Desarrollo de Aplicaciones Web

LABORATORIO Nº 6

Patrón DAO + MVC

Alumno(s):	*Salas Arenas Marcelo Aldahir*	***Nota***	
Grupo:	*b*	***Ciclo: IV***	

Criterio de Evaluación	*Excelente (4pts)*	*Bueno (3pts)*	*Requiere mejora (2pts)*	*No acept. (0pts)*	*Puntaje Logrado*
Entiende los conceptos del patrón MVC.					
Entiende los conceptos del patrón DAO.					
Logra programar los ejercicios propuestos.					
Sabe conectar la importancia de la IA.					
Es puntual y redacta el informe adecuadamente.					

TEMA: Patrón MVC - DAO usando la tecnología de Java EE

OBJETIVOS

- Comprender el patrón MVC + DAO.

REQUERIMIENTOS

- **Servidor Tomcat instalado.**
- **Base de datos - SQLite.**

PROCEDIMIENTO

- ***El laboratorio se ha diseñado para ser desarrollado en grupos de 2.***
- ***Solo un integrante sube el documento al Canvas.***

MARCO TEÓRICO

Ventajas de combinar DAO y MVC

- **Separación de responsabilidades**:
  - **DAO** encapsula toda la lógica de acceso a datos (consultas SQL, conexión a SQLite), aislando la base de datos del resto de la aplicación.
  - **MVC** organiza la lógica de negocio (`Producto`), la presentación (`productos.jsp`) y la gestión de peticiones (`ProductoServlet`).
- **Mantenibilidad**:
  - Cambiar la base de datos (por ejemplo, de SQLite a PostgreSQL) solo requiere modificar `ProductoDAOImpl` y `DatabaseConfig`.
  - Modificar la interfaz de usuario solo afecta a `productos.jsp`.
- **Reusabilidad**:
  - El DAO puede usarse en otras aplicaciones o módulos.
  - La estructura MVC es escalable para agregar más funcionalidades (editar productos, buscar, etc.).

1. **Crear un proyecto en Java Web que utilice el patrón MVC + DAO.**

Cómo funciona

- **Modelo**:
  - `Producto`: Representa la entidad con atributos `id`, `nombre` y `precio`.
- **DAO**:
  - `ProductoDAOImpl`: Maneja las operaciones CRUD (listar, agregar, eliminar) en SQLite. Crea la tabla `productos` si no existe.
- **Controlador**:
  - `ProductoServlet`: Procesa las solicitudes HTTP:
    - **GET**: Obtiene la lista de productos y la envía a la vista.
    - **POST**: Maneja las acciones de agregar o eliminar productos.
- **Vista**:
  - `productos.jsp`: Muestra un formulario para agregar productos y una tabla con la lista de productos, incluyendo botones para eliminar.
- **SQLite**:
  - La base de datos se almacena en el archivo `tienda.db` en la raíz del proyecto. SQLite es ligero y no requiere configuración de servidor.

1 .1 Dependencias Maven (pom.xml)

```
    <dependencies>
    <!-- Servlet API -->
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>4.0.1</version>
        <scope>provided</scope>
    </dependency>
    <!-- SQLite JDBC Driver -->
    <dependency>
        <groupId>org.xerial</groupId>
        <artifactId>sqlite-jdbc</artifactId>
        <version>3.46.0</version>
    </dependency>
    <!-- JSTL para JSP -->
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>jstl</artifactId>
        <version>1.2</version>
    </dependency>
</dependencies>
```

1.2 Modelo: Entidad

- model/Producto.java

```
package model;

public class Producto {
    private int id;
    private String nombre;
    private double precio;

    // Constructor
    public Producto() {}
    public Producto(int id, String nombre, double precio) {
        this.id = id;
        this.nombre = nombre;
        this.precio = precio;
    }

    // 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; }
}
```

1.3 DAO: Acceso a datos

- dao/ProductoDAO.java (Interfaz)

```
package dao;

import model.Producto;
import java.util.List;

public interface ProductoDAO {
    List<Producto> listarTodos();
    void agregar(Producto producto);
    void eliminar(int id);
}
```

