



DEPARTAMENTO DE CIENCIAS DE LA COMPUTACION

PROGRAMACION ORIENTADA A OBJETOS

CONTROL DE LECTURA N° 2

TEMA: MODELO VISTA CONTROLADOR (MVC) Y PATRONES DE DISEÑO



REALIZADO POR:

Pablo Guber Camacho Bravo

Estalyn Daniel Licuy Mecías

Nayeli Scarleth Loachamin Tipan

Deisy Abigail Quillupangui Tupe

DOCENTE:

Luis Enrique Jaramillo Montaño

Sangolquí – Ecuador

Modelo Vista Controlador (MVC) y Patrones de Diseño

1. Introducción

En el desarrollo de software, la arquitectura y los patrones de diseño son esenciales para crear aplicaciones escalables, mantenible y eficientes. El Modelo Vista Controlador (MVC) es uno de los patrones arquitectónicos más utilizados en el diseño de aplicaciones, especialmente en aplicaciones web y móviles. Este patrón se enfoca en separar la lógica de la aplicación en tres componentes principales: el Modelo, la Vista y el Controlador. Al emplear el MVC, los desarrolladores pueden facilitar la reutilización del código, mejorar la organización y facilitar el mantenimiento.

Junto con el MVC, existen diversos patrones de diseño que ayudan a estructurar mejor las aplicaciones, cada uno con su propósito específico para resolver problemas comunes durante el desarrollo de software, como la creación de interfaces de usuario, la gestión de la persistencia de datos, o la comunicación entre componentes.

2. Objetivos

2.1 Objetivo General

Analizar el Modelo Vista Controlador (MVC) y los Patrones de Diseño para comprender su impacto en el desarrollo de software y su aplicabilidad en distintos contextos.

2.2 Objetivo Específico

- a. Explicar la estructura y funcionamiento del Modelo Vista Controlador (MVC), detallando sus componentes principales y su rol en la arquitectura de software.
- b. Identificar y describir los principales Patrones de Diseño utilizados en el desarrollo de software, resaltando su importancia y aplicabilidad.
- c. Comparar las ventajas y desventajas del uso de MVC y los Patrones de Diseño, analizando su impacto en la escalabilidad y mantenimiento de sistemas.
- d. Proporcionar recomendaciones y buenas prácticas para la implementación de MVC y Patrones de Diseño en proyectos de software, basadas en estudios y experiencias de la industria.

3. Marco Teórico

El patrón Modelo Vista Controlador (MVC) es un enfoque arquitectónico que divide una aplicación en tres componentes principales:

- **Modelo:** Maneja la lógica de negocio y los datos.
- **Vista:** Se encarga de la presentación de la información.
- **Controlador:** Actúa como intermediario entre el modelo y la vista, gestionando la interacción del usuario.

Estos patrones mejoran la modularidad, reutilización de código y escalabilidad en el desarrollo de software.

3.1 Análisis del proceso

El proceso de implementación de patrones de diseño implica varias etapas clave:

1. **Identificación de problemas en el diseño:** Se analizan los requerimientos del software y se identifican problemas recurrentes en la arquitectura.
2. **Selección del patrón adecuado:** Se elige el patrón de diseño más adecuado para resolver el problema identificado.
3. **Implementación del patrón:** Se incorpora el patrón en el código fuente de la aplicación, asegurando su correcta integración.
4. **Evaluación y optimización:** Se realizan pruebas para verificar la eficiencia del patrón y se optimiza en caso de ser necesario.

3.2 Análisis de requisitos

Requisitos Funcionales:

- Separación clara de responsabilidades en la aplicación.
- Implementación de una arquitectura flexible y modular.
- Facilitar la reutilización de componentes del software.
- Mejorar la interacción entre los diferentes módulos del sistema.

Requisitos No Funcionales:

- Mejorar la mantenibilidad del código.
- Garantizar la escalabilidad y adaptabilidad del software.
- Facilitar la depuración y pruebas del sistema.
- Optimizar el rendimiento del software mediante estructuras bien definidas.

