

Desarrollo rápido de aplicaciones.

Actividad Spring Boot.

Jesús Fornieles Muñoz

Índice:

3. Actividades	3
3.2. Añadir un listado de superpoderes a los héroes	3

3. Actividades

3.1. Crear servicios REST para la entidad Hero

Se ha seguido el tutorial de Angular "Tour of Heroes" adaptando la parte de backend con Spring Boot para crear un servicio REST que suministre la información de los héroes desde una base de datos PostgreSQL.

Entidad Hero

Se creó la entidad Hero que incluye:

- id (Long)
- name (String)

Añadir entidad Power

Se ha creado la entidad Power (Superpoder), relacionada con Hero mediante la anotación @OneToMany.

Relación utilizada:

```
@OneToMany(cascade = CascadeType.ALL, orphanRemoval = true)
@JoinColumn(name = "hero_id")
private List<Power> powers;
```

Entidad heroe:

```
1  package com.example.demo.entity;
2
3  import jakarta.persistence.*;
4  import java.util.List;
5
6  @Entity
7  public class Hero {
8
9      @Id
10     @GeneratedValue(strategy = GenerationType.AUTO)
11     private Long id;
12
13     private String name;
14
15     @OneToMany(cascade = CascadeType.ALL, orphanRemoval = true)
16     @JoinColumn(name = "hero_id")
17     private List<Power> powers;
18
19     public Hero() {}
20
21     public Hero(String name, List<Power> powers) {
22         this.name = name;
23         this.powers = powers;
24     }
25
26     public Long getId() {
27         return id;
28     }
29
30     public String getName() {
31         return name;
32     }
33
34     public List<Power> getPowers() {
35         return powers;
36     }
37
38     public void setId(Long id) {
39         this.id = id;
40     }
41
42     public void setName(String name) {
43         this.name = name;
44     }
45
46     public void setPowers(List<Power> powers) {
47         this.powers = powers;
48     }
49 }
```

Entidad Power:

```
1  package com.example.demo.entity;
2
3  import jakarta.persistence.*;
4
5  @Entity
6  public class Power {
7
8      @Id
9      @GeneratedValue(strategy = GenerationType.AUTO)
10     private Long id;
11
12     private String name;
13
14     public Power() {}
15
16     public Power(String name) {
17         this.name = name;
18     }
19
20     public Long getId() {
21         return id;
22     }
23
24     public String getName() {
25         return name;
26     }
27
28     public void setId(Long id) {
29         this.id = id;
30     }
31
32     public void setName(String name) {
33         this.name = name;
34     }
35 }
```

Controlador HeroController

Se expusieron los siguientes endpoints:

- GET /api/heroes — Obtener todos los héroes.
- POST /api/heroes — Crear un nuevo héroe con lista de superpoderes.

```
1  package com.example.demo.controller;
2
3  import com.example.demo.dto.HeroDTO;
4  import com.example.demo.entity.Hero;
5  import com.example.demo.mapper.HeroMapper;
6  import com.example.demo.repository.HeroRepository;
7  import org.springframework.web.bind.annotation.*;
8
9  import java.util.List;
10 import java.util.stream.Collectors;
11
12 @RestController
13 @RequestMapping("/heroes")
14 public class HeroController {
15
16     private final HeroRepository heroRepository;
17     private final HeroMapper heroMapper;
18
19     public HeroController(HeroRepository heroRepository, HeroMapper heroMapper) {
20         this.heroRepository = heroRepository;
21         this.heroMapper = heroMapper;
22     }
23
24     @GetMapping
25     public List<HeroDTO> getAllHeroes() {
26         return heroRepository.findAll().stream()
27             .map(heroMapper::toDto)
28             .collect(Collectors.toList());
29     }
30
31     @PostMapping
32     public HeroDTO createHero(@RequestBody HeroDTO heroDTO) {
33         Hero hero = heroMapper.toEntity(heroDTO);
34         Hero savedHero = heroRepository.save(hero);
35         return heroMapper.toDto(savedHero);
36     }
37 }
```

Pruebas de funcionamiento:

Vamos a utilizar thunder Client, que es una extensión de vscode que nos permite probar apis de forma sencilla:

Post:

The screenshot shows the Thunder Client interface with a POST request to `http://localhost:8080/api/users`. The request body is a JSON object: `{ "name": "Pedro", "surname": "Garcia", "email": "pedro.garcia@example.com" }`. The response status is `201 Created` with a size of `227 Bytes` and a time of `191 ms`. The response body is a JSON object: `{ "name": "Pedro", "email": "pedro.garcia@example.com", "_links": { "self": { "href": "http://localhost:8080/api/users/1" }, "user": { "href": "http://localhost:8080/api/users/1" } } }`.

Get:

The screenshot shows the Thunder Client interface with a GET request to `http://localhost:8080/api/users`. The response status is `200 OK` with a size of `487 Bytes` and a time of `19 ms`. The response body is a JSON object: `{ "_embedded": { "users": [{ "name": "Pedro", "email": "pedro.garcia@example.com", "_links": { "self": { "href": "http://localhost:8080/api/users/1" }, "user": { "href": "http://localhost:8080/api/users/1" } } }] } }`.

3.2. Crear un endpoint personalizado

Se ha creado un endpoint personalizado para buscar usuarios mediante su apellido.

Repositorio UserRepository

Se añadió el método personalizado en la interfaz del repositorio:

```
1 package com.example.demo.repository;
2
3 import com.example.demo.entity.User;
4 import org.springframework.data.repository.CrudRepository;
5 import org.springframework.stereotype.Repository;
6
7 import java.util.List;
8
9 @Repository
10 public interface UserRepository extends CrudRepository<User, Long> {
11     List<User> findBySurname(String surname);
12 }
```

Controlador UserController

Se creó el controlador REST que expone:

- GET /api/users — Obtener todos los usuarios.
- POST /api/users — Crear un nuevo usuario.
- GET /api/users/search?surname={surname} — Buscar usuario por apellido.


```
1  package com.example.demo.controller;
2
3  import com.example.demo.entity.User;
4  import com.example.demo.repository.UserRepository;
5  import org.springframework.beans.factory.annotation.Autowired;
6  import org.springframework.web.bind.annotation.*;
7
8  import java.util.List;
9
10 @RestController
11 @RequestMapping("/api/users")
12 public class UserController {
13
14     @Autowired
15     private UserRepository userRepository;
16
17     @GetMapping
18     public List<User> getAllUsers() {
19         return (List<User>) userRepository.findAll();
20     }
21
22     @PostMapping
23     public User createUser(@RequestBody User user) {
24         return userRepository.save(user);
25     }
26
27     @GetMapping("/search")
28     public List<User> searchUsersBySurname(@RequestParam String surname) {
29         return userRepository.findBySurname(surname);
30     }
31 }
```

Pruebas de funcionamiento

Vamos a añadir un par de ejemplos de usuarios más al endpoint de users para hacer una búsqueda por apellido:

Estado inicial:

GET

http://localhost:8080/api/users

Send

Query

Headers 2

Auth

Body 1

Tests

Pre Run

Query Parameters

☐

parameter

value

Status: 200 OK

Size: 1.17 KB

Time: 23 ms

Response

Headers 5

Cookies

Results

Docs

```
4 {
5   "name": "Carlos",
6   "surname": "Fernandez",
7   "email": "carlos.fernandez@example.com",
8   "_links": {
9     "self": {
10      "href": "http://localhost:8080/api/users/1"
11    },
12    "user": {
13      "href": "http://localhost:8080/api/users/1"
14    }
15  },
16 },
17 {
18   "name": "Pedro",
19   "surname": "Garcia",
20   "email": "pedro.garcia@example.com",
21   "_links": {
22     "self": {
23      "href": "http://localhost:8080/api/users/2"
24    },
25    "user": {
26      "href": "http://localhost:8080/api/users/2"
27    }
28  },
29 },
30 {
31   "name": "Lucia",
32   "surname": "Gomez",
33   "email": "lucia.gomez@example.com",
34   "_links": {
```

Response

Chart

Endpoint personalizado de busqueda por apellido:

GET

http://localhost:8080/api/users/search/findBySurname?surname=Garcia

Send

Query

Headers 2

Auth

Body 1

Tests

Pre Run

Query Parameters

☒

surname

Garcia

☐

parameter

value

Status: 200 OK

Size: 469 Bytes

Time: 14 ms

Response

Headers 5

Cookies

Results

Docs

```
1 {
2   "_embedded": {
3     "users": [
4       {
5         "name": "Pedro",
6         "surname": "Garcia",
7         "email": "pedro.garcia@example.com",
8         "_links": {
9           "self": {
10            "href": "http://localhost:8080/api/users/2"
11          },
12          "user": {
13            "href": "http://localhost:8080/api/users/2"
14          }
15        }
16      }
17    ]
18  }
19 }
```

Response

Chart