- dao/ProductoDAOImpl.java (Implementación con SQLite)

```
package dao;

import model.Producto;
import config.DatabaseConfig;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class ProductoDAOImpl implements ProductoDAO {
    private Connection getConnection() throws SQLException {
```

```
        return DatabaseConfig.getConnection();
    }

    // Crear la tabla si no existe
    private void initDatabase() {
        String sql = "CREATE TABLE IF NOT EXISTS productos (" +
                     "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
                     "nombre TEXT NOT NULL, " +
                     "precio REAL NOT NULL)";
        try (Connection conn = getConnection();
             Statement stmt = conn.createStatement()) {
            stmt.execute(sql);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public ProductoDAOImpl() {
        initDatabase(); // Inicializar la tabla al crear el DAO
    }

    @Override
    public List<Producto> listarTodos() {
        List<Producto> productos = new ArrayList<>();
        String sql = "SELECT * FROM productos";
        try (Connection conn = getConnection();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                productos.add(new Producto(
                    rs.getInt("id"),
                    rs.getString("nombre"),
                    rs.getDouble("precio")
                ));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return productos;
    }

    @Override
    public void agregar(Producto producto) {
        String sql = "INSERT INTO productos (nombre, precio) VALUES (?, ?)";
        try (Connection conn = getConnection();
             PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setString(1, producto.getNombre());
            stmt.setDouble(2, producto.getPrecio());
            stmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void eliminar(int id) {
        String sql = "DELETE FROM productos WHERE id = ?";
        try (Connection conn = getConnection();
             PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setInt(1, id);
            stmt.executeUpdate();
```

```
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

1.4 Configuración de SQLite

- config/DatabaseConfig.java

```
package config;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConfig {
    private static final String URL = "jdbc:sqlite:tienda.db"; //poner toda la ruta

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL);
    }
}
```

1.5 Controlador: Servlet MVC

- controller/ProductoServlet.java

```
package controller;

import dao.ProductoDAO;
import dao.ProductoDAOImpl;
import model.Producto;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.List;

@WebServlet("/productos")
public class ProductoServlet extends HttpServlet {
    private ProductoDAO productoDAO;

    @Override
    public void init() throws ServletException {
        productoDAO = new ProductoDAOImpl();
    }

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        List<Producto> productos = productoDAO.listarTodos();
        req.setAttribute("productos", productos);
```

```
        req.getRequestDispatcher("productos.jsp").forward(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        String accion = req.getParameter("accion");

        if ("agregar".equals(accion)) {
            String nombre = req.getParameter("nombre");
            double precio = Double.parseDouble(req.getParameter("precio"));
            Producto producto = new Producto(0, nombre, precio);
            productoDAO.agregar(producto);
        } else if ("eliminar".equals(accion)) {
            int id = Integer.parseInt(req.getParameter("id"));
            productoDAO.eliminar(id);
        }

        resp.sendRedirect("productos");
    }
}
```

1.6 Vista: JSP

- productos.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
    <title>Lista de Productos</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 20px; }
        table { border-collapse: collapse; width: 100%; }
        th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }
        th { background-color: #f2f2f2; }
    </style>
</head>
<body>
    <h2>Gestión de Productos</h2>
    <form action="productos" method="post">
        <input type="hidden" name="accion" value="agregar">
        <label>Nombre:</label>
        <input type="text" name="nombre" required>
        <label>Precio:</label>
        <input type="number" step="0.01" name="precio" required>
        <button type="submit">Agregar Producto</button>
    </form>

    <h3>Lista de Productos</h3>
    <table>
        <tr>
            <th>ID</th>
            <th>Nombre</th>
            <th>Precio</th>
            <th>Acción</th>
        </tr>
        <c:forEach var="producto" items="${productos}">
```

```
            <tr>
                <td>${producto.id}</td>
                <td>${producto.nombre}</td>
                <td>${producto.precio}</td>
                <td>
                    <form action="productos" method="post">
                        <input type="hidden" name="accion" value="eliminar">
                        <input type="hidden" name="id" value="${producto.id}">
                        <button type="submit">Eliminar</button>
                    </form>
                </td>
            </tr>
        </c:forEach>
    </table>
</body>
</html>
```

