

Modelo Vista Controlador

PHP

Vamos a situarnos

Todos habéis hecho ya una aplicación web grande.

En programación hay muchas formas diferentes de organizar la lógica y el código de nuestras aplicaciones.

Es muy importante conseguir una buena forma de organización para no caer en uno de los **mayores problemas**:

No tener un código ordenado y bien aplicado.

¿Qué es el MVC?

(MVC) Es un paradigma de desarrollo web que separa el código en **3 partes** más o menos independientes

Lo importante es que separa la lógica de negocio (requisitos de la empresa) de la interfaz de usuario.

Utilizar este tipo de patrones de desarrollo facilita la **funcionalidad**, el **mantenimiento** y la **escalabilidad** del sistema ya que obtenemos un código muy estándar, óptimo y legible

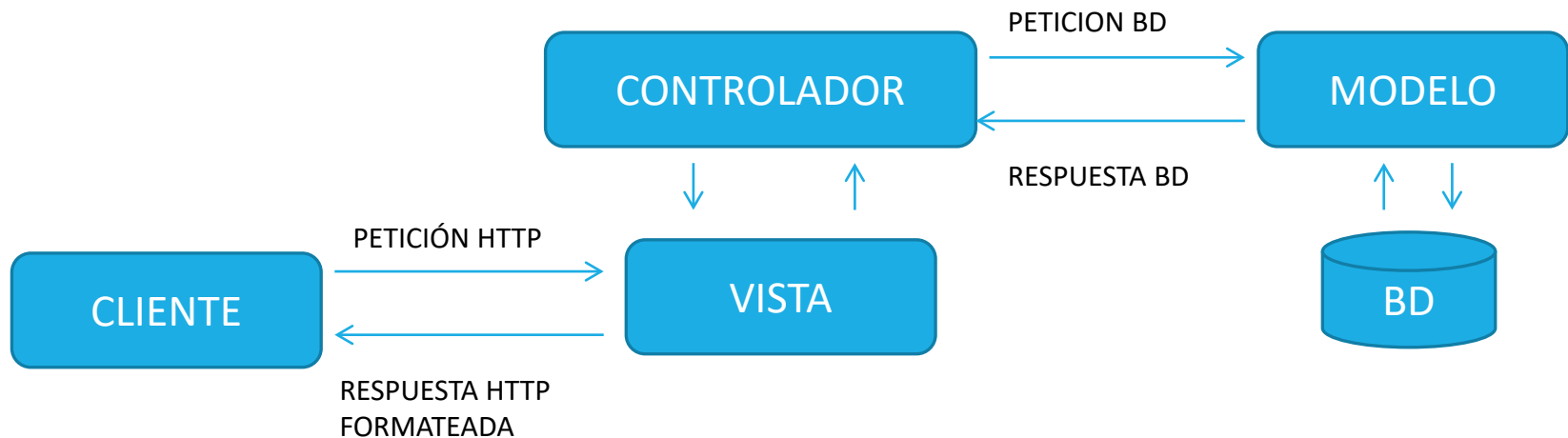
¿Cómo se organiza el MVC?

El MVC divide las aplicaciones en 3 niveles de abstracción

- **Modelo:** gestiona todo lo relacionado con la información de la BD. Un modelo es un conjunto de clases y métodos que se comunican directamente con la BD. Para ser fieles al paradigma de la POO tendremos un **modelo por cada tabla de la base de datos**.
- **Vista:** encargada de mostrar al usuario la información. De forma gráfica (diseño). Será la parte que interacciona con el usuario (formularios, menús, etc)
- **Controlador:** intermediario. Controla las interacciones del usuario en la vista, pide al modelo los datos (invocando sus clases y métodos) y los devuelve de nuevo al usuario.

¿Qué es el MVC?

1. El usuario hace una petición HTTP a una vista
2. La vista hace la llamada al controlador correspondiente
3. El controlador hace la llamada al modelo correspondiente
4. El modelo interactúa con la BD y devuelve al controlador los datos
5. El controlador envía la información a la vista
6. La vista muestra la información



Implementar el MVC en PHP

Implementación: sistema de ficheros

- **Bd:** incluirá todos los ficheros de configuración de la base de datos
- **Controladores:** aquí estarán todos los archivos que se encargan de recibir y filtrar datos que les llegan de las vistas, llamar a los modelos necesarios y pasar los datos de los modelos a las vistas.
- **Modelos:** aquí irán todos los modelos (clases de acceso a la base de datos). Un modelo por cada tabla de la BD.
- **Vistas:** aquí irán todas las vistas, es decir, donde se imprimen por pantalla los datos.
- Lo más normal es que encontréis sus nombres en inglés: db, controllers, models, views...



Ejemplo

- En este ejemplo haremos una pequeña aplicación que permitirá al usuario interactuar con la base de datos “centro”
- El usuario se encontrará un formulario:
 - Si lo envía vacío, la aplicación le mostrará los datos de todos los alumnos
 - Si lo envía relleno, la aplicación le mostrará los datos de los alumnos que coincidan con su búsqueda.

bd/bd.php

```
<?php
class conectar
{
    public static function conexion ()
    {
        $conexion = new mysqli("localhost", "root", "", "centro");
        $conexion->set_charset("utf8");
        return $conexion;
    }
}
?>
```

En BD sólo está la función
que devuelve el conector

Index.php

```
echo "<form action='#' method='post'>
NOMBRE DE ALUMNO <input type='text' name='nom'>
<br> <input type='submit' name='enviar'>
</form>";

if(isset($_POST['enviar']))
{
    include "bd/bd.php";
    include "controladores/controlador_alumnos.php";
}
```

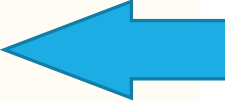
El index sólo hace una
llamada al controlador

```
<?php
```

```
class modelo_alumnos{
    private $bd;
    private $alumnos;
    public function __construct(){
        //el constructor solo se conecta a la bd
        $this->bd=conectar::conexion();
    }
    public function get_alumnos(){
        //saca toda la información de los alumnos
        $consulta=$this->bd->query("select * from alumnos;");
        while($filas=$consulta->fetch_assoc()){
            $this->alumnos[]=$filas;
        }
        return $this->alumnos;
    }
    public function get_alumnos_nombre($nom){
        //saca los alumnos que coincidan con la búsqueda
        $consulta = $this->bd->prepare("select * from alumnos where nombre=?");
        $consulta->bind_param("s", $nom);
        $consulta->bind_result($dni, $nombre, $edad);
        $consulta->execute();
        $consulta->store_result();

        $i=0;
        while($fila = $consulta->fetch())
        {
            $this->alumnos[$i]["dni"]=$dni;
            $this->alumnos[$i]["nombre"]=$nombre;
            $this->alumnos[$i]["edad"]=$edad;
            $i++;
        }
        return $this->alumnos;
    }
}
```

modelos/modelo_alumnos.php




El modelado de los alumnos incluye una clase que permitirá manejar los datos de la tabla alumnos

```
<?php
//Llamada al modelo
require_once "modelos/modelo_alumnos.php";

$per=new modelo_alumnos();

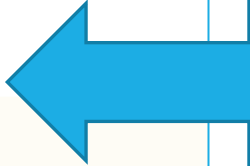
if(isset($_POST["nom"]) && $_POST["nom"]!="")
{
    //si han introducido algo, llamo a un método
    $datos = $per->get_alumnos_nombre($_POST["nom"]);
}
else
{
    //si no han introducido nada, llamo a otro método
    $datos=$per->get_alumnos();
}
//Llamada a la vista
//A la vista le da igual si hay muchos alumnos en $datos
//o solo uno
include "vistas/vista_alumnos.php";
?>
```



Aquí está toda la funcionalidad de la aplicación, haciendo llamadas a los modelos de datos para obtener datos de la BD y a las vistas correspondientes para que los muestren

- Será el controlador el que decida qué método de la clase “modelo_alumnos” es al que necesita llamar para responder a las necesidades del usuario

```
<html lang="es">
  <head>
    <meta charset="UTF-8" />
    <title>Alumnos</title>
  </head>
  <body>
    <?php
      foreach ($datos as $dato) {
        echo "$dato[dni] - $dato[nombre] - $dato[edad]<br/>";
      }
    ?>
  </body>
</html>
```



Las vistas,
simplemente dan
forma los datos que
generan los
controladores

- En este caso, a la vista que muestra los alumnos le da igual si los datos que le llegan vienen de haber ejecutado un método u otro.
- Sabe que tiene una matriz con los alumnos que sean y que de cada alumno tiene dni, nombre y edad (le da igual el número)

Modelo Vista Controlador

PHP