**TEMA 5: Sistema MVC**

**1. Introducción**

La orientación a objetos ya supone un primer paso hacia modelos, arquitecturas y patrones como es el MVC (Modelo-Vista-Controlador).

Esto permite mejorar la organización del código, separando las funciones de las clases, lo que a su vez mejora la escalabilidad y mantenimiento de nuestras aplicaciones.

**2. ¿Por qué utilizar el patrón MVC?**

Es importante saber cuando utilizar MVC, ya que puede resultar más aparatoso y costoso.

Una aplicación de una única página con una sola base de datos no requiere de MVC. Pero las aplicaciones de hoy en día son mucho más extensas.

El patrón MVC permite:

* Reutilizar código de forma clara y ordenada.
* Trabajar en equipo por división de tareas.
* Mantener y escalar el código.
* Eliminar dependencias fuertes en el código.

**3. Patrón MVC**

Se construye con 3 componentes distintos:

* Modelo
* Vista
* Controlador

Se basa en una arquitectura de software que permita reutilizar código y separar conceptos.

**3.1 Patrón MVC: Modelo**

El Modelo representa la información con la que opera el sistema. Gestiona los accesos a la información (Consultas y actualizaciones), así como los privilegios de acceso.

En definitiva, engloba todo el desarrollo que interactúe con la base de datos.

Al centrarse solo en esa parte, no será necesario crear un nuevo fichero para cada base de datos, ya que podremos reutilizar una clase construida previamente que haga esas funciones.

Esto hace el Modelo mucho más mantenible, ya que todas las ampliaciones se realizarán en el mismo fichero.

**3.2 Patrón MVC: Vista**

La Vista muestra el Modelo en un formato interactivo. Normalmente se materializa en la interfaz de usuario.

**3.3 Patrón MVC: Controlador**

El Controlador es la parte más compleja de programar y abstraer. Sus funciones son:

* Recibir las peticiones del usuario
* Gestiona la información requerida y acude al Modelo si es necesario
* Muestra la información al usuario a través de la Vista.

**4. Vistas y formularios**

El patrón MVC y los formularios se desarrollan en la Vista.

Primero habría que crear el Modelo, el cual gestionará tanto la administración de la base de datos (parámetros y control de errores), como las consultas a la misma.

Otro fichero se encargará de gestionar la Vista y mostrar los datos obtenidos de la BD. La interacción con el usuario se incorpora mediante formularios que recogen las peticiones y traducen el cambio

**5. CRUD**

CRUD es el acrónimo de las operaciones básicas de una base de datos:

* CREATE: Nuevo registro con Insert.
* READ: Leer registros con Select.
* UPDATE: Actualizar registros con Update.
* DELETE : Borrar registros con Delete.

Para crear un CRUD, se siguen los siguientes pasos:

* Se modifica el Modelo para incorporar la inserción de datos en las tablas.
* Se crean formularios que envíen esta información.
* La actualización de datos es muy similar a la inserción, solo habrá que vigilar que se actualiza el registro correcto.
* El borrado de datos es muy similar a la actualización, solo que no serán necesarios todos los datos para hacerlo.

**6. Funciones, POO y MVC**

**6.1 Parámetros por defecto**

Los parámetros por defecto permiten dar un valor a un parámetro que no se ha definido en una función. Esto permite múltiples definiciones en una misma función.

**6.2. MVC y funciones**

En el modelo MVC los objetos mysqli se quedarán en la parte del Modelo, utilizando allí la función fetch\_assoc para devolver a la Vista la información en forma de Objeto o Array.

**6.3 Herencia y MVC**

Cuando se utiliza una estructura de clases Padre-Hija, puede darse el caso que necesitemos realizar la sobrecarga de los métodos de la clase Padre en la clase Hija.

Para poder acceder a los métodos de la clase Padre, utilizamos el operador de ámbito **parent::**

De esta manera, en el Modelo la Herencia tendrá la siguiente aplicación:

* La clase padre se encargará de la conexión e interacción con la base de datos, normalmente generalizada como “db”.
* Las clases hijas se encargarán de manejar la información. Suelen llamarse como las tablas con las que trabajan

Esta es la tipología clásica de los frameworks basados en MVC. De esta manera conseguimos que:

* Las consultas específicas SQL solo aparecerán en las clases hijas. Necesitará de la clase padre para poder realizar la conexión con la base de datos.
* Se utilizará una función genérica que se encargue de comprobar y realizar las consultas.