

## BACKEND – SPRINGBOOT

Comenzaremos creando proyecto en SpringBoot para diseñar la gestión de la base de datos.

Versión de spring: 3.5.6

JDK: 21 o superior

Agregar dependencias:

- Spring web
- Spring data jpa
- Spring devtools
- MySql Driver
- Lombok

Crear esquema de base de datos en WorkBench: [base](#)

Configurar conexión en application.properties

### # parámetros de la conexión

```
spring.datasource.url=jdbc:mysql://localhost:3306/base?serverTimezone=UTC&useSSL=false
```

```
spring.datasource.username=root
```

```
spring.datasource.password=sasa
```

### # Configuración de JPA

```
spring.jpa.hibernate.ddl-auto=update
```

```
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
```

### # Configuración de logs (opcional)

```
spring.jpa.show-sql=true
```

```
spring.jpa.properties.hibernate.format_sql=true
```

Configuramos la estructura del proyecto a través de los packages: entities, repository, services, restcontrollers

## ENTIDADES:

Comenzamos por la creación de las clases Categoría, Producto:

```

J ProductoRepositories.java J Producto.java J Categoría.java X
src > main > java > com > vivitasol > projectbackend > entities > J Categoría.java > {} com.vivitasol.projectbackend.entities
1 package com.vivitasol.projectbackend.entities;
2 import jakarta.persistence.Entity;
3 import jakarta.persistence.GeneratedValue;
4 import jakarta.persistence.GenerationType;
5 import lombok.AllArgsConstructor;
6 import lombok.Data;
7 import lombok.NoArgsConstructor;
8 import jakarta.persistence.Id;
9 import jakarta.persistence.OneToMany;
10 import java.util.List;
11
12 @Data
13 @NoArgsConstructor
14 @AllArgsConstructor
15
16 @Entity
17 public class Categoría {
18
19     @Id
20     @GeneratedValue(strategy = GenerationType.IDENTITY)
21     private Long Id;
22     private String nombre;
23
24     @OneToMany(mappedBy = "categoria")
25     private List<Producto> productos;
26
27     @Override
28     public String toString() {
29         return "Categoría [Id=" + Id + ", nombre=" + nombre + ", productos=" + productos + "];"
30     }
31 }
32
33

```

## Clase Producto:

```

ProductoRepositories.java  Producto.java X  Categoria.java
src > main > java > com > vivitasol > projectbackend > entities > J Producto.java > {} com.vivita
1 package com.vivitasol.projectbackend.entities;
2
3 import jakarta.persistence.Entity;
4 import jakarta.persistence.GeneratedValue;
5 import jakarta.persistence.GenerationType;
6 import jakarta.persistence.Id;
7 import jakarta.persistence.JoinColumn;
8 import jakarta.persistence.ManyToOne;
9 import lombok.AllArgsConstructor;
10 import lombok.Data;
11 import lombok.NoArgsConstructor;
12
13 @Data
14 @NoArgsConstructor
15 @AllArgsConstructor
16 @Entity
17 public class Producto {
18
19     @Id
20     @GeneratedValue(strategy=GenerationType.IDENTITY)
21     private Long id;
22     private String nombre;
23     private String descripcion;
24     private Long precio;
25
26     @ManyToOne
27     @JoinColumn(name="categoria.id")
28     private Categoria categoria;
29
30 }
31

```

REPOSITORIOS:

```

ProductoRepositories.java X
src > main > java > com > vivitasol > projectbackend > repositories > J ProductoRepositories.java > ...
1 package com.vivitasol.projectbackend.repositories;
2
3 import org.springframework.data.repository.CrudRepository;
4
5 import com.vivitasol.projectbackend.entities.Producto;
6
7 public interface ProductoRepositories extends CrudRepository <Producto, Long>{
8
9 }
10

```

```

ProductoRepositories.java  Producto.java  CategoriaRepositories.java X  Categoria.java
src > main > java > com > vivitasol > projectbackend > repositories > J CategoriaRepositories.java > ...
1 package com.vivitasol.projectbackend.repositories;
2
3 import org.springframework.data.repository.CrudRepository;
4
5 import com.vivitasol.projectbackend.entities.Categoria;
6
7 public interface CategoriaRepositories extends CrudRepository<Categoria, Long> {
8
9 }
10

```

## SERVICIOS

Comenzamos con el servicio de categoría:  
implementamos interface y clase;  
**Interface CategoriaServices**

```
Producto.java  J CategoriaRepositories.java  J CategoriaServices.java X  J Ca
main > java > com > vivitasol > projectbackend > services > J CategoriaServices.java > ...
package com.vivitasol.projectbackend.services;

import com.vivitasol.projectbackend.entities.Categoria;
import java.util.List;

public interface CategoriaServices {

    Categoria crear(Categoria categoria);
    Categoria obtenerId(Long id);
    List<Categoria> listarTodas();
    void eliminar(Long id);
    Categoria actualizar(Long id, Categoria categoriaActualizada);
}
```

**Clase CategoriaServicesImpl:**

```
J CategoriaServicesImpl.java X
src > main > java > com > vivitasol > projectbackend > services > J CategoriaServicesImpl.java > Language Support for Java
10
11 @Service
12 public class CategoriaServicesImpl implements CategoriaServices {
13
14     @Autowired
15     private CategoriaRepositories categoriaRepositories;
16
17     @Override
18     public Categoria crear(Categoria categoria) {
19         return categoriaRepositories.save(categoria);
20     }
21
22     @Override
23     public Categoria obtenerId(Long id) {
24         return categoriaRepositories.findById(id)
25             .orElseThrow(() -> new RuntimeException("Categoría no encontrada"));
26     }
27
28     @Override
29     public List<Categoria> listarTodas() {
30         return (List<Categoria>) categoriaRepositories.findAll();
31     }
32
33 }
```

```
J CategoriaServicesImpl.java X
src > main > java > com > vivitasol > projectbackend > services > J CategoriaServicesImpl.java > Language Sup
12 public class CategoriaServicesImpl implements CategoriaServices {
13
14     @Override
15     public List<Categoria> listarTodas() {
16         return (List<Categoria>) categoriaRepositories.findAll();
17     }
18
19     @Override
20     public void eliminar(Long id) {
21         if (!categoriaRepositories.existsById(id)) {
22             throw new RuntimeException("Categoría no encontrada");
23         }
24         categoriaRepositories.deleteById(id);
25     }
26
27     @Override
28     public Categoria actualizar(Long id, Categoria categoriaActualizada) {
29         Categoria existente = obtenerId(id);
30         existente.setNombre(categoriaActualizada.getNombre());
31         return categoriaRepositories.save(existente);
32     }
33
34 }
```

Nota: implementar servicio para Productos

## RESTCONTROLLERS

```
4
5 @RestController
6 @RequestMapping("/api/categorias")
7 public class CategoriaRestControllers {
8
9     @Autowired
10     private CategoriaServices categoriaServices;
11
12     @PostMapping
13     public ResponseEntity<Categoria> crearCategoria(@RequestBody Categoria categoria) {
14         Categoria nuevaCategoria = categoriaServices.crear(categoria);
15         return ResponseEntity.ok(nuevaCategoria);
16     }
17
18     @GetMapping("/{id}")
19     public ResponseEntity<Categoria> obtenerCategoriaPorId(@PathVariable Long id) {
20         Categoria categoria = categoriaServices.obtenerId(id);
21         return ResponseEntity.ok(categoria);
22     }
23
24     @GetMapping
25     public ResponseEntity<List<Categoria>> listarCategorias() {
26         List<Categoria> categorias = categoriaServices.listarTodas();
27         return ResponseEntity.ok(categorias);
28     }
29
30 }
```

```
@GetMapping
public ResponseEntity<List<Categoria>> listarCategorias() {
    List<Categoria> categorias = categoriaServices.listarTodas();
    return ResponseEntity.ok(categorias);
}

@DeleteMapping("/{id}")
public ResponseEntity<Void> eliminarCategoria(@PathVariable Long id) {
    categoriaServices.eliminar(id);
    return ResponseEntity.noContent().build();
}

@PutMapping("/{id}")
public ResponseEntity<Categoria> actualizarCategoria(@PathVariable Long id, @RequestBody Categoria categoriaActualizada) {
    Categoria categoria = categoriaServices.actualizar(id, categoriaActualizada);
    return ResponseEntity.ok(categoria);
}
}
```

Implementar servicios y controller para Productos