

# TUTORIAL

## PATRÓN MVC



[www.facebook.com/codigonexogr](http://www.facebook.com/codigonexogr)



[@codigonexo](https://twitter.com/codigonexo)



[www.plus.google.com/+Codigonexogr/](http://www.plus.google.com/+Codigonexogr/)



## Contenidos

1. Introducción al patrón MVC
  - 1.1. Conceptos básicos – Qué es un patrón
  - 1.2. Estructura del patrón MVC
  - 1.3. Cómo funciona el patrón MVC
2. Un sencillo ejemplo
  - 2.1. Requisitos previos
  - 2.2. El modelo
  - 2.3. La vistas
  - 2.4. El controlador
  - 2.5. El resultado

## Descripción del tutorial

Primeros pasos en el patrón modelo-vista-controlador. Introducción, definición, estructura y funcionamiento ilustrados con un ejemplo en PHP

## 1. INTRODUCCIÓN AL PATRÓN MVC

### 1.1. CONCEPTOS BÁSICOS – QUÉ ES UN PATRÓN

MVC son las siglas de modelo-vista-controlador, uno de los patrones más utilizados por ser la base sobre la que se asientan distintos gestores de contenido como Wordpress o Joomla!

Para entender lo que significa debemos empezar por explicar qué es un patrón arquitectónico: con este término nos referimos a la forma de organizar los componentes de un sistema aplicando ciertas normas de diseño para fortalecer su usabilidad y prepararlo para su evolución.

¿En castellano, por favor? Fácil: se trata de la forma de organizar los componentes de un sistema para conseguir que la calidad del software sea satisfactoria cumpliendo los siguientes atributos:

- ✿ **Configurabilidad:** que el usuario pueda realizar ciertos cambios en el sistema
- ✿ **Integrabilidad** de los módulos independientes del sistema
- ✿ **Integridad:** que la información solo pueda ser modificada por quien esté autorizado y de manera controlada.
- ✿ **Interoperabilidad** con otros sistemas
- ✿ **Modificabilidad** con vistas al futuro
- ✿ **Mantenibilidad** sencilla
- ✿ **Portabilidad:** que se pueda ejecutar en distintos softwares y hardware
- ✿ **Reusabilidad:** que la estructura pueda ser reutilizada en futuros sistemas
- ✿ **Escalabilidad:** que el sistema se pueda ampliar
- ✿ **Testeabilidad:** facilidad para ser sometido a pruebas que aseguren que el sistema falla cuando es lo que se espera
- ✿ **Confidencialidad:** no se permite el acceso no autorizado
- ✿ Cumplimiento de la **funcionalidad** requerida
- ✿ **Seguridad** externa e interna: prevenir ataques de terceros o usos no autorizados

### 1.2. ESTRUCTURA DEL PATRÓN MVC

El patrón MVC es un patrón de arquitectura de software encargado de separar la lógica del negocio de la interfaz del usuario y es el más utilizado en aplicaciones Web, ya que facilita la funcionalidad, mantenibilidad y escalabilidad del sistema de forma sencilla, a la vez que permite no mezclar lenguajes de programación en el mismo código.

La programación de la aplicación se separa en tres componentes con sus respectivas responsabilidades: un modelo, una o varias vistas, y uno o varios controladores.

### **El modelo**

El modelo implementa la lógica de la aplicación, es decir, almacena todos los datos y el estado de la aplicación y tiene los métodos que manipulan esos datos. Por ejemplo, en una aplicación bancaria el modelo lo forman aquellas clases que representan las cuentas, transacciones, etc. es decir, los datos importantes para el dominio del problema así como los métodos que permiten manipularlos. El modelo no es "consciente" de la vista y el controlador (no los referencia directamente).

### **La vista**

La vista es la interfaz de usuario. Muestra al usuario una representación visual del modelo, sus datos y estado, tomándolos directamente del modelo. También contiene los elementos de la interfaz que permiten al usuario interactuar con el programa, tales como botones y menús. Sin embargo, no es tarea de la vista implementar cómo se deben comportar esos elementos.

