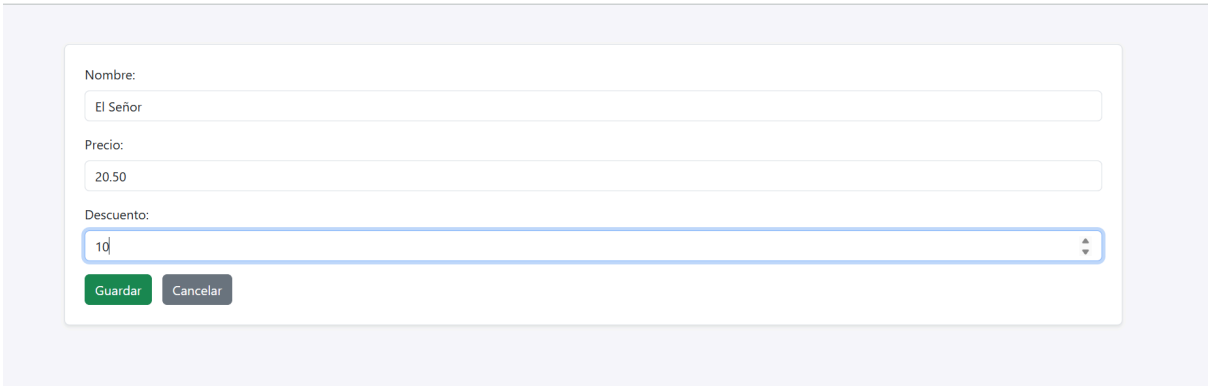
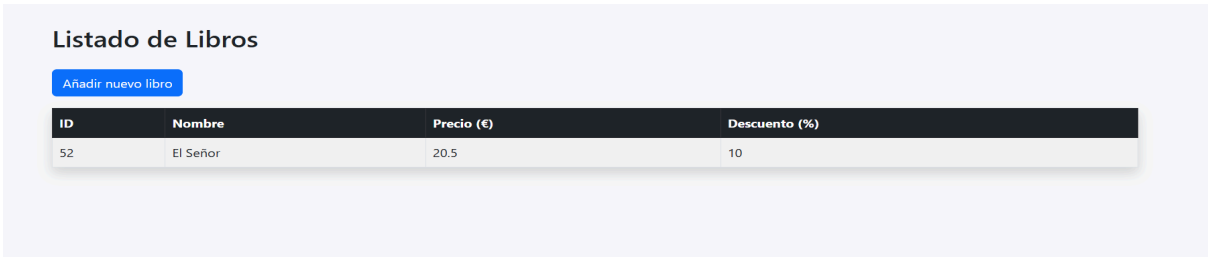
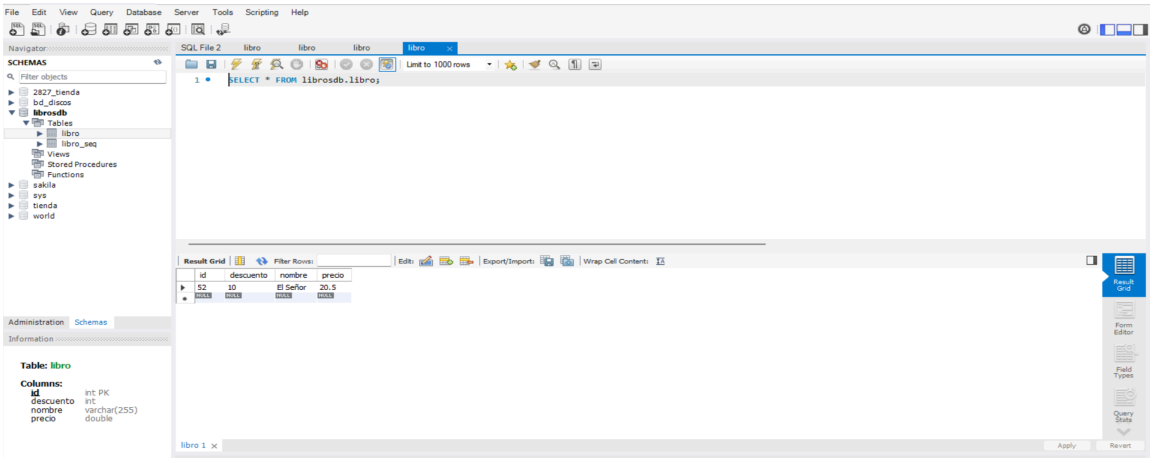
Listado con 0 Libros

