Contenido

Desarrollar una aplicación web completa usando Spring Boot que permita realizar operaciones CRUD de productos visto en las clases

Application.properties

index.html | PruebaApplication.java | application.properties | Producto.java

```
# MySQL Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/productos_db?useSSL=false&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=root

# JPA
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

# Thymeleaf
spring.thymeleaf.prefix=classpath:/templates/
spring.thymeleaf.suffix=.html

# Server
server.port=8080
```

Ilustración 1CODIGO DE PROPIEDADES

ProductoController

```
@Controller
public class ProductoController {

    @Autowired
    private ProductoService productoService;

    @GetMapping("/")
    public String index(Model model) {
        var productos = productoService.listarProductos();
        model.addAttribute( attributeName: "productos", productos);
        return "index";
    }

    @PostMapping("/guardar")
    public String guardar(Producto producto) {
        productoService.guardar(producto);
        return "redirect:/";
    }

    @GetMapping("/eliminar/{id}")
    public String eliminar(@PathVariable Long id) {
        Producto producto = new Producto();
        producto.setId(id);
        productoService.eliminar(producto);
        return "redirect:/";
    }
```

Ilustración 2CODIFO PRRODUCTO CONTROLADOR

Producto(modelo)

```java
package com.example.Prueba.models;

import jakarta.persistence.*;

@Entity
@Table(name = "productos")
public class Producto {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "nombre", nullable = false, length = 100)  4 usages
    private String nombre;

    @Column(name = "precio", nullable = false)  4 usages
    private Double precio;

    // Constructor vacío (OBLIGATORIO para JPA)
    public Producto() {
    }

    // Constructor con parámetros
    public Producto(String nombre, Double precio) {  no usages
        this.nombre = nombre;
        this.precio = precio;
    }
```

Ilustración 3CODIGO DE PRODUCTO MODELO

ProductoDao

```java
package com.example.Prueba.dao;
import com.example.Prueba.models.Producto;
import org.springframework.data.repository.CrudRepository;
import java.util.List;

public interface ProductoDao extends CrudRepository<Producto, Long> {  2 usages
    List<Producto> findAll();
}
```

Ilustración 4PRODUCTO DAO

ProductoService(interfaz)

```java
package com.example.Prueba.services;

import com.example.Prueba.models.Producto;
import java.util.List;

public interface ProductoService {  3 usages  1 implementation
    List<Producto> listarProductos();  1 usage  1 implementation
    void guardar(Producto producto);  1 usage  1 implementation
    void eliminar(Producto producto);  1 usage  1 implementation
    Producto encontrarProducto(Producto producto);  no usages  1 implementation
}
```

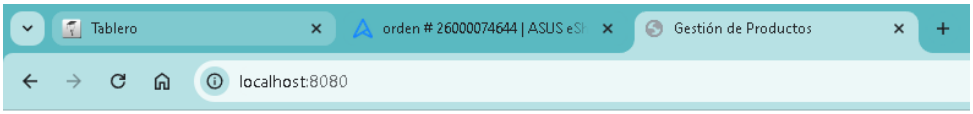
Ilustración 5INTERFAZ PRODUCTO

ProductoServiceImpl

```java
package com.example.Prueba.services;

import com.example.Prueba.dao.ProductoDao;
import com.example.Prueba.models.Producto;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import java.util.List;

@Service
public class ProductoServiceImpl implements ProductoService {

    @Autowired
    private ProductoDao productoDao;

    @Override  1 usage
    @Transactional(readOnly = true)
    public List<Producto> listarProductos() {
        return productoDao.findAll();
    }

    @Override  1 usage
    @Transactional
    public void guardar(Producto producto) {
        productoDao.save(producto);
    }
```

Ilustración 6 PRODUCTOSERVCELMPL

Index (vista)



Gestión de Productos

Agregar Producto

Nombre | Precio | Guardar

Lista de Productos

ID	Nombre	Precio	Acciones
1	coca	23.0	Eliminar
2	chocolate	34.0	Eliminar

Ilustración 7VISTA

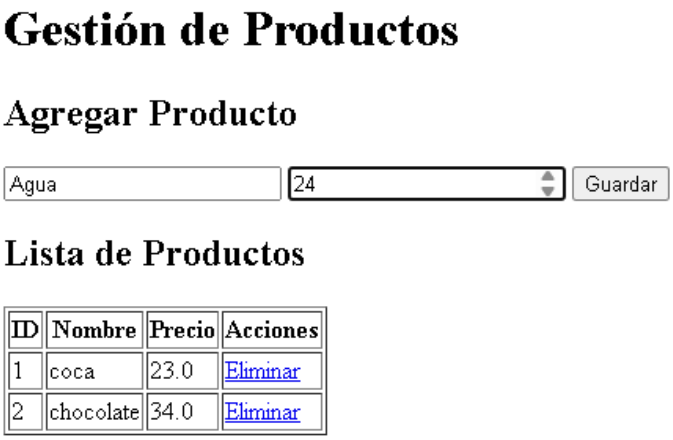


Ilustración 8agregar productos

Conclusión

El sistema funciona mediante una arquitectura que interactúa de manera coordinada:
el Controller recibe las peticiones HTTP, el Service ejecuta la lógica de negocio,
el DAO gestiona la persistencia, y la Vista renderiza la interfaz.