PROGRAMACIÓN WEB EN ENTORNO SERVIDOR UD-14

MODELO VISTA-CONTROLADOR EN PHP

ÍNDICE

[DEFINICIONES 3](#_Toc155882240)

[VENTAJAS DEL MVC 3](#_Toc155882241)

[EJEMPLO 3](#_Toc155882242)

[A. HANDLER 4](#_Toc155882243)

[B. CONTROLADOR 4](#_Toc155882244)

[ESTRUCTURA 6](#_Toc155882245)

[A. config/config.php 6](#_Toc155882246)

[B. controller/ControladorNota.php 6](#_Toc155882247)

[HANDLER 6](#_Toc155882248)

# DEFINICIONES

Un patrón arquitectónico, define una plantilla para construir el Software, siendo una particularidad del estilo arquitectónico elegido.

El MVC puede ser enmarcado dentro del estilo arquitectónico orientado a objetos MVC es un patrón de arquitectura de software, que aporta cierto orden y escalabilidad en los desarrollos. En un primer momento puede resultar intimidante o complejo, pero si lo analizamos con calma, comprobaremos que tal dificultad no es real.

MVC significa modelo (model) vista (view) controlador (controller). Esto es lo que significan cada uno de esos componentes.

* **Controlador** (Ordenes) se encarga de solicitar datos al modelo y enviarlos a la vista. Interconecta el frontend con el backend. Diagrama de clases diseño, DFD, Casos de uso, Diagramas de actividad y Diagramas de Despliegue.
* **Modelo** (Datos) se encarga de, por ejemplo, cargar datos y realizar operaciones en ellos. Modelo ER, Diagrama de clases conceptual.
* **Vista** (Vista) el frontend o interfaz gráfica de usuario (GUI), es la capa que ve el usuario. Plantillas, boceto.

# VENTAJAS DEL MVC

Entre las ventajas que nos aporta el modelo vista controlador podremos destacar: escalabilidad, orden de código y fácilmente entendible.

MVC está ampliamente implantado y aceptado en el desarrollo de software desde hace muchísimos años.

Te ayuda a dividir el código frontend y el código backend, permitiendo así realizar cambios en las aplicaciones a varios equipos, por ejemplo, programadores PHP y diseñadores de interfaces web.

# EJEMPLO

En MVC, todo comienza con una petición del usuario. En una aplicación Web, la petición del usuario podría ser, por ejemplo, “agregar un nuevo registro”.

Vamos a programar en PHP una aplicación muy simple que nos permita listar, editar y eliminar notas de texto.

Este ejemplo es muy sencillo, ya que sólo utilizaremos un controlador y dos modelos. El único controlador se encargará de gestionar las notas, y los dos modelos serán uno para notas y otro para la base de datos.

## HANDLER

¿Cómo maneja el controlador, la petición del usuario? A través de Handler.

Un handler (o “manejador”) de peticiones, es un objeto encargado de gestionar las peticiones del usuario a nivel de la aplicación. Este handler, descifrará dichas peticiones, realizando un trabajo de pseudo ingeniería inversa, sobre la URI.

* dominio/modulo/modelo/recurso[/atributos]
* http://app.dominio.com/usuarios/usuario/agregar-usuario/

Por ejemplo, para la URI anterior, el handler podría realizar los siguientes pasos:

1. Identificar el dominio, eliminarlo del contexto y así solo obtener el último tramo de la URI:

$dominio = "http://{$\_SERVER['SERVER\_NAME']}";

$uri = $dominio . $\_SERVER['REQUEST\_URI'];

$ultimo\_tramo = str\_replace($dominio."/", "", $uri);

1. Hacer un explode del último tramo de la URI, para obtener un array de 3 elementos (módulo, modelo y recurso respectivamente):

$partes = explode("/", $ultimo\_tramo);

1. Finalmente, el objeto handler, retornará al controlador, el módulo, el modelo y el recurso solicitados por el usuario, y será el controlador, quien prosiga con el resto.

## CONTROLADOR

¿Qué hace el controlador una vez que recibe la información retornada por el handler? La analiza para saber a qué modelo deberá instanciar.

Una vez que el controlador recibe la información retornada por el handler, procede a analizarla en conjunto:

* El handler me envió la siguiente información:
  + Módulo: usuarios
  + Modelo: usuario
* Recurso: agregar-usuario Entonces, debo agregar un nuevo usuario. Para ello, voy a instanciar el objeto, modificar sus propiedades y llamar al método correspondiente.

¿Qué hace el controlador con la información retornada por el modelo? La entrega a la vista, para que ésta, se la muestre al usuario.

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente

Acceso a bases de dato.

Un “objeto-conector” a nivel del core, que actúe como intermediario entre los modelos y la base de datos, será muy recomendable para respetar el verdadero espíritu del patrón arquitectónico MVC.

Este objeto, será una clase preferentemente estática. Es decir, que necesitará ser instanciada para hacer uso de los métodos destinados a acceder a las bases de datos.

Se puede definir también un atributo en las clases de datos a modo de composición, pero no es tan correcto.

La BD que vamos a usar será muy simple:

notas (id,titulo,contenido);

# ESTRUCTURA

Texto

Descripción generada automáticamente

## config/config.php

En este fichero guardaremos información básica que atañe tanto a la conexión con base de datos como o tros parámetros. Por ejemplo, el controlador y la acción por defecto.

Si esta aplicación creciera, seguramente el uso de este fichero estaría más justificado.

## controller/ControladorNota.php

Será el único controlador de nuestra aplicación. Se encarga de recibir las peticiones desde la vista, solicitar información y/u ordenar cambios al modelo.

# HANDLER

Hasta ahora, accedíamos a nuestra aplicación, realizando las peticiones mediante HTTP GET. Con las URL amigables, reemplazamos dicho método, tomando el valor de cada argumento (anteriormente enviado por HTTP GET) y convirtiéndolo en parte de la URL.

Petición antigua con HTTP GET:

<http://aws.es/ejemplomvc/index.php?controller=ControladorNota&action=list&pagina=2>

Petición con handler metiendo los parámetros en la URI:

/ControladorNota/list/2

Es evidente que, si tenemos más de una página en nuestro sitio, podemos tener que indicar la ruta de nuestro programa.

<http://aws.es/ejemplomvc/ControladorNota/list/2>

Si deseamos usar un nombre de dominio pero no queremos instalar un servidor DNS, podemos asociar un nombre de dominio a nuestra ip modificando el archivo hosts:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Vamos a necesitar:

1. Configurar Apache, habilitando el módulo rewrite.

2. Modificar nuestro virtual host, permitiendo sustituir las URL.

3. Crear un archivo .htaccess

Vamos a crear una clase Handler que contenga como atributos el nombre de la aplicación, el controlador, el recurso y un array con los parámetros que se han pasado. En el constructor se va a analizar la URI y determinar los atributos.

Texto

Descripción generada automáticamente