formulario nuevo libro

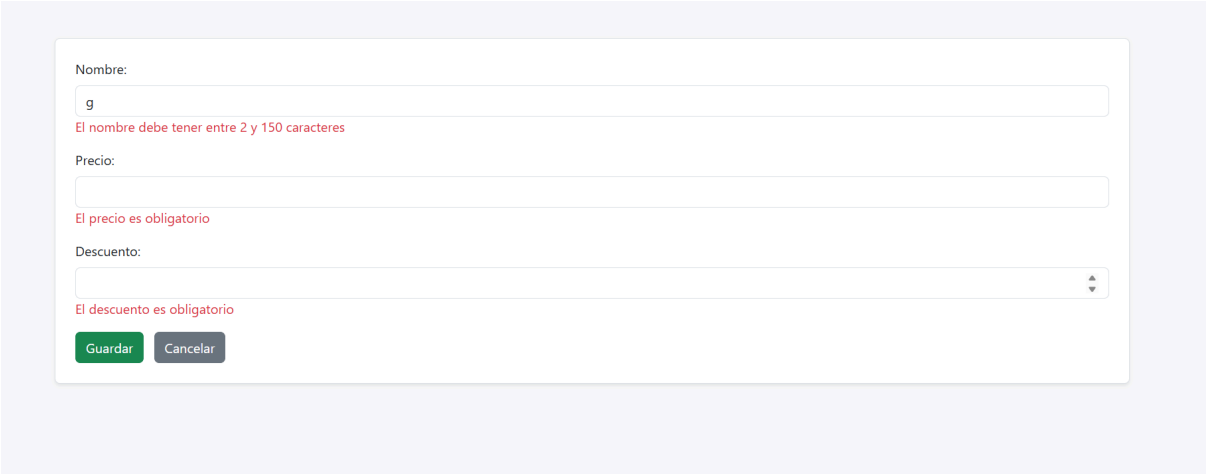


libro agregado, web, Bdd terminal.





```
Hibernate: select next_val as id_val from libro_seq for update
Hibernate: update libro_seq set next_val= ? where next_val=?
Hibernate: insert into libro (descuento,nombre,precio,id) values (?,?,?,?)
Hibernate: select l1_0.id,l1_0.descuento,l1_0.nombre,l1_0.precio from libro l1_0
```



LibroController.java

```java
package com.ejemplo.libros.controller;

import com.ejemplo.libros.model.Libro;
import com.ejemplo.libros.repository.LibroDAO;
import jakarta.validation.Valid;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;

@Controller
@RequestMapping("/libros")
public class LibroController {

    private final LibroDAO libroDAO;

    // ✅ INYECCIÓN POR CONSTRUCTOR
    public LibroController(LibroDAO libroDAO) {
        this.libroDAO = libroDAO;
    }

    @GetMapping
    public String mostrarLibros(Model model) {
```

```
        model.addAttribute("libros", libroDAO.findAll());
        return "libros";
    }

    @GetMapping("/nuevo")
    public String mostrarFormulario(Model model) {
        model.addAttribute("libro", new Libro());
        return "formulario";
    }

    @PostMapping("/guardar")
    public String guardarLibro(@Valid @ModelAttribute Libro libro,
BindingResult result) {
        if (result.hasErrors()) {
            return "formulario";
        }
        libroDAO.save(libro);
        return "redirect:/libros";
    }
}
```

Libro.java

```
package com.ejemplo.libros.model;

import jakarta.persistence.*;
import jakarta.validation.constraints.*;

@Entity
public class Libro {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @NotBlank(message = "El nombre es obligatorio")
    @Size(min = 2, max = 150, message = "El nombre debe tener entre 2 y
150 caracteres")
    private String nombre;
```

```
    @NotNull(message = "El precio es obligatorio")
    @DecimalMin(value = "0.01", inclusive = true, message = "El precio
debe ser mayor que 0")
    @Digits(integer = 10, fraction = 2, message = "El precio debe tener
como máximo dos decimales")
    private BigDecimal precio;

    @NotNull(message = "El descuento es obligatorio")
    @Min(value = 0, message = "El descuento debe ser mínimo 0")
    @Max(value = 100, message = "El descuento no puede superar 100")
    private Integer descuento;

    // --- Getters y Setters ---

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public Double getPrecio() {
        return precio;
    }

    public void setPrecio(Double precio) {
        this.precio = precio;
    }

    public Integer getDescuento() {
        return descuento;
    }

    public void setDescuento(Integer descuento) {
```

```
        this.descuento = descuento;
    }
}
```

