****

Facultad de Ciencias e Ingeniería

Escuela académico profesional

Ingeniería de sistemas e informática

“Tema de investigación: MVC”

**Estudiantes:**

CONDORI CHIPANA, Marco

TRINIDAD QUIÑONES, Oscar

ZAPATA PAULINI, Joselyn

**Docente:**

Prof. Coronel Castillo Erick Gustavo

INDICE

* Resumen
* Antecedentes
* Desarrollo del tema
* Casos
* Conclusiones
* Recomendaciones

**Resumen**

El presente trabajo de investigación consiste en definir como se emplea el MVC con PHP, que es un patrón de arquitectura de software encargado de separar la lógica de negocio de la interfaz del usuario y es el más utilizado en aplicaciones Web, ya que facilita la funcionalidad, mantenibilidad y la escalabilidad del sistema, de forma simple y sencilla, a la vez que permite no mezclar lenguajes de programación en el mismo código.

Aquí hablaremos sobre los antecedentes con este patrón de arquitectura, porque del uso en páginas web.

Además en que consiste el MVC, cual es el proceso de desarrollo al usar el patrón.

Presentaremos un pequeño ejemplo con el código del proceso de MVC en PHP.

Al final concluiremos en que sirve en su totalidad el MVC y una recomendación de uso.

**Antecedentes**

Para aclarar, no existe una definición única, exacta, abarcativa e inequívoca de “arquitectura de software”. La bibliografía sobre el tema es tan extensa como la cantidad de definiciones que en ella se puede encontrar. Por lo tanto, puede definirse sino más bien introducir a un concepto simple y sencillo que permita comprender el punto de vista de esta investigación, sin animo que se presente como “una definición más”

A grandes rasgos puede decirse que…

“La arquitectura de software es la forma en la que se organizan los componentes de un sistema, interactúan y se relacionan entre si y el contexto, aplicando normas y principios de diseño y calidad que fortalezcan y fomenten la usabilidad a la vez que dejan preparado el sistema para su propia evolución” (Brest, 2001).

El patrón MVC fue una de las primeras ideas en el campo de las interfaces gráficas de usuario y uno de los primeros trabajos en describir e implementar aplicaciones software en términos de sus diferentes funciones.

MVC fue introducido por Trygve Reenskaug (web personal) en Smalltalk-76 durante su visita a Xerox Parc en los años 70 y, seguidamente, en los años 80, Jim Althoff y otros implementaron una versión de MVC para la biblioteca de clases de Smalltalk-80. Solo más tarde, en 1988, MVC se expresó como un concepto general en un artículo sobre Smalltalk-80.

En esta primera definición de MVC ***“…el controlador se definía como el módulo que se ocupa de la entrada****"* (Krasner, 1988) (de forma similar a como la vista" se ocupa de la salida"). Esta definición no tiene cabida en las aplicaciones modernas en las que esta funcionalidad es asumida por una combinación de la 'vista' y algún framework moderno para desarrollo. El 'controlador', en las aplicaciones modernas de la década de 2000, es un módulo o una sección intermedia de código, que hace de intermediario de la comunicación entre el 'modelo' y la 'vista', y unifica la validación (utilizando llamadas directas o el "observer" para desacoplar el 'modelo' de la 'vista' en el 'modelo' activo).

Algunos aspectos del patrón MVC han evolucionado dando lugar a ciertas variantes del concepto original, ya que *"****las partes del MVC clásico realmente no tienen sentido para los clientes actuales****"*(Krasner, 2000):

• HMVC (MVC Jerárquico)

• MVA (Modelo-Vista-Adaptador)

• MVP (Modelo-Vista-Presentador)

• MVVM (Modelo-Vista Vista-Modelo)

... y otros que han adaptado MVC a diferentes contextos.

**Desarrollo del Tema**

**CAPITULO I: ¿Qué ES MVC?**

Según Martin Fowler en The evolution of MVC:

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

Como vemos en la cita, el MVC es un patrón muy importante en las aplicaciones web ya que nos da facilidad en el funcionamiento del sistema y además no mezcla, en el mismo código, lenguajes de programación distintos.

**CAPITULO II: DIVISION DE MVC**

MVC divide las aplicaciones en tres niveles de abstracción:

**Modelo:** representa la lógica de negocios. Es el encargado de accesar de forma directa a los datos actuando como “intermediario” con la base de datos.

**Vista:** es la encargada de mostrar la información al usuario de forma Gráfica y “humanamente legible”

**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 esta, lo que al usuario, de forma “humanamente legible”

**CAPITULO III: FUNCIONAMIENTO DE MVC**

Su funcionamiento es básico:

El usuario realiza una petición, mientras que el controlador captura el evento (puede hacerlo mediante un manejador de eventos –handler-, por ejemplo)

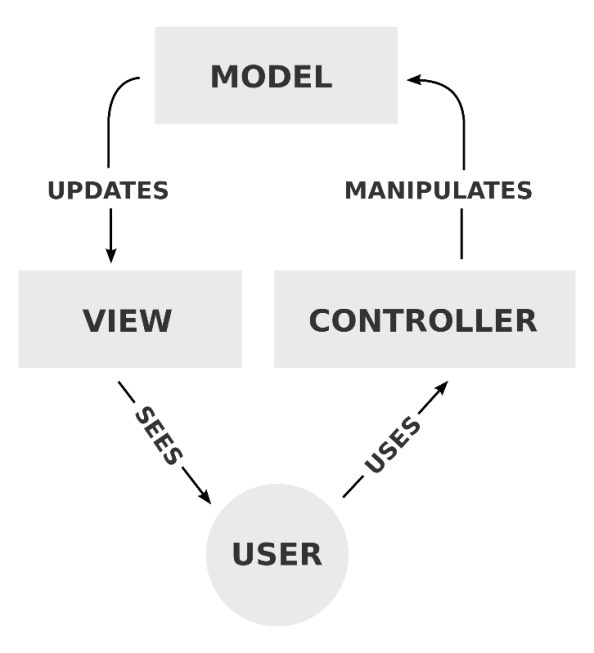
Hace la llamada al modelo/modelos correspondientes (por ejemplo, mediante una llamada de retorno -callback-) efectuando las modificaciones pertinentes sobre el modelo.

El modelo será el encargado de interactuar con la base de datos, ya sea en forma directa, con una capa de abstracción para ello, un web service, etc. Y retornara esta información al controlador.

El controlador recibe la información y la envía a la vista.

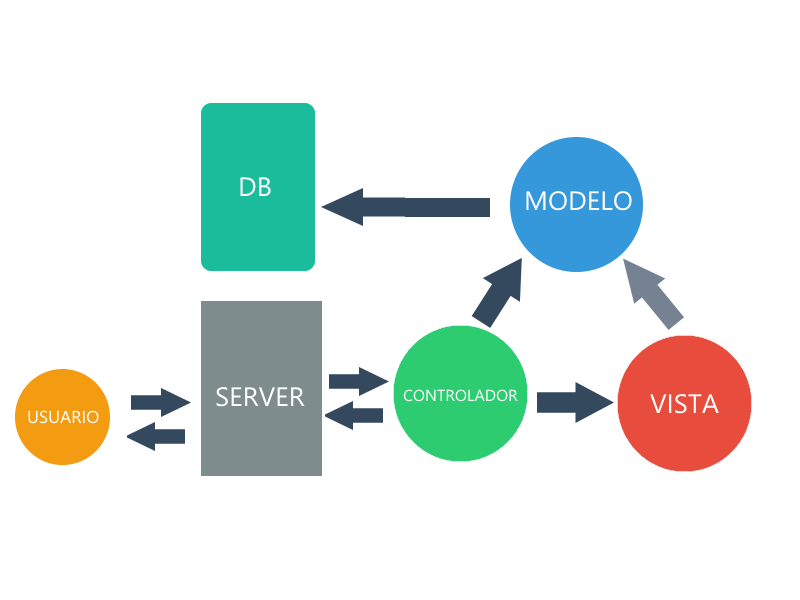
La vista, procesa esta información pudiendo hacerlo, creando una capa de abstracción para la lógica (quien se encargara de procesar los datos) y la otra para el diseño de interfaz gráfica o GUI.

