Ejercicio Carnavales

Proyecto: Gestión de Productos con Servlets y JSP

Objetivo:

Desarrollar una aplicación web en Java utilizando Servlets, JSP y MySQL para gestionar un inventario de productos.

La aplicación debe permitir:
✅ Registrar nuevos productos.
✅ Listar todos los productos en una tabla.
✅ Editar la información de un producto.
✅ Eliminar productos con confirmación previa.
✅ Proteger rutas mediante filtros y sesiones.

Requisitos Técnicos:

1. Estructura del Proyecto

Debe seguir el patrón **MVC (Modelo-Vista-Controlador)** con los siguientes paquetes:

- **`com.empresa.controller`** → Contendrá los Servlets (`LoginController`, `ProductoController`).
- **`com.empresa.service`** → Manejará la lógica de negocio (`ProductoService`).
- **`com.empresa.dao`** → Se encargará de la conexión a la base de datos (`ProductoDAO`).
- **`com.empresa.model`** → Definirá la clase `Producto`.
- **`com.empresa.filter`** → Implementará un filtro para restringir acceso a páginas privadas (`AuthFilter`).

---

2. Funcionalidades a Implementar

2.1. Registro de Productos (`altaProducto.jsp`)

Crear un formulario donde los usuarios puedan ingresar:

- Nombre del producto
- Descripción
- Precio
- Cantidad en stock

Los datos deben guardarse en la base de datos MySQL.

2.2. Inicio de Sesión (`login.jsp`)

El usuario ingresará su nombre de usuario y contraseña. Si las credenciales son correctas, se iniciará una sesión y se redirigirá a `index.jsp`. En caso contrario, se mostrará un mensaje de error.

2.3. Listado de Productos (`productos.jsp`)

Mostrar en una tabla todos los productos de la base de datos con las siguientes acciones:
✅ **Editar** → Redirige a un formulario con los valores rellenos (`editarProducto.jsp`).
❌ **Eliminar** → Muestra un mensaje de confirmación antes de borrar.

2.4. Edición de Productos (`editarProducto.jsp`)

Permitir modificar los datos de un producto y actualizar la base de datos.

2.5. Eliminación con Confirmación (`productos.jsp`)

Al hacer clic en "Eliminar", debe mostrarse un **modal de confirmación** antes de borrar el producto de la base de datos.

2.6. Protección de Páginas Privadas (`AuthFilter.java`)

Crear un **filtro** que impida el acceso a `index.jsp` y `productos.jsp` si el usuario no ha iniciado sesión.

---

3. Base de Datos (`inventario` en MySQL)

Crear la tabla `producto` con la siguiente estructura:

```
CREATE DATABASE inventario;
USE inventario;

CREATE TABLE producto (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100),
    descripcion TEXT,
    precio DECIMAL(10,2),
    stock INT
);
```

---

4. Funcionalidades extra

4.1. Agregar Roles de Usuario (Admin y Usuario)

📌 Descripción:

Modificar la base de datos para incluir una tabla usuarios con un campo rol que puede ser "admin" o "usuario".

Los administradores pueden ver, agregar, editar y eliminar productos.

Los usuarios normales solo pueden ver la lista de productos.

📌 Modificaciones necesarias:

Agregar un UsuarioDAO, UsuarioService y modificar LoginController.

Adaptar el AuthFilter para verificar los roles y restringir acciones según corresponda.

4.2. Paginación en la Lista de Productos

📌 Descripción:

Si hay muchos productos en la base de datos, la lista puede volverse demasiado larga.

Implementar un sistema de paginación que muestre solo 10 productos por página y permita navegar entre páginas.

📌 Modificaciones necesarias:

Cambiar la consulta SQL en ProductoDAO para incluir LIMIT y OFFSET.

Agregar botones "Siguiente" y "Anterior" en productos.jsp.

4.3 Búsqueda y Filtros en la Lista de Productos

📌 Descripción:

Permitir que los usuarios busquen productos por nombre o filtren por precio o stock.

Implementar un formulario en productos.jsp con un campo de búsqueda y filtros de rango de precio.

📌 Modificaciones necesarias:

Modificar ProductoDAO para permitir la búsqueda por nombre con LIKE y filtrado con BETWEEN.

Agregar lógica en ProductoController para recibir parámetros de búsqueda y filtro.

4.4. Subida de Imágenes para los Productos

📌 Descripción:

Permitir que cada producto tenga una imagen asociada.

Guardar las imágenes en una carpeta del servidor y almacenar la ruta en la base de datos.

📌 Modificaciones necesarias:

Agregar un campo imagen en la base de datos.

Modificar altaProducto.jsp y editarProducto.jsp para permitir la subida de archivos.

Adaptar ProductoDAO para almacenar y recuperar la ruta de la imagen.

4.5. Implementar un Carrito de Compras

📌 Descripción:

Permitir que los usuarios agreguen productos a un carrito de compras.

Mostrar los productos seleccionados y calcular el total.

📌 Modificaciones necesarias:

Crear una sesión para almacenar el carrito.

Agregar botones "Añadir al carrito" en productos.jsp.

Crear una nueva página carrito.jsp para mostrar los productos seleccionados.

5. Entrega y Evaluación

📌 **Requisitos para la entrega:**

- Código organizado en la estructura de paquetes definida.
- Uso correcto de sesiones y filtros.
- Implementación del CRUD (Create, Read, Update, Delete).
- Funcionamiento completo de la aplicación con MySQL.