Unidad 7.

MVC

Patrones de diseño de software

En ingeniería del software, como en cualquier otra ingeniería, existen una serie de soluciones estandarizadas que se ajustan sorprendentemente bien a una enorme variedad de situaciones diferentes. Estas soluciones se denominan patrones de software o patrones de diseño.

Patrones de diseño

Los patrones de software son *soluciones comprobadas a problemas comunes en el desarrollo de software*. La arquitectura MVC, de hecho, es un patrón, porque se ha probado infinidad de veces y se adapta a la perfección a multitud de problemas diferentes.

Para que un patrón pueda considerarse tal cosa, tiene que cumplir estas condiciones:

- Debe haber sido comprobado en otros sistemas.
- Debe ser fácilmente reutilizable.
- Debe ser aplicable a diferentes circunstancias.
- Debe estar bien documentado.

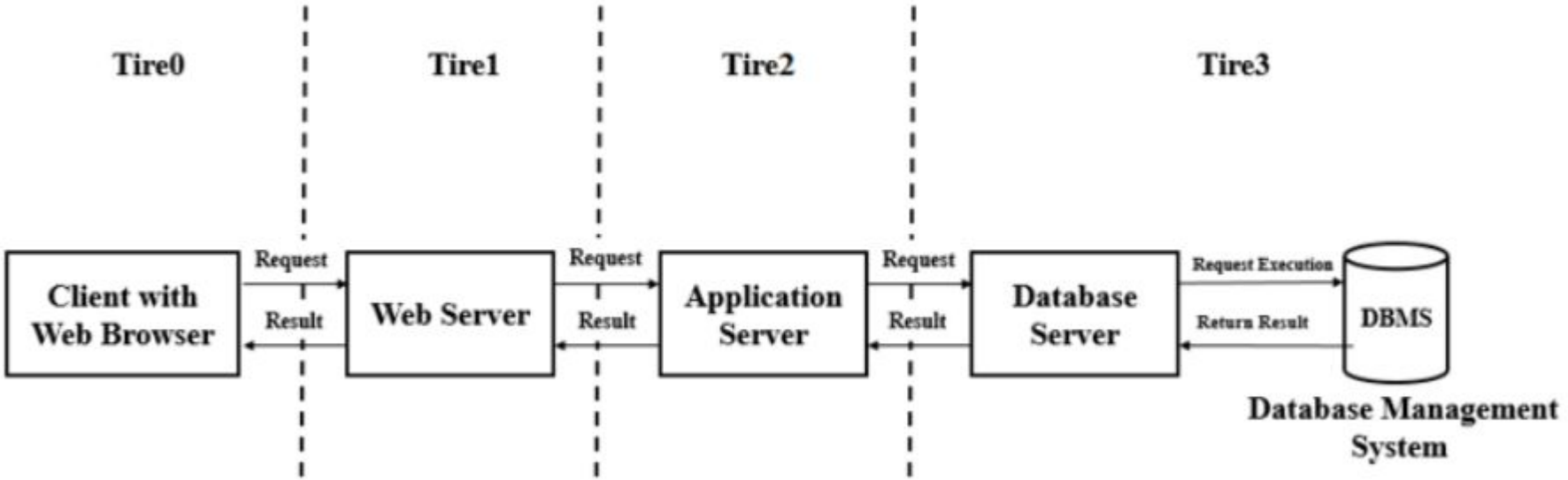
Tipos de patrones

Dependiendo del grado de abstracción del patrón, existen patrones de diverso tipo:

- De arquitectura
- De diseño
- De creación de objetos
- De estructura de clases
- De comportamiento
- De dialectos
- De interacción o interfaz de usuario
- De análisis
- De dominio

Arquitectura de una aplicación web

Usamos el término *“arquitectura de una aplicación”* para referirnos a dos cosas distintas: la *arquitectura física* y la *arquitectura lógica*.



Arquitectura lógica

La arquitectura de una aplicación también puede referirse a sus capas (*layers* en inglés) lógicas. Es decir, a las capas de software que nosotros, los desarrolladores/as, creamos.

La idea es fragmentar nuestra aplicación en capas de niveles de abstracción cada vez mayor. En un extremo, la capa más abstracta es la que interacciona con el usuario: ahí se implementará nuestro interfaz de usuario, o lo que en aplicaciones web se llama ***front-end***.

En el otro extremo, la capa menos abstracta es la que está en contacto con el **hardware** de la máquina. Bueno, en el caso de una aplicación web, no hay contacto directo con el hardware, sino con otras capas aún menos abstractas que están fuera de nuestra aplicación, como el sistema operativo, el servidor web o el gestor de bases de datos. Esas capas externas a nuestra aplicación son las que, realmente, interaccionan con el hardware en última instancia.

Ventajas de las arquitecturas multicapa

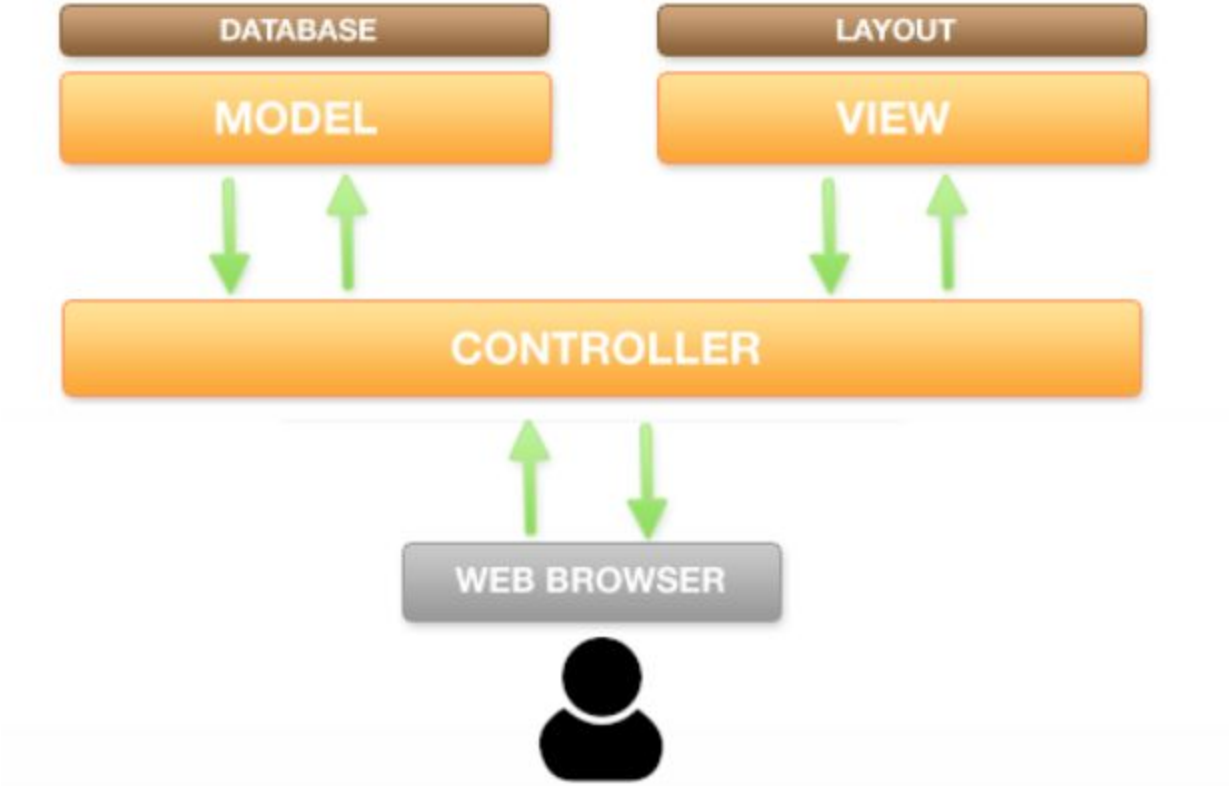
Las arquitecturas multicapa permiten varias cosas que no pueden hacerse con los códigos monolíticos. Entre otras:

