DAWS – 2º

Patrón MVC – Modelo Vista Controlador

Introducción

JESÚS MOLINA HERNÁNDEZ
jmolina@florida-uni.es

Florida
Secundària

Índice

¿Qué es MVC?

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

- **Modelo**: Gestiona los datos, la lógica de negocio y las interacciones con la base de datos.
- **Vista**: Se encarga de la presentación y la interfaz de usuario (UI). Muestra los datos que recibe del modelo.
- **Controlador**: Maneja las solicitudes del usuario y coordina las interacciones entre el modelo y la vista.

MVC separa las preocupaciones, lo que facilita el desarrollo y mantenimiento de aplicaciones grandes al dividirlas en componentes con responsabilidades claras.

Jesús Molina Hernández – jmolina@florida-uni.es

¿Relación con Principios y Patrones de diseño?

MVC no es un patrón de diseño, sino un **patrón arquitectónico**, pero **utiliza varios patrones de diseño** para organizar la estructura interna y **cumple con principios de diseño** como **SOLID**:

Principios de Diseño

- **SRP (Single Responsibility)**

Cada componente única responsabilidad (modelo: lógica; vista: presentación; controlador: coordinación).

- **DIP (Dependency Inversion)**

El controlador depende de abstracciones (modelos, vistas), lo que permite mayor flexibilidad.

Patrones de Diseño

- **Patrón Factory**

Utilizado para instanciar modelos y vistas en el controlador.

- **Patrón Observer**

La vista puede ser un "observador" de los cambios en el modelo.

- **Patrón Strategy**

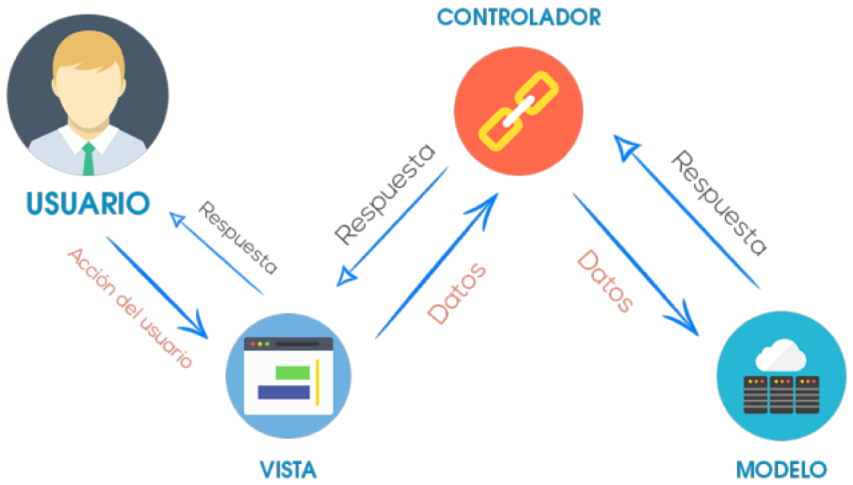
Los controladores pueden implementar diferentes estrategias de manejo de solicitudes.

Conclusión: MVC organiza la aplicación, y los patrones de diseño son usados para resolver problemas específicos dentro de la estructura MVC.

Jesús Molina Hernández – jmolina@florida-uni.es

Diagrama de flujo MVC

MVC no es un patrón de diseño, sino un **patrón arquitectónico**, pero **utiliza varios patrones de diseño** para organizar la estructura interna y **cumple con principios de diseño** como **SOLID**:



1. El usuario realiza una petición desde el navegador
2. El controlador captura el evento
3. El controlador realiza una llamada al modelo/s correspondientes
4. El modelo interactúa con la base de datos y retorna la información al controlador
5. El controlador recibe la información que procesa y la envía a la vista
6. La vista, procesa la información
    1. Capa de abstracción para la lógica (procesa datos)
    2. Capa para el diseño de la interfaz gráfica(los " al GUI)

Jesús Molina Hernández – jmolina@florida-uni.es

Componentes: Modelo

Responsabilidad: El **Modelo** se encarga de la lógica de negocio y la gestión de datos. En aplicaciones PHP, interactúa directamente con la base de datos utilizando **MySQLi** o **PDO**.

Funciones:

- Realiza operaciones CRUD (Crear, Leer, Actualizar, Eliminar).
- Implementa reglas de negocio y validaciones.
- No tiene conocimiento de la vista o el controlador.

```php
class Producto {
    public function obtenerTodos() {
        // Conexión y consulta a la base de datos
        $query = "SELECT * FROM productos";
        $result = $db->query($query);
        return $result->fetch_all(MYSQLI_ASSOC);
    }
}
```


Jesús Molina Hernández – jmolina@florida-uni.es

Componentes: Vista

Responsabilidad: La **Vista** es responsable de la presentación. Genera el **HTML** que se envía al navegador del usuario, utilizando los datos proporcionados por el modelo.

Funciones:

- Renderiza la interfaz de usuario.
- No tiene lógica de negocio, solo contiene la presentación.

```php
<!DOCTYPE html>
<html>
<head>
    <title>Productos</title>
</head>
<body>
<h1>Lista de productos</h1>
<ul>
    <?php foreach ($productos as $producto): ?>
        <li><?php echo $producto['nombre']; ?> - <?php echo $producto['precio']; ?></li>
    <?php endforeach; ?>
</ul>
</body>
</html>
```


Jesús Molina Hernández – jmolina@florida-uni.es

Componentes: Controlador

Responsabilidad: El **Controlador** recibe las solicitudes del usuario (normalmente mediante peticiones HTTP), interactúa con el modelo para obtener o modificar datos y actualiza la vista con los resultados.

Funciones:

- Procesa las solicitudes del usuario.
- Coordina la interacción entre el modelo y la vista.
- No contiene la lógica de presentación ni manipula directamente los datos.

```php
class ProductoControlador {
    public function mostrarProductos() {
        // Crear instancia del modelo
        $productoModel = new Producto();

        // Obtener datos del modelo
        $productos = $productoModel->obtenerTodos();

        // Incluir la vista, pasando los datos
        include 'vistas/productos.php';
    }
}
```


Jesús Molina Hernández – jmolina@florida-uni.es

Ejemplo de MVC sencillo

Verlo en PHPStorm

Jesús Molina Hernández – jmolina@florida-uni.es