4. Recomendaciones

- a. Uso adecuado de la separación de responsabilidades: Es crucial mantener una clara separación entre el Modelo, la Vista y el Controlador, ya que esto facilita la escalabilidad y el mantenimiento de la aplicación. Asegúrese de que el Modelo no dependa de la Vista y viceversa.
- b. Modularización del código: Los patrones de diseño deben ser implementados de manera modular, donde cada componente se pueda desarrollar, probar y mantener de forma independiente.
- c. Adaptación a nuevas tecnologías: Al implementar MVC, es importante estar al tanto de las nuevas herramientas y marcos de trabajo que apoyan este patrón, como React, Angular o Laravel, para mejorar la productividad y las buenas prácticas.
- d. Pruebas unitarias y de integración: Realizar pruebas exhaustivas, especialmente a nivel de controladores, ya que son los que gestionan la interacción entre el Modelo y la Vista. Esto garantizará una aplicación robusta y libre de errores

5. Conclusiones

- a. Facilita la mantenibilidad: La separación clara entre el Modelo, la Vista y el Controlador en MVC facilita el mantenimiento y la modificación del software. Esto es especialmente útil en aplicaciones a largo plazo donde los requisitos cambian frecuentemente.
- b. Mejora la escalabilidad: Al dividir la aplicación en tres componentes, el patrón MVC hace que sea más fácil escalar y añadir nuevas funcionalidades sin comprometer la estructura general del sistema.
- c. Fomenta el desarrollo colaborativo: MVC permite que equipos de desarrollo trabajen de forma más eficiente. Los desarrolladores de la Vista, por ejemplo, pueden trabajar sin interferir con el Modelo o el Controlador, y viceversa.
- d. Compatible con múltiples plataformas: El patrón MVC es muy flexible y puede adaptarse a diversas plataformas y lenguajes de programación, lo que lo convierte en una opción popular en el desarrollo web y móvil.

6. Resultados

Aquí se presenta las capturas del funcionamiento de los cogidos

Sistemadegestiondebiblioteca

ruta 1:

Sistemadegestiondebiblioteca\src\main\java\com\mycompany\sistemadegestiondebiblioteca

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
Debug: AplicacionBiblioteca + - [ ] ... ^ x

PS C:\Users\essta\Documents\NetBeansProjects\Sistemadegestiondebiblioteca> & 'D:\JDK\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:52202' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\essta\Documents\NetBeansProjects\Sistemadegestiondebiblioteca\target\classes' 'com.mycompany.sistemadegestiondebiblioteca.AplicacionBiblioteca'

Catálogo de Libros:
1 - Don Quijote de la Mancha por Miguel de Cervantes (Disponible)
2 - Cien años de soledad por Gabriel García Márquez (Disponible)
3 - El principito por Antoine de Saint-Exupéry (Disponible)

Mensaje: Libro agregado correctamente

Catálogo de Libros:
1 - Don Quijote de la Mancha por Miguel de Cervantes (Disponible)
2 - Cien años de soledad por Gabriel García Márquez (Disponible)
3 - El principito por Antoine de Saint-Exupéry (Disponible)
4 - 1984 por George Orwell (Disponible)

Mensaje: Libro prestado correctamente

Detalles del Libro:
ID: 2
Título: Cien años de soledad
Autor: Gabriel García Márquez
Estado: Prestado

Mensaje: El libro no está disponible para préstamo

Mensaje: Libro devuelto correctamente

Detalles del Libro:
ID: 2
Título: Cien años de soledad
Autor: Gabriel García Márquez
Estado: Disponible
PS C:\Users\essta\Documents\NetBeansProjects\Sistemadegestiondebiblioteca>
```

Tiendaonlineconproductos\src\main\java\com\mycompany\tiendaonlineconproductos inedaonlineconproductos

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
Debug: AplicacionTienda + - [ ] ... ^ x

Precio: $499.99
Stock: 20 unidades

Mensaje: Producto agregado al carrito

Mensaje: Producto agregado al carrito

Contenido del Carrito:
-----
Producto      Precio  Cantidad  Subtotal
-----
Laptop        $999.99    1      $999.99
Auriculares   $79.99     2     $159.98
-----
TOTAL:                $1159.97

Mensaje: Producto agregado al carrito

Contenido del Carrito:
-----
Producto      Precio  Cantidad  Subtotal
-----
Laptop        $999.99    1      $999.99
Auriculares   $79.99     2     $159.98
Mouse         $29.99     1       $29.99
-----
TOTAL:                $1189.96

Mensaje: Compra realizada con éxito

El carrito está vacío

Catálogo de Productos:
-----
ID  Nombre      Precio  Stock
-----
1   Laptop      $999.99    9
2   Smartphone $499.99   20
3   Auriculares $79.99    48
4   Teclado     $59.99   15
5   Mouse       $29.99   24
-----
PS C:\Users\essta\Documents\NetBeansProjects\Tiendaonlineconproductos>
```

Sistema Gestión de Productos

```
1 package com.mycompany.sisgestionproductos;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 //Modelo
7 class Producto {
8     public int id;
9     public String nombre;
10    public double precio;
11    public int stock;
12
13    public Producto(int id, String nombre, double precio, int stock) {
14        this.id = id;
15        this.nombre = nombre;
16        this.precio = precio;
17        this.stock = stock;
18    }
19
20    // Getters y Setters
21    public int getId() { return id; }
22    public void setId(int id) { this.id = id; }
23    public String getNombre() { return nombre; }
24    public void setNombre(String nombre) { this.nombre = nombre; }
25    public double getPrecio() { return precio; }
26    public void setPrecio(double precio) { this.precio = precio; }
27    public int getStock() { return stock; }
28    public void setStock(int stock) { this.stock = stock; }
29
30    // Vista
31    class ProductoVista {
32        public void mostrarProducto(Producto producto) {
33            System.out.println("Detalles del producto:");
34            System.out.println("ID: " + producto.getId());
35            System.out.println("Nombre: " + producto.getNombre());
36            System.out.println("Precio: $" + producto.getPrecio());
37            System.out.println("Stock: " + producto.getStock());
38        }
39
40        public void mostrarListaProductos(List<Producto> productos) {
41            System.out.println("Lista de productos:");
42            for (Producto producto : productos) {
43                System.out.println("ID: " + producto.getId() +
44                    ", Nombre: " + producto.getNombre() +
45                    ", Precio: $" + producto.getPrecio() +
46                    ", Stock: " + producto.getStock());
47            }
48        }
49
50        public void mostrarMensaje(String mensaje) {
51            System.out.println("Mensaje: " + mensaje);
52        }
53    }
54
55    // Controlador
56    class ProductoControlador {
57        public List<Producto> modeloLista;
58        public ProductoVista vista;
59
60        public ProductoControlador(List<Producto> modelo, ProductoVista vista) {
61            this.modeloLista = modelo;
62            this.vista = vista;
63        }
64
65        public void agregarProducto(int id, String nombre, double precio, int stock) {
66            Producto producto = new Producto(id, nombre, precio, stock);
67            modeloLista.add(producto);
68        }
69    }
70
71
72 }
```

```
73     lista.mostrarMensaje("Producto agregado exitosamente");
74 }
75
76 public void actualizarProducto(int id, String nombre, double precio, int stock) {
77     for (Producto producto : modeloLista) {
78         if (producto.getId() == id) {
79             producto.setNombre(nombre);
80             producto.setPrecio(precio);
81             producto.setStock(stock);
82             lista.mostrarMensaje("Producto actualizado exitosamente");
83             return;
84         }
85     }
86     lista.mostrarMensaje("Producto no encontrado");
87 }
88
89 public void eliminarProducto(int id) {
90     modeloLista.removeIf(producto -> producto.getId() == id);
91     lista.mostrarMensaje("Producto eliminado exitosamente");
92 }
93
94 public void mostrarProducto(int id) {
95     for (Producto producto : modeloLista) {
96         if (producto.getId() == id) {
97             lista.mostrarProducto(producto);
98             return;
99         }
100     }
101     lista.mostrarMensaje("Producto no encontrado");
102 }
103
104 public void mostrarTodosLosProductos() {
105     lista.mostrarListaProductos(modeloLista);
106 }
107 }
```

Output: Run Selectochonoma

```
Output: Run Selectochonoma
Decompiling the module because of changed source code.
Compiling 1 source file with javac [boot classpath C:\Program Files\Java\jdk-11.0.10\jmods\module-info.class] to target platform C:\Program Files\Java\jdk-11.0.10\jmods\module-info.class

Mensaje: Producto agregado exitosamente
Mensaje: Producto agregado exitosamente
Mensaje: Producto agregado exitosamente
Lista de Productos:
ID: 1, Nombre: Laptop, Precio: 3999.99, Stock: 50
ID: 2, Nombre: Mouse, Precio: 229.99, Stock: 50
ID: 3, Nombre: Teclado, Precio: 349.99, Stock: 50
Mensaje: Producto actualizado exitosamente
Mensaje: Producto actualizado exitosamente
Mensaje: Producto eliminado exitosamente
Mensaje: Producto eliminado exitosamente
Lista de Productos:
ID: 1, Nombre: Laptop, Precio: 3999.99, Stock: 50
ID: 2, Nombre: Mouse Teclado, Precio: 229.99, Stock: 50
-----
MENU PRINCIPAL
-----
TOTAL TIEMPO: 1.015 s
Ejecutado en: 2022-02-02T20:24:49-03:00
```

