



TECNOLOGICO NACIONAL DE MEXICO INSTITUTO TECNOLOGICO DE OAXACA

Carrera:

Ingeniería en Sistemas Computacionales

Materia:

Diseño e Implementación de software con patrones

Documentación de patrón singleton

Docente:

Espinoza Pérez Jacob

Equipo:

Ordaz Pacheco Ruudvan

Santos Manuel Julia Marlenny

Vera Acevedo Héctor Aramís

Grupo:

7SB

Fecha de entrega:

07/03/2025

Documentación del Patrón Singleton en el Sistema de Ventas

Introducción

El patrón Singleton es un patrón de diseño creacional que garantiza que una clase tenga solo una instancia y proporciona un punto de acceso global a dicha instancia. En el sistema de ventas, este patrón se implementó en la clase Conexion para administrar la conexión a la base de datos, asegurando que todas las operaciones utilicen la misma instancia de conexión.

Clase Conexion (Singleton)

La clase Conexion fue rediseñada para implementar el patrón Singleton, convirtiéndose en el núcleo de la gestión de conexiones a la base de datos. Sus características principales son:

- Constructor privado para prevenir instanciación externa
- Variable estática que almacena la única instancia
- Método getConexion() sincronizado para acceso thread-safe
- Método cerrarConexion() para liberación controlada de recursos

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class Conexion {
   // l. Instancia única como variable estática
   private static Connection instancia:
    // 2. Constructor privado para evitar instanciación
   private Conexion() {}
    // 3. Método estático para obtener la instancia (versión thread-safe)
   public static synchronized Connection getConexion() {
       if (instancia == null) {
           try (
               // 4. Crear la conexión solo si no existe
                instancia = DriverManager.getConnection(
                   "jdbc:mysql://localhost/bd_sistema_ventas",
                   "root",
                   "1234");
              System.out.println("Conexión establecida");
           } catch (SQLException e) {
               System.err.println("Error al conectar: " + e.getMessage());
               throw new RuntimeException("Error al establecer conexión", e);
       return instancia;
    // 5. Método para cerrar la conexión
   public static void cerrarConexion() {
       if (instancia != null) {
           try {
               instancia.close();
               instancia = null;
               System.out.println("Conexión cerrada");
           } catch (SQLException e) {
               System.err.println("Error al cerrar conexión: " + e.getMessage());
```

Adaptación en Controladores

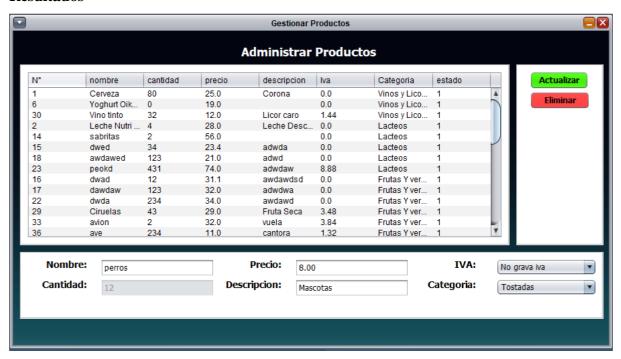
Todos los controladores del sistema (Ctrl_Usuario, Ctrl_Producto, etc.) fueron modificados para utilizar la nueva implementación Singleton:

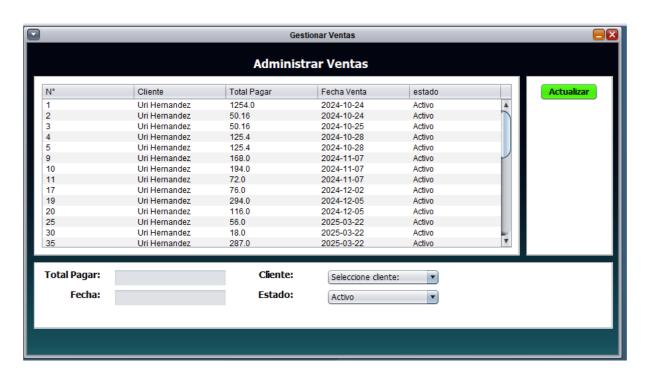
Nota: De igual manera se implementó la conexión nen todas las clases que lo necesitaban como lo son: Todos los controladores, en modelo la clase Ventas y en las vistas serian todas a excepción de FrmLogin y FrmMenu.

Ventajas del Patrón Singleton en este Código

- 1. **Control centralizado:** Todas las partes del sistema utilizan la misma instancia de conexión
- 2. **Eficiencia en recursos:** Evita la creación múltiple de conexiones a la BD
- 3. **Consistencia:** Garantiza que todas las operaciones trabajen con el mismo estado de conexión
- 4. **Facilidad de mantenimiento:** Cambios en la configuración de conexión se realizan en un solo lugar
- 5. **Thread-safe:** La implementación sincronizada previene problemas en entornos concurrentes

Resultados











Conclusión

La implementación del patrón Singleton en la clase Conexion ha proporcionado una solución robusta y eficiente para la gestión de conexiones a la base de datos en el sistema de ventas. Este enfoque ha demostrado ser particularmente beneficioso para:

- Reducir el consumo de recursos
- Simplificar el código de los controladores
- Mejorar la consistencia en las operaciones con la base de datos
- Facilitar el mantenimiento futuro del sistema