



Posteriormente no se encuentra el héroe:



También se puede validar intentando borrarlo de nuevo:

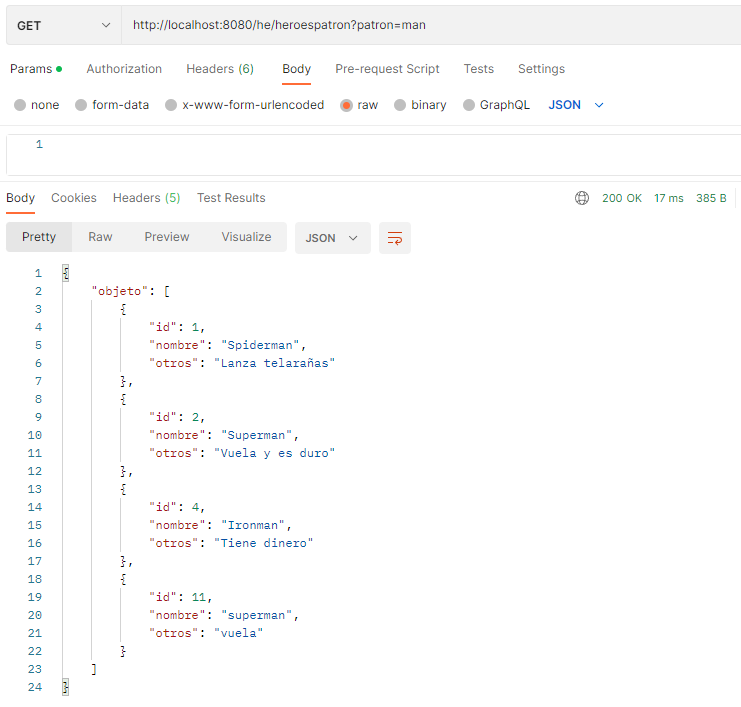


## Buscar por patrón

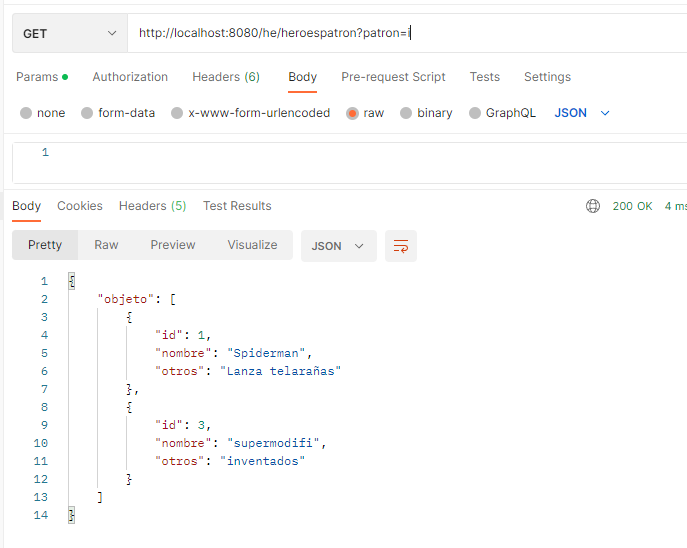
Realiza una búsqueda entre todos los nombres y se retornan aquellos que contienen la cadena facilitada por el patrón.

Desde postman (GET) http://localhost:8080/he/heroespatron?patron={patrón a buscar}

Buscando por patrón “man”:

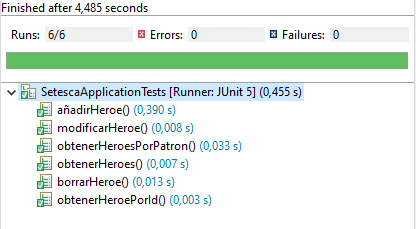


Buscando por patrón “i”:



# Test

Para cada una de las entradas que ofrece el servicio rest, se ha añadido un test:



# Comentarios

Se ha creado un fichero “data.sql” para tener unos registros iniciales.

Dado que el proyecto se ha hecho prácticamente del tirón y no hay varias subidas, para el git únicamente se han usado 2 ramas.

Para la creación inicial del Git se ha realizado en main y posteriormente se ha creado la rama master.

Se ha subido la documentación en master y mergeado en main:

