TEMA 13

MVC

Modelo

Vista

Controlador

Parte 1: Introducción a MVC

MVC

Separation of concerns

(Concern = Asuntos, responsabilidades, incumbencia)

¿Qué es MVC?

MVC es un patrón de diseño para el desarrollo de software.

- No es algo específico de PHP. Es algo genérico del desarrollo del software. Es usado en aplicaciones con interfaces de usuario.
- Popularizado especialmente en el desarrollo web, pero lleva mucho tiempo usándose con java.
- Usado por la mayoría de los frameworks como laravel y symfony.

Ojo! No necesitamos un framework para poder desarrollar siguiendo el patrón MVC.

¿Por qué MVC?

Porque está comprobado que aporta ventajas en el desarrollo del software…

…y sobre todo su ciclo de vida. Por ejemplo:

- Escalabilidad.
- Modularidad.
- Versatilidad.
- Trabajo en equipo.
- Mantenimiento.

Antes de MVC

"Spaghetti code"

- Escritura del código de manera lineal, intrincado, difícil de seguir.
- Poca reutilización del código.
- Poca separación del código por responsabilidades.

«Spaghetti».
Publicado bajo la licencia CC BY-SA 3.0 vía Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Spaghetti.jpg#/media/File:Spaghetti.jpg.

Antes de MVC

"Spaghetti code"

Se ha comprobado que produce código con difícil mantenimiento, poco escalable y muy acoplado.

ATENCIÓN: El desarrollo de un software lo podemos terminar en 6 meses, pero probablemente lo vamos a estar manteniendo toda la vida.

«Spaghetti».
Publicado bajo la licencia CC BY-SA 3.0 vía Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Spaghetti.jpg#/media/File:Spaghetti.jpg.

Separación de responsabilidades

Este concepto lo traemos al mundo de la informática y la arquitectura del software, pero existe y se aplica en muchos otros campos.

- Fábrica, cadena de montaje.
- Restaurante.
- Equipo de Fútbol.
- Institutos.

En todas estas organizaciones cada una de las personas o grupos de ellas se encarga de tareas específicas, sin entrometerse en el trabajo del otro.

Separación de responsabilidades

Nos trae diversos beneficios, deseables en las organizaciones:

- **Claridad.** Somos capaces de entender mejor cada una de las partes del sistema.
- **Escalabilidad.** Somos capaces de crecer mejor.
- **Desacoplamiento.** Si una de las partes cambia, no afecta demasiado al resto de partes del sistema.

Programación por capas

Dividimos el código de nuestro software en diversas partes, atendiendo a su responsabilidad.

Cada "capa" se encarga de hacer una parte concreta del trabajo de la aplicación.

Obtenemos los mismos beneficios:

Claridad, escalabilidad, desacoplamiento.

Tendremos que "pensar" más cuando estemos desarrollando, pero notaremos la mejoría en el mantenimiento.

MVC al detalle

Modelo - Vista - Controlador

Son las capas que propone este patrón de diseño del software. Podemos encontrar otros patrones con otras separaciones, pero éste es el más extendido.

"No hay balas de plata"

Es decir, este patrón no es la panacea. De hecho hay variaciones al mismo en populares herramientas de desarrollo de software.

- **Modelo**

  Trabaja con los datos de la aplicación.

- **Vista**

  Representa la información y la interfaz de usuario.

- **Controlador**

  Coordina la aplicación, trabaja con los modelos y las vistas.

  Los modelos y las vistas no se conocen entre sí.

Modelo

Mantiene los datos de la aplicación.

- Conoce la estructura de los datos.
- Define las cosas que se pueden y no pueden hacerse con los datos.
- Conoce y aplica las reglas que deben seguirse para el tratamiento de los datos (lo que comúnmente se llama “lógica de negocio”).

El modelo conoce todas esas operaciones con los datos, las define y es capaz de procesarlas, pero no sabe cuando debe hacerlo. Alguien le tiene que ordenar que las haga.

Vista

Es la encargada de presentar la información y los controles para interactuar con la aplicación.

- Toda salida (output) debe realizarse desde una vista.
- Debemos proporcionarle qué datos se deben mostrar. La vista sólo sabe cómo debe mostrarlos.

Cada una de las pantallas de la aplicación podría corresponder con una vista. La vista escribe esa pantalla, representando la información necesaria. La vista no sabe de donde se han sacado los datos para rellenar esa pantalla, sólo sabe de qué manera deben representarse.

Controlador

Se encarga de comunicar los modelos y las vistas.

Se coloca entre medias de los modelos y las vistas para coordinar las acciones a realizar en una aplicación. El controlador sabe cuando pedir a los modelos hacer las operaciones y también conoce y acciona a las vistas que deben representar las pantallas de la aplicación.