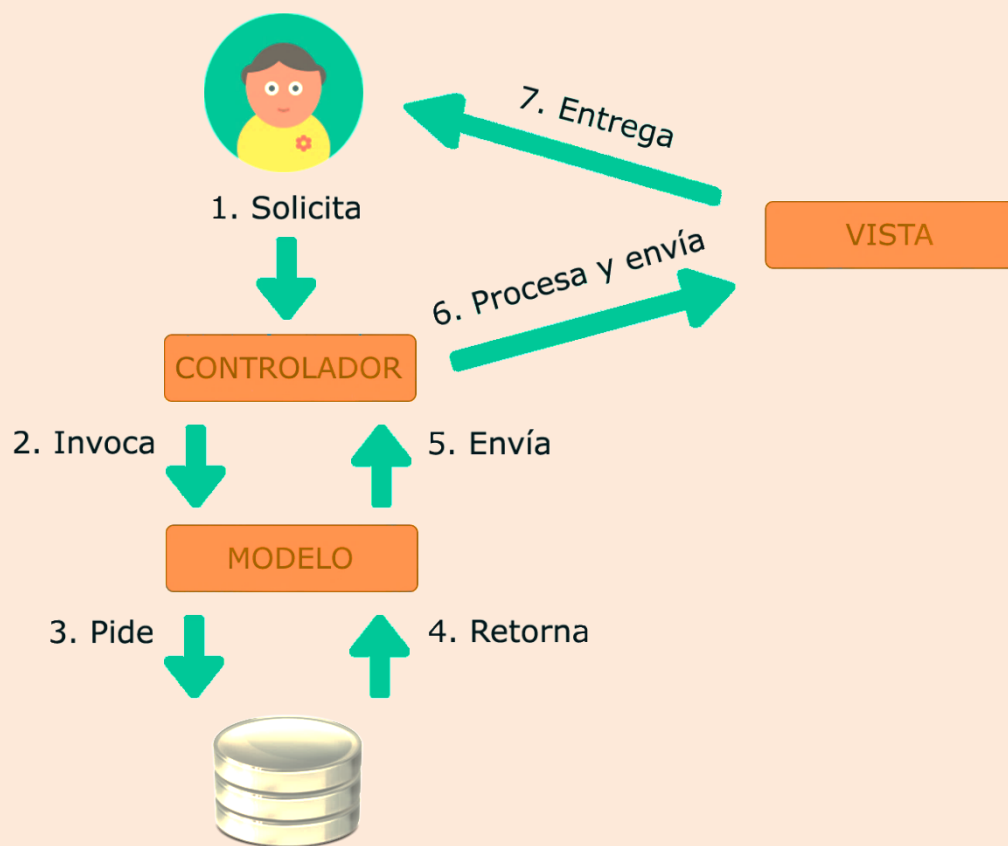
### **El controlador**

El controlador es el intermediario entre la vista y el modelo. Es quien controla las interacciones del usuario solicitando los datos al modelo y entregándolos a la vista para que ésta lo presente al usuario.

## **1.3. CÓMO FUNCIONA EL PATRÓN MVC**

El funcionamiento básico del patrón MVC es el siguiente:

1. El usuario realiza una petición que captura el controlador
2. El controlador llama al modelo correspondiente
3. El modelo solicita la información a la base de datos
4. El modelo recoge la información de la base de datos
5. El controlador recibe la información
6. El controlador procesa y envía la información a la vista
7. La vista entrega al usuario la información de forma "humanamente legible".



## 2. UN SENCILLO EJEMPLO

### 2.1. REQUISITOS PREVIOS

Vamos a ilustrar el funcionamiento del patrón MVC con un sencillo ejemplo en PHP nativo (sin frameworks) en el que mostraremos una tabla con información guardada en la base de datos.

Para ello, lo primero que debemos hacer es iniciar nuestro servidor (Xampp, Wamp, Lamp, Mamp...) y acceder a localhost/phpmyadmin.

Crearemos una nueva Base de datos llamada "patronmvc" (sin comillas) y una tabla llamada "personas" (sin comillas) con las siguientes columnas:

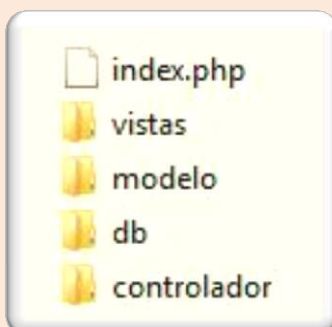
- id (clave primaria)
- nombre
- edad
- estatura

Insertamos algunos valores para mostrarlos posteriormente, y ya tendremos lista nuestra base de datos:

+ Opciones

						id	nombre	edad	estatura	
<input type="checkbox"/>		Editar		Copiar		Borrar	1	Bruno	16	1.65
<input type="checkbox"/>		Editar		Copiar		Borrar	2	Adolfo	54	1.82
<input type="checkbox"/>		Editar		Copiar		Borrar	3	Olga	32	1.68
<input type="checkbox"/>		Editar		Copiar		Borrar	4	Sara	15	1.59
<input type="checkbox"/>		Editar		Copiar		Borrar	9	Lucas	82	1.73

A continuación, accedemos a la carpeta \htdocs si estamos utilizando Xampp, o a \www si por el contrario usamos Wamp. Creamos una nueva carpeta en la raíz llamada "mvc", y dentro de esta carpeta crearemos la siguiente estructura de ficheros:



- ✦ index.php
- ✦ Carpeta "vistas" con los ficheros `personas_vista1.php` y `personas_vista2.php` en su interior
- ✦ Carpeta "modelo" con el fichero `personas_modelo.php` en su interior
- ✦ Carpeta "controlador" con el fichero `personas_controlador.php` en su interior
- ✦ Carpeta "db" con el fichero `db.php` en su interior

Comenzaremos por el index.php puesto que su única función es incluir el fichero de la base de datos y el controlador, necesarios para lanzar nuestro ejemplo:

#### #index.php

```
<?php
    require_once ("db/db.php");
    require_once ("controlador/personas_controlador.php");
?>
```

Continuamos con el fichero db.php, que está localizado dentro de la carpeta /db. Este fichero solo se encarga de establecer una clase Conectar para abrir una conexión con la base de datos, no forma parte del modelo, de la vista, ni del controlador. Indicaremos el usuario de la base de datos (en nuestro caso es "root"), la contraseña (aquí la dejamos en blanco porque no tenemos ninguna) y el nombre de la base de datos (que como bien recordarás es "patronmvc").

#### #db/db.php

```
<?php
class Conectar{
    public static function conexion(){
        $conexion=new mysqli("localhost", "root", "", "patronmvc");
        $conexion->query("SET NAMES 'utf8'");
        return $conexion;
    }
}
?>
```

## 2.2. EL MODELO

Como ya sabes, el modelo es la lógica de negocios e implementa las clases y métodos que se comunican directamente con la base de datos.

Para configurar el modelo crearemos un fichero llamado personas\_modelo.php dentro de nuestra carpeta /modelo. Aquí vamos a generar una clase formada por:

- Dos atributos privados ("db" y "personas")
- Un constructor en el que asignaremos a "db" la conexión con la base de datos y estableceremos "personas" como un array
- Una función get\_personas en la que haremos la consulta a la base de datos y devolveremos los resultados en el array de personas.

De este modo, acabamos de crear una clase que se encargará de pedir la información que necesitemos a la base de datos. El array que devuelve la función get\_personas contiene dichos datos (el id, el nombre, la edad y la estatura de todas las personas).

## #modelo/personas\_modelo.php

```
<?php
class personas_modelo{
    private $db;
    private $personas;

    public function __construct(){
        $this->db=Conectar::conexion();
        $this->personas=array();
    }
    public function get_personas(){
        $consulta=$this->db->query("select * from
personas;");

        while($filas=$consulta->fetch_assoc()){
            $this->personas[]=$filas;
        }

        return $this->personas;
    }
}

?>
```

### 2.3. LAS VISTAS

La vista se encargará de mostrar la información al usuario de forma gráfica y legible. Ya habrás observado que dentro de la carpeta /vistas vamos a guardar dos ficheros:

- personas\_vista1.php: mostrará una tabla con el nombre de las personas y su respectiva edad
- personas\_vista2.php: mostrará una tabla con el nombre de las personas y su respectiva estatura

Es frecuente tener más de una vista en el patrón MVC puesto que así podemos elegir entre distintas maneras de mostrar la información al usuario. Más adelante explicaremos cómo cambiar de una vista a otra realizando tan solo un sencillo cambio en el controlador. Por ahora vamos a configurar las vistas:

Comenzamos por el archivo personas\_vista1.php. En el body dibujaremos una tabla con dos columnas, la primera mostrará el nombre de las personas, y la segunda su edad. Desde el controlador (que veremos a continuación) hemos indicado que los datos se envíen a la vista en una array llamada "\$datos", de modo que utilizaremos php para recorrerlo con un foreach.

**Nota:** Aquí nos limitamos a mostrar las tablas sin ningún tipo de formato puesto que solo es un ejemplo. Por supuesto, si quisiéramos dar formato al código html podríamos asociarle contenido .css como de costumbre.

### #vistas/personas\_vista1.php

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8" />
    <title>Nombre y edad</title>
  </head>
  <body>
    <table>
      <tr>
        <td>Nombre</td>
        <td>Edad</td>
      </tr>

      <?php
      foreach ($datos as $dato) {
        echo "<tr><td>". $dato['nombre']. "</td>";
        echo "<td>". $dato['edad']. "</td></tr>";
      }
      ?>

    </table>
  </body>
</html>
```

La otra vista es exactamente igual, salvo que en lugar de mostrar la edad de las personas mostramos su estatura. Por lo demás, el funcionamiento es el mismo:

### #vistas/personas\_vista1.php

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8" />
    <title>Nombre y estatura</title>
  </head>
  <body>
    <table>
      <tr>
        <td>Nombre</td>
        <td>Estatura</td>
      </tr>

      <?php
      foreach ($datos as $dato) {
        echo "<tr><td>". $dato['nombre']. "</td>";
        echo "<td>". $dato['estatura']. "</td></tr>";
      }
      ?>

    </table>
  </body>
</html>
```



## 2.4. EL CONTROLADOR

El controlador actúa como intermediario entre la vista y el modelo, para ello debemos copiar el siguiente código en `personas_controlador.php` (carpeta `/controlador`).

Por una parte, lo que hacemos en estas líneas es llamar al modelo con un `require_once`, y pedirle los datos creando un objeto “\$per” (recordemos que en el modelo habíamos hecho una clase llamada `personas_modelo`). Hecho esto, llamamos al método `get_personas()` que también habíamos definido en el modelo, y guardamos el array que devuelve en una variable llamada `$datos`. Como bien habrás imaginado ya, esta es la variable `$datos` con la que hemos trabajado en la vista.

Pero, ¿cómo ha podido acceder la vista a `$datos`? Sencillamente, ha podido porque después de llamar al modelo también llamamos a la vista con otro `require`, y es aquí donde indicamos cuál de las dos vistas queremos utilizar. En el ejemplo estamos utilizando la vista 2, por lo que la salida nos mostrará los nombres y la estatura. Si quisiéramos mostrar la edad en lugar de la estatura solo tenemos que modificar la última línea del código indicando que queremos la vista 1 en el `require`.

**Nota:** primero se ha de llamar al modelo y luego a la vista, este orden es obligatorio.

### #controlador/personas\_controlador.php

```
<?php
//Llamada al modelo
require_once("modelo/personas_modelo.php");
$per=new personas_modelo();
$datos=$per->get_personas();

//Llamada a la vista
require_once("vistas/personas_vista2.php");
?>
```

## 2.5. RESULTADO

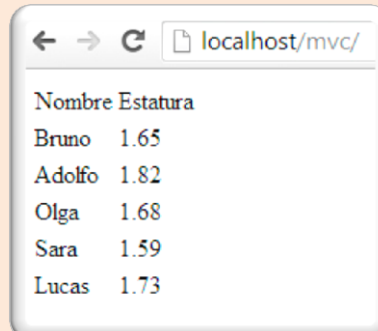
Ya podemos llamar a nuestro index escribiendo `localhost/mvc` en nuestro navegador.

Vista 1



Nombre	Edad
Bruno	16
Adolfo	54
Olga	32
Sara	15
Lucas	82

Vista 2



Nombre	Estatura
Bruno	1.65
Adolfo	1.82
Olga	1.68
Sara	1.59
Lucas	1.73