2. **Realiza un MVC + DAO de un objeto en el mismo proyecto.**


```
package config;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.ResultSet;

public class DatabaseConfig {
    private static final String URL =
"jdbc:sqlite:C:/Users/GALAX/Documents/NetBeansProjects/Lab06/tienda.db";

    public static Connection getConnection() throws SQLException {
        try {
            // Intentar cargar el driver SQLite
            Class.forName("org.sqlite.JDBC");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
            throw new SQLException("SQLite JDBC driver no encontrado.", e);
        }
        return DriverManager.getConnection(URL);
    }

    public static void testConnection() {
        try (Connection conn = getConnection()) {
            if (conn != null) {
                System.out.println("Conexión a la base de datos establecida.");

                // Realizar una consulta simple para verificar la conexión
                Statement stmt = conn.createStatement();
                String query = "SELECT name FROM sqlite_master WHERE type='table';";
                ResultSet rs = stmt.executeQuery(query);
```

```
            while (rs.next()) {
                System.out.println("Tabla: " + rs.getString("name"));
            }
        }
    } catch (SQLException e) {
        System.out.println("Error de conexión a la base de datos: " + e.getMessage());
    }
}

public static void main(String[] args) {
    testConnection();
}
}


package controller;

import dao.ProductoDAO;
import dao.ProductoDAOImpl;
import model.Producto;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.List;

@WebServlet("/productos")
public class ProductoServlet extends HttpServlet {
    private ProductoDAO productoDAO;

    @Override
    public void init() throws ServletException {
        productoDAO = new ProductoDAOImpl();
    }

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        List<Producto> productos = productoDAO.listarTodos();
        req.setAttribute("productos", productos);
        req.getRequestDispatcher("productos.jsp").forward(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        String accion = req.getParameter("accion");

        if ("agregar".equals(accion)) {
```

```
            String nombre = req.getParameter("nombre");
            double precio = Double.parseDouble(req.getParameter("precio"));
            Producto producto = new Producto(0, nombre, precio);
            productoDAO.agregar(producto);
        } else if ("eliminar".equals(accion)) {
            int id = Integer.parseInt(req.getParameter("id"));
            productoDAO.eliminar(id);
        }

        resp.sendRedirect("productos");
    }
}

package dao;

import model.Producto;
import java.util.List;

public interface ProductoDAO {
    List<Producto> listarTodos();
    void agregar(Producto producto);
    void eliminar(int id);
}

package dao;

import model.Producto;
import config.DatabaseConfig;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class ProductoDAOImpl implements ProductoDAO {

    private Connection getConnection() throws SQLException {
        return DatabaseConfig.getConnection();
    }

    // Método para crear la tabla si no existe
    private void initDatabase() {
        String sql = "CREATE TABLE IF NOT EXISTS productos (" +
                     "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
                     "nombre TEXT NOT NULL, " +
                     "precio REAL NOT NULL)";
        try (Connection conn = getConnection();
             Statement stmt = conn.createStatement()) {
            stmt.execute(sql);
            System.out.println("Base de datos y tabla 'productos' creadas o ya existentes.");
        } catch (SQLException e) {
            System.err.println("Error al crear la base de datos: " + e.getMessage());
            e.printStackTrace();
        }
```