Sistema Gestión de Estudiantes

```

1 // ... 3 líneas
2
3 package com.mycompany.viewgestionestudiante;
4
5 // Modelo
6 class Estudiante {
7     public String nombre;
8     public int id;
9     public double promedio;
10
11     public Estudiante(String nombre, int id, double promedio) {
12         this.nombre = nombre;
13         this.id = id;
14         this.promedio = promedio;
15     }
16
17     public String getNombre() { return nombre; }
18     public void setNombre(String nombre) { this.nombre = nombre; }
19     public int getId() { return id; }
20     public void setId(int id) { this.id = id; }
21     public double getPromedio() { return promedio; }
22     public void setPromedio(double promedio) { this.promedio = promedio; }
23 }
24
25 // Vista
26 class EstudianteVista {
27     public void mostrarDetallesEstudiante(String nombre, int id, double promedio) {
28         System.out.println("Estudiante: " + nombre);
29         System.out.println("Nombre: " + nombre);
30         System.out.println("ID: " + id);
31         System.out.println("Promedio: " + promedio);
32     }
33 }
34
35 // Controlador
36 class EstudianteControlador {
37     public Estudiante modelo;
38     public EstudianteVista vista;
39
40     public EstudianteControlador(Estudiante modelo, EstudianteVista vista) {
41         this.modelo = modelo;
42         this.vista = vista;
43     }
44
45     public void setEstudianteNombre(String nombre) {
46         modelo.setNombre(nombre);
47     }
48
49     public String getEstudianteNombre() {
50         return modelo.getNombre();
51     }
52
53     public void setEstudianteId(int id) {
54         modelo.setId(id);
55     }
56
57     public int getEstudianteId() {
58         return modelo.getId();
59     }
60
61     public void setEstudiantePromedio(double promedio) {
62         modelo.setPromedio(promedio);
63     }
64
65     public double getEstudiantePromedio() {
66         return modelo.getPromedio();
67     }
68 }
69

```



```
71 public void actualizarVista() {
72     vista.mostrarDetallesEstudiante(modelo.getNombre(),
73                                     modelo.getId(),
74                                     modelo.getPromedio());
75 }
76
77 public class DisControlEstudiantes {
78
79     public static void main(String[] args) {
80         // Crear datos del estudiante de la base de datos
81         Estudiante modelo = new Estudiante("Juan Sanchez", 1, 9.5);
82
83         // Crear la vista para mostrar los datos del estudiante
84         EstudianteVista vista = new EstudianteVista();
85
86         // Crear el controlador
87         EstudianteControlador controlador = new EstudianteControlador(modelo, vista);
88
89         // Actualizar vista
90         controlador.actualizarVista();
91
92         // Actualizar datos del modelo a través del controlador
93         controlador.setEstudianteNombre("Juan Carlos Sanchez");
94         controlador.setEstudiantePromedio(9.8);
95
96         // Mostrar datos actualizados
97         controlador.actualizarVista();
98     }
99 }
100
```

Output - Run (GetDetails.java)

Running the project...

----- com.myspringframework.EstudioEstudiantes -----

Building DisControlEstudiantes 1.0-SNAPSHOT

from pom.xml

[1s]

----- com.myspringframework.EstudioEstudiantes -----

slip on existing installation C:\Users\ADMIN\Documents\My Documents\My Recent Documents\EstudioEstudiantes\src\main\resources

Building to compile - all classes are up to date.

----- com.myspringframework.EstudioEstudiantes -----

Execution:

main: Run Method

ID: 1

Promedio: 9.5

ESTUDIANTE:

Nombre: Juan Carlos Sanchez

ID: 1

Promedio: 9.8

Exit: Success

Total time: 0.00 s

Finished at: 2025-10-27 20:10:10 -05:00