Controlador

- Es el encargado de realizar las acciones necesarias para desempeñar las tareas de una aplicación.
- Interactúa con los modelos y las vistas.
- Gobiernan la aplicación.

El controlador es capaz de saber a qué modelos debe llamar para procesar los datos y sabe a qué vistas debe invocar para producir la salida.

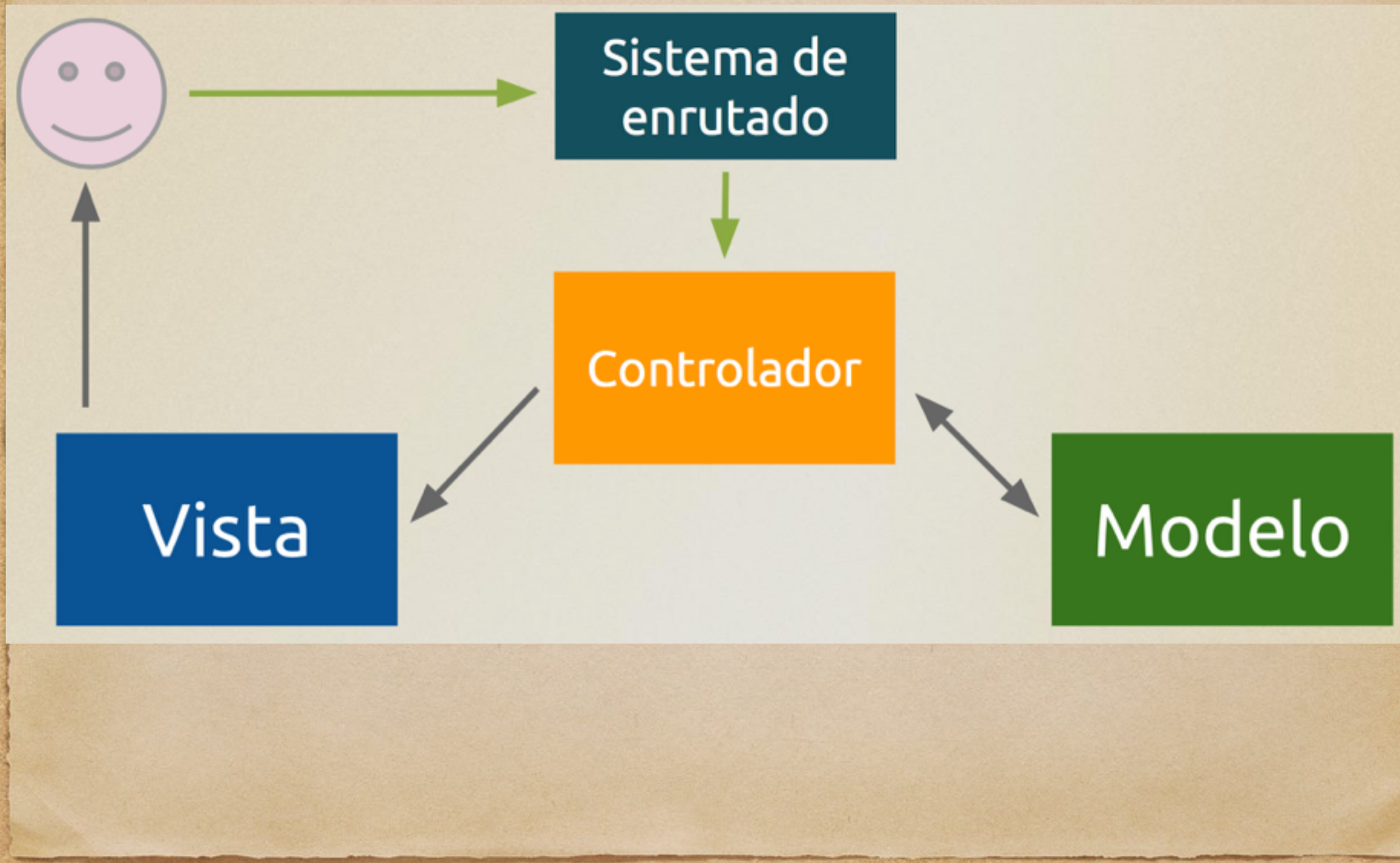
Por eso se dice a veces que en él reside la "**lógica de la aplicación**".

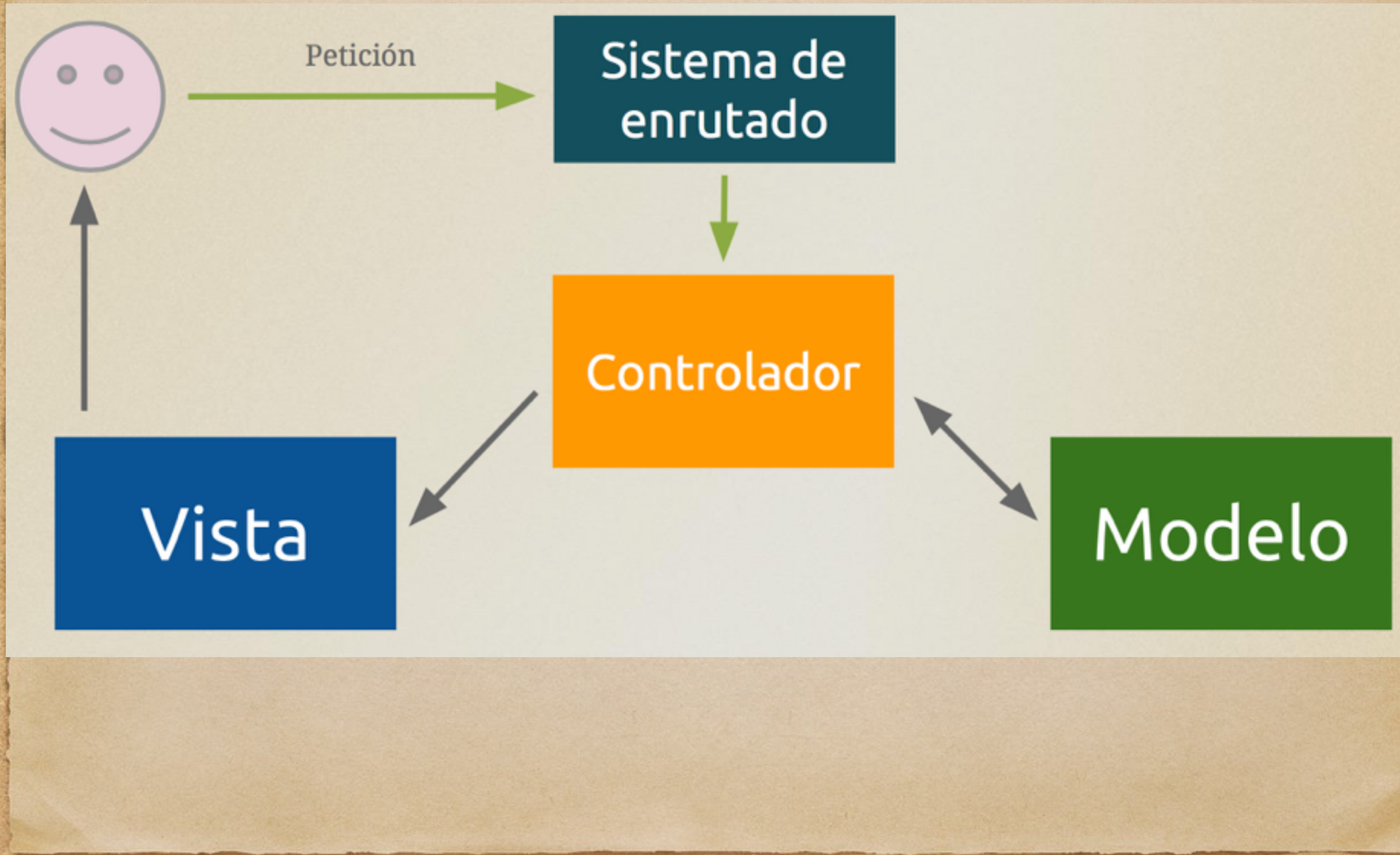
En el contexto de una aplicación web:

Modelo. Realizará las operaciones con la base de datos, consultas, actualizaciones, inserciones. Validarán si aquellos datos que se van a insertar son correctos y válidos para el negocio. OBS: Los datos pueden venir de cualquier lugar.

Vista. Escribe el HTML para mostrar los datos. Será típicamente HTML, pero también podremos tener vistas que generan salidas en otros formatos (como por ejemplo en PDF).

Controlador. Recibirá las peticiones por medio de URL u operaciones POST, y realizará las operaciones necesarias para poder procesar esas peticiones. Desde un controlador puede ser necesario llamar a más de un modelo o más de una vista.


Sistema de enrutado
Controlador
Vista
Modelo


Petición
Sistema de enrutado
Controlador
Vista
Modelo


Petición
Sistema de enrutado
Se invoca al controlador adecuado para procesar esa petición
Controlador
Vista
Modelo


Petición
Sistema de enrutado
Se invoca al controlador adecuado para procesar esa petición
Controlador
El controlador pide datos al modelo adecuado y el modelo los devuelve
Vista
Modelo


Petición
Sistema de enrutado
Se invoca al controlador adecuado para procesar esa petición
Controlador
El controlador pide datos al modelo adecuado y el modelo los devuelve
Vista
El controlador usa la vista adecuada para representar la información, enviando los datos necesarios.
Modelo


Petición
Sistema de enrutado
Se invoca al controlador adecuado para procesar esa petición
La vista es la que envía la salida al usuario
Controlador
El controlador pide datos al modelo adecuado y el modelo los devuelve
Vista
El controlador usa la vista adecuada para representar la información, enviando los datos necesarios.
Modelo

Sistema de Enrutado

Todo comienza con una solicitud de un cliente al servidor solicitando una página:

El sistema de enrutado (también llamado controlador frontal) permite seleccionar el controlador adecuado para cada dirección.

example.com/motor/resultados/gp-singapur

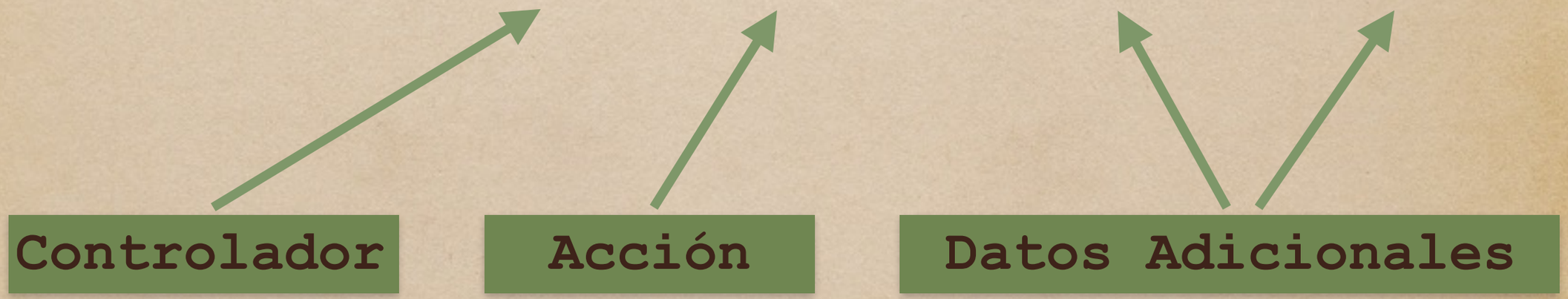
*URL amigable a buscadores

Sistema de Enrutado

Todo comienza con una solicitud de un cliente al servidor solicitando una página:

El sistema de enrutado (también llamado controlador frontal) permite seleccionar el controlador adecuado para cada dirección.

example.com/motor/resultados/gp-singapur/q1



Sistema de enrutado

Todas las URL se suelen procesar por el mismo archivo.

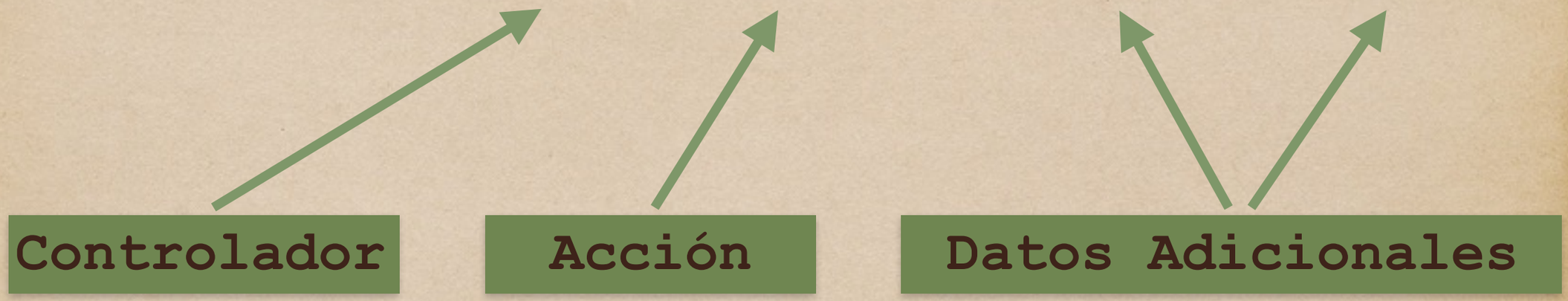
En **index.php** la URL se descompone y se analiza para dirigir la solicitud hacia el **controlador** adecuado, ejecutar la **acción** solicitada en él, enviando los **parámetros** adecuados.

Sistema de enrutado

- No necesitamos crear físicamente archivos para procesar las solicitudes.
- Tenemos un único lugar donde pasa el código de cualquier solicitud y podemos hacer cosas globales a toda la aplicación, como variables de aplicación, objetos, etc.
- Somos capaces de gestionar los errores de la aplicación desde PHP, como cuando nos solicitan páginas que no existen.

Sistema de Enrutado

example.com/motor/resultados/gp-singapur/q1



En términos de código se implementa con…

Clase Método Parámetros del Método