Page<Libro> libroPage=new PageImpl<>(listaRespuesta);

Pasar una lista a Page, para el ResponseEntity.

Consultar por id

@GetMapping("/librosPorBiblioteca/{idBiblioteca}")  
public ResponseEntity<Page<Libro>> librosPorBiblioteca  
 (@Valid @PathVariable Integer idBiblioteca,Pageable pageable){  
 List<Libro> listaRespuesta=new ArrayList<>();  
  
 Optional<Biblioteca> bibliotecaOptional=bibliotecaRepository  
 .findById(idBiblioteca);  
 if (!bibliotecaOptional.isPresent()){  
 return ResponseEntity.*unprocessableEntity*().build();  
 }  
 Page<Libro> libroList = libroRepository.findAll(pageable);  
  
 for (Libro libro:libroList){  
 if(bibliotecaOptional.get().getId()==libro.getBiblioteca().getId()){  
 listaRespuesta.add(libro);  
 }  
  
 }  
 System.*out*.println(listaRespuesta.toString());  
 Page<Libro> libroPage=new PageImpl<>(listaRespuesta);  
 return ResponseEntity.*ok*(libroPage);  
  
}

Consultar por nombre

@GetMapping("/librosPorNombreBiblioteca/{nombreBiblioteca}")  
public ResponseEntity<Page<Libro>> librosPorNombreDEBiblioteca  
 (@Valid @PathVariable String nombreBiblioteca,Pageable pageable){  
 Integer idB=null;  
 List<Libro> listaRespuesta=new ArrayList<>();  
 System.*out*.println(nombreBiblioteca);  
 List<Biblioteca> bibliotecaList=bibliotecaRepository.findAll();  
 System.*out*.println(bibliotecaList.toString());  
 for (Biblioteca biblioteca:bibliotecaList){  
 if (nombreBiblioteca.equals(biblioteca.getNombre())){  
 idB=biblioteca.getId();  
 }  
 }  
 System.*out*.println(idB);  
 if (idB==null){  
 return ResponseEntity.*unprocessableEntity*().build();  
 }  
  
 Page<Libro> libroList = libroRepository.findAll(pageable);  
  
 for (Libro libro:libroList){  
 if(idB==libro.getBiblioteca().getId()){  
 listaRespuesta.add(libro);  
 }  
  
 }  
 System.*out*.println(listaRespuesta.toString());  
 Page<Libro> libroPage=new PageImpl<>(listaRespuesta);  
 return ResponseEntity.*ok*(libroPage);  
  
}

URI url=ServletUriComponentsBuilder.fromCurrentRequest()

.path("/{id}").buildAndExpand(libroGuardado.getId()).toUri();

Otra forma de poner el URI:

//otra forma de poner el URI  
return ResponseEntity.*created*(new URI("/api/maker/save")).build();

Lamda para sacar un optional de una lista

equipo.stream().filter(equipo-> equipo.getNombrePais().equalsIgnoreCase(nombre)).findFirst();

de esto se obtiene un Optional<Equipo> optionalEquipo

Como trabajo con BigDecimal

El **método java.math.BigDecimal.compareTo(BigDecimal bg)** comprueba la igualdad de *este* objeto BigDecimal y BigDecimal **bg** pasado como parámetro. El método considera dos objetos BigDecimal iguales incluso si tienen el mismo valor independientemente de la escala.

**Sintaxis:**

public int compareTo(*BigDecimal bg*)

**Parámetros:** esta función acepta solo un *objeto* BigDecimal de tipo BigDecimal para compararlo con *este* objeto BigDecimal.

**Valor de retorno:** este método puede devolver los siguientes valores:

* **0** : si el valor de *este* BigDecimal es *igual* al del objeto BigDecimal pasado como parámetro.
* **1** : si el valor de *este* BigDecimal es *mayor que* el del objeto BigDecimal pasado como parámetro.
* **-1** : si el valor de *este* BigDecimal es *menor que* el del objeto BigDecimal pasado como parámetro.

Ejemplo

// Java program to demonstrate compareTo() method

import java.io.\*;

import java.math.\*;

public class GFG {

    public static void main(String[] args)

    {

        // Creating 2 BigDecimal objects

        BigDecimal b1, b2;

        b1 = new BigDecimal(67891);

        b2 = new BigDecimal(12346);

        if (b1.compareTo(b2) == 0) {

            System.out.println(b1 + " and " + b2 + " are equal.");

        }

        else if (b1.compareTo(b2) == 1) {

            System.out.println(b1 + " is greater than " + b2 + ".");

        }

        else {

            System.out.println(b1 + " is lesser than " + b2 + ".");

        }

    }

}