La lógica de la vista, una vez procesados los datos, los “acomodara” en base al diseño de la GUI -layout- y los entregara al usuario de forma “humanamente legible”.



**CAPITULO IV: PATRON DE MVC**

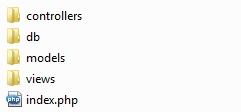
Para comprender un poco más este patrón de diseño veamos cómo funciona en una aplicación web común:



* El usuario solicita una acción al servidor
* El servidor atiende la petición y manda a llamar al controlador
* El controlador llama al modelo necesario
  + El modelo atiende la petición y realiza las operaciones de datos correspondientes
  + El modelo regresa el resultado
* El controlador llama a la vista, enviándole los datos procesados del modelo
  + La vista presenta los datos
* El controlador devuelve la vista al servidor
* El servidor presenta el resultado al cliente

**Caso**

Para implementar el MVC es imprescindible crear una estructura de ficheros parecida a esta:



**Estructura de directorios MVC**

Veamos un ejemplo típico del uso del MVC con PHP. Para comprenderlo sería ideal que supieras PHP Orientado a objetos y cómo funciona la extensión MySQLi

**# index.php**

<?php

require\_once("db/db.php");

require\_once("controllers/personas\_controller.php");

?>

**# db.php**

<?php

class Conectar{

public static function conexion(){

$conexion=new mysqli("localhost", "root", "", "mvc");

$conexion->query("SET NAMES 'utf8'");

return $conexion;

}

}

?>

**# model/personas\_model.php**

<?php

class personas\_model{

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;

}

}

?>

**# contoller/personas\_controller.php**

<?php

//Llamada al modelo

require\_once("models/personas\_model.php");

$per=new personas\_model();

$datos=$per->get\_personas();

//Llamada a la vista

require\_once("views/personas\_view.phtml");

?>

\* El controlador debe tener siempre esta estructura llamada al modelo y debajo a la vista, si hubiera mas modelos y vistas se sigue haciendo así con todos.

**# view/personas\_view.phtml**

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8" />

<title>Personas</title>

</head>

<body>

<?php

foreach ($datos as $dato) {

echo $dato["nombre"]."<br/>";

}

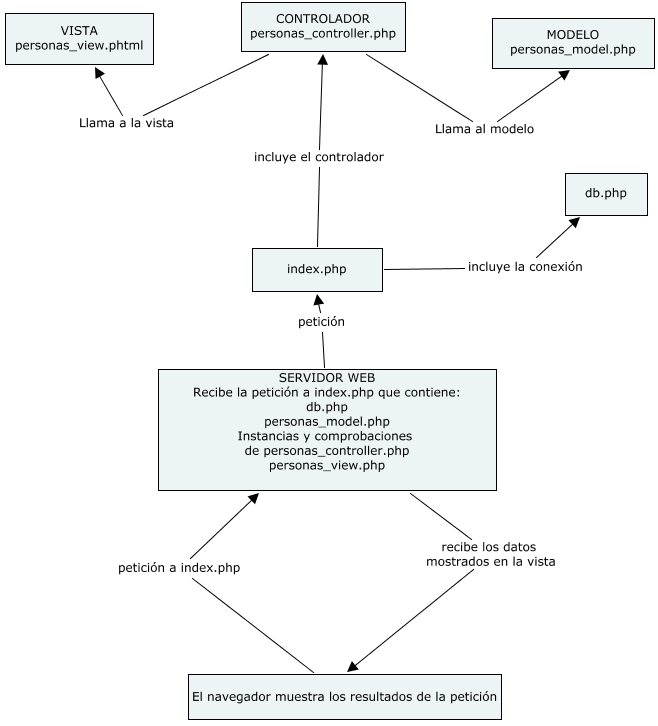
?>

</body>

</html>

Los ficheros de la vista según el estándar de Zend Framework debemos usar .phtml, pero se podría sin ningún problema usar la extensión .php

Muchos dicen que es recomendable usar CamelCase en los nombres de los ficheros y las clases, a efectos prácticos da igual usarlo que no, incluso algunos frameworks como Codeigniter nos sugieren que usemos nombres como “wellcome\_model” por eso no he utilizado CamelCase. Si puedes y quieres abusa del CamelCase, porque así lo dicen los estándares.



**Conclusión**

MVC es un patrón de diseño enfocado a separar las responsabilidades dentro de nuestra aplicación y es muy utilizado en la web por su enfoque y las ventajas que ofrece con respecto a algunas otras formas o patrones de desarrollo de aplicaciones web. Puedes encontrar frameworks prácticamente para cualquier lenguaje web, por ejemplo:

* ASP.NET MVC (C# , VBasic)
* Laravel (PHP)
* django (Python)
* Ruby on Rails

**Recomendaciones**

Aunque los patrones de diseño son muy útiles en el desarrollo de software, no siempre es necesario usarlos y eso no significa que la aplicación esté “mal diseñada”, algunas veces no son necesarios y mucho menos obligatorios, a veces abusar de los patrones de diseño puede traer más problemas de los que resuelve.

**Bibliografía**

1. [Simple Example of MVC (Model View Controller) Design Pattern for Abstraction](http://www.codeproject.com/Articles/25057/Simple-Example-of-MVC-Model-View-Controller-Design)
2. [Best MVC Practices](http://www.yiiframework.com/doc/guide/1.1/en/basics.best-practices)
3. <http://c2.com/cgi/wiki?ModelViewControllerHistory> Historia del Modelo-Vista-Controlador
4. [Notes and Historical documents](http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html) from Trygve Reenskaug, inventor of MVC.
5. "A note on DynaBook requirements", Trygve Reenskaug, 22 March 1979, [SysReq.pdf](http://folk.uio.no/trygver/1979/sysreq/SysReq.pdf).
6. [[http://web.archive.org/web/http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html](http://web.archive.org/web/http:/st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html) How to use Model-View-Controller (MVC)
7. Krasner, Glenn E.; Stephen T. Pope (Aug/Sep de 1988). [«A cookbook for using the model-view controller user interface paradigm in Smalltalk-80»](http://dl.acm.org/citation.cfm?id=50757.50759).[*The JOT*](https://es.wikipedia.org/w/index.php?title=The_Journal_of_Object_Technology&action=edit&redlink=1) (SIGS Publications). Also published as "[A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System](http://web.archive.org/web/http:/www.itu.dk/courses/VOP/E2005/VOP2005E/8_mvc_krasner_and_pope.pdf)" (Report), ParcPlace Systems; Retrieved 2012-06-05.
8. <http://msdn.microsoft.com/en-us/library/ff649643.aspx>
9. [The evolution of MVC and other UI architectures](http://martinfowler.com/eaaDev/uiArchs.html) from Martin Fowler.
10. [How to Use Model–View–Controller (MVC)](http://web.archive.org/web/http:/st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html).
11. Leff, Avraham; James T. Rayfield (September 2001). «Web-Application Development Using the Model/View/Controller Design Pattern». IEEE Enterprise Distributed Object Computing Conference. pp. 118–127.
12. [MCV en javascript con AJAX y jQuery](http://www.cristalab.com/tutoriales/mvc-en-javascript-con-ajax-y-jquery-parte-2-controladores-c95719l/)