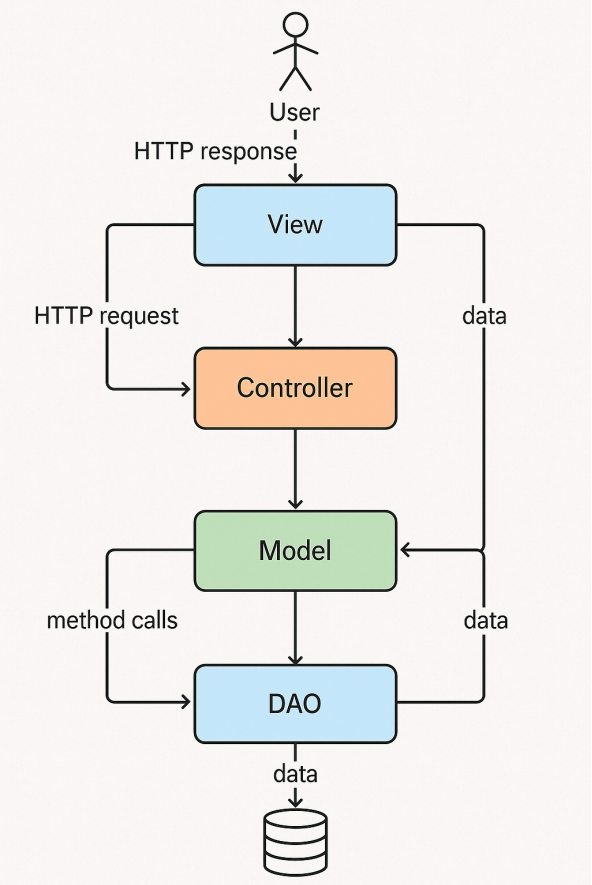
```
}

public ProductoDAOImpl() {
    initDatabase();  // Inicializa la base de datos
}

@Override
public List<Producto> listarTodos() {
    List<Producto> productos = new ArrayList<>();
    String sql = "SELECT * FROM productos";
    try (Connection conn = getConnection();
         Statement stmt = conn.createStatement();
         ResultSet rs = stmt.executeQuery(sql)) {
        while (rs.next()) {
            productos.add(new Producto(
                rs.getInt("id"),
                rs.getString("nombre"),
                rs.getDouble("precio")
            ));
        }
        System.out.println("Productos listados: " + productos.size());
    } catch (SQLException e) {
        System.err.println("Error al listar productos: " + e.getMessage());
        e.printStackTrace();
    }
    return productos;
}

@Override
public void agregar(Producto producto) {
    String sql = "INSERT INTO productos (nombre, precio) VALUES (?, ?)";
    try (Connection conn = getConnection();
         PreparedStatement stmt = conn.prepareStatement(sql)) {
        stmt.setString(1, producto.getNombre());
        stmt.setDouble(2, producto.getPrecio());
        stmt.executeUpdate();
        System.out.println("Producto agregado: " + producto.getNombre());
    } catch (SQLException e) {
        System.err.println("Error al agregar producto: " + e.getMessage());
        e.printStackTrace();
    }
}

@Override
public void eliminar(int id) {
    String sql = "DELETE FROM productos WHERE id = ?";
    try (Connection conn = getConnection();
         PreparedStatement stmt = conn.prepareStatement(sql)) {
        stmt.setInt(1, id);
        stmt.executeUpdate();
        System.out.println("Producto eliminado con ID: " + id);
    } catch (SQLException e) {
```

```
            System.err.println("Error al eliminar producto con ID " + id + ": " + e.getMessage());
            e.printStackTrace();
        }
    }
}

package model;

public class Producto {
    private int id;
    private String nombre;
    private double precio;

    public Producto() {}
    public Producto(int id, String nombre, double precio) {
        this.id = id;
        this.nombre = nombre;
        this.precio = precio;
    }

    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; }
}

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
    <title>Lista de Productos</title>
</head>
<body>
    <h2>Gestión de Productos</h2>
    <form action="productos" method="post">
        <input type="hidden" name="accion" value="agregar">
        <label>Nombre:</label>
        <input type="text" name="nombre" required>
        <label>Precio:</label>
        <input type="number" step="0.01" name="precio" required>
        <button type="submit">Agregar Producto</button>
    </form>

    <h3>Lista de Productos</h3>
    <table border="1">
        <tr>
            <th>ID</th>
            <th>Nombre</th>
            <th>Precio</th>
            <th>Acción</th>
```

```
        </tr>
        <c:forEach var="producto" items="${productos}">
            <tr>
                <td>${producto.id}</td>
                <td>${producto.nombre}</td>
                <td>${producto.precio}</td>
                <td>
                    <form action="productos" method="post">
                        <input type="hidden" name="accion" value="eliminar">
                        <input type="hidden" name="id" value="${producto.id}">
                        <button type="submit">Eliminar</button>
                    </form>
                </td>
            </tr>
        </c:forEach>
    </table>
</body>
</html>


<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.mycompany</groupId>
    <artifactId>Lab06</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>war</packaging>
    <name>Lab06-1.0-SNAPSHOT</name>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <jakartaee>10.0.0</jakartaee>
    </properties>

    <dependencies>
    <!-- Jakarta EE API (ya estaba) -->
    <dependency>
        <groupId>jakarta.platform</groupId>
        <artifactId>jakarta.jakartaee-api</artifactId>
        <version>${jakartaee}</version>
        <scope>provided</scope>
    </dependency>

    <!-- JDBC SQLite -->
    <dependency>
        <groupId>org.xerial</groupId>
        <artifactId>sqlite-jdbc</artifactId>
        <version>3.46.0.0</version>
    </dependency>
```

```
    <!-- JSTL para JSP (compatibilidad) -->
    <dependency>
      <groupId>jakarta.servlet.jsp.jstl</groupId>
      <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
      <version>2.0.0</version>
    </dependency>
    <dependency>
      <groupId>org.glassfish.web</groupId>
      <artifactId>jakarta.servlet.jsp.jstl</artifactId>
      <version>2.0.0</version>
    </dependency>
  </dependencies>


    <build>
      <plugins>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-compiler-plugin</artifactId>
          <version>3.8.1</version>
          <configuration>
            <source>11</source>
            <target>11</target>
          </configuration>
        </plugin>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-war-plugin</artifactId>
          <version>3.3.2</version>
        </plugin>
      </plugins>
    </build>
</project>
```

3. **Realice un diagrama de los patrones MVC + DAO.**



4. **¿Qué opina del avance actual de la IA y cómo se está aplicando en la nueva generación de apps?**

La inteligencia artificial está transformando las aplicaciones al hacerlas más inteligentes, personalizadas y eficientes. Se aplica en áreas como automatización de tareas, salud, vehículos autónomos, seguridad, y creación de contenido. La IA mejora la experiencia del usuario y optimiza procesos, pero también plantea desafíos en privacidad, ética y transparencia, los cuales deben ser gestionados responsablemente.

Conclusiones:

Indicar las conclusiones que llegó después de los temas tratados de manera práctica en este laboratorio.

He aprendido a integrar correctamente bases de datos como SQLite en aplicaciones Java usando JDBC, y he visto cómo manejar excepciones y optimizar las conexiones.

La configuración y el uso de Maven han sido fundamentales para manejar dependencias y automatizar tareas, como la ejecución de servidores o el empaquetado de aplicaciones, lo que facilita mucho el desarrollo.

La experiencia me ha ayudado a mejorar mis habilidades con Jakarta EE, aprendiendo cómo usar frameworks y plugins para gestionar el desarrollo de aplicaciones web robustas.