LibroApp.java

```
package com.ejemplo.libros;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class LibroApp {
    public static void main(String[] args) {
        SpringApplication.run(LibroApp.class, args);
    }
}
```

Formulario.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org" lang="es">
<head>
  <meta charset="UTF-8">
  <title>Nuevo Libro</title>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.m
in.css" rel="stylesheet">
</head>
<body class="container py-5 bg-light">

  <form th:action="@{/libros/guardar}" th:object="${libro}"
method="post" class="border p-4 rounded shadow-sm bg-white">

    <div class="mb-3">
```

```
      <label for="nombre" class="form-label">Nombre:</label>
      <input id="nombre" type="text" th:field="*{nombre}"
class="form-control">
      <div class="text-danger" th:if="${#fields.hasErrors('nombre')}"
th:errors="*{nombre}"></div>
    </div>

    <div class="mb-3">
      <label for="precio" class="form-label">Precio:</label>
      <input id="precio" type="number" step="0.01" th:field="*{precio}"
class="form-control">
      <div class="text-danger" th:if="${#fields.hasErrors('precio')}"
th:errors="*{precio}"></div>
    </div>

    <div class="mb-3">
      <label for="descuento" class="form-label">Descuento:</label>
      <input id="descuento" type="number" th:field="*{descuento}"
class="form-control">
      <div class="text-danger"
th:if="${#fields.hasErrors('descuento')}"
th:errors="*{descuento}"></div>
    </div>

    <button type="submit" class="btn btn-success">Guardar</button>
    <a href="/libros" class="btn btn-secondary ms-2">Cancelar</a>

  </form>

</body>
</html>
```

Libros.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org" lang="es">
<head>
    <meta charset="UTF-8">
    <title>Listado de Libros</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.m
in.css" rel="stylesheet">
```

```
</head>
<body class="bg-light">
<div class="container mt-5">
    <h2 class="mb-4">Listado de Libros</h2>

    <a href="/libros/nuevo" class="btn btn-primary mb-3">Añadir nuevo
libro</a>

    <table class="table table-striped table-bordered shadow">
        <thead class="table-dark">
            <tr>
                <th scope="col">ID</th>
                <th scope="col">Nombre</th>
                <th scope="col">Precio (€)</th>
                <th scope="col">Descuento (%)</th>
            </tr>
        </thead>
        <tbody>
            <tr th:each="libro : ${libros}">
                <td th:text="${libro.id}">1</td>
                <td th:text="${libro.nombre}">Nombre</td>
                <td th:text="${libro.precio}">0.0</td>
                <td th:text="${libro.descuento}">0%</td>
            </tr>
        </tbody>
    </table>
</div>
</body>
</html>
```

LibroDao.class

```
// Source code is decompiled from a .class file using FernFlower
decompiler.
package com.ejemplo.libros.repository;

import com.ejemplo.libros.model.Libro;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface LibroDAO extends JpaRepository<Libro, Integer> {
```

```
}
```

application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/librosdb?useSSL=false
&serverTimezone=UTC
spring.datasource.username=itxine
spring.datasource.password=libros_pass

spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect
spring.jpa.show-sql=true
```

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.ejemplo</groupId>
    <artifactId>libro-app-mysql</artifactId>
    <version>1.0.0</version>
    <packaging>jar</packaging>
    <name>Gestor de Libros</name>
    <description>Aplicación Spring Boot para gestionar
libros</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.1.5</version>
        <relativePath/>
    </parent>

    <properties>
        <java.version>21</java.version>
    </properties>

    <dependencies>
        <!-- Web MVC + REST -->
        <dependency>
```

```
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <!-- Plantillas Thymeleaf -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>

        <!-- Validaciones con @Valid -->
        <dependency>
            <groupId>jakarta.validation</groupId>
            <artifactId>jakarta.validation-api</artifactId>
        </dependency>

        <!-- JPA + Hibernate -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>

        <!-- Conector MySQL -->
        <dependency>
            <groupId>com.mysql</groupId>
            <artifactId>mysql-connector-j</artifactId>
            <scope>runtime</scope>
        </dependency>

        <!-- H2 solo si quieres hacer pruebas en memoria -->
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <scope>runtime</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
```

```
        </plugins>
    </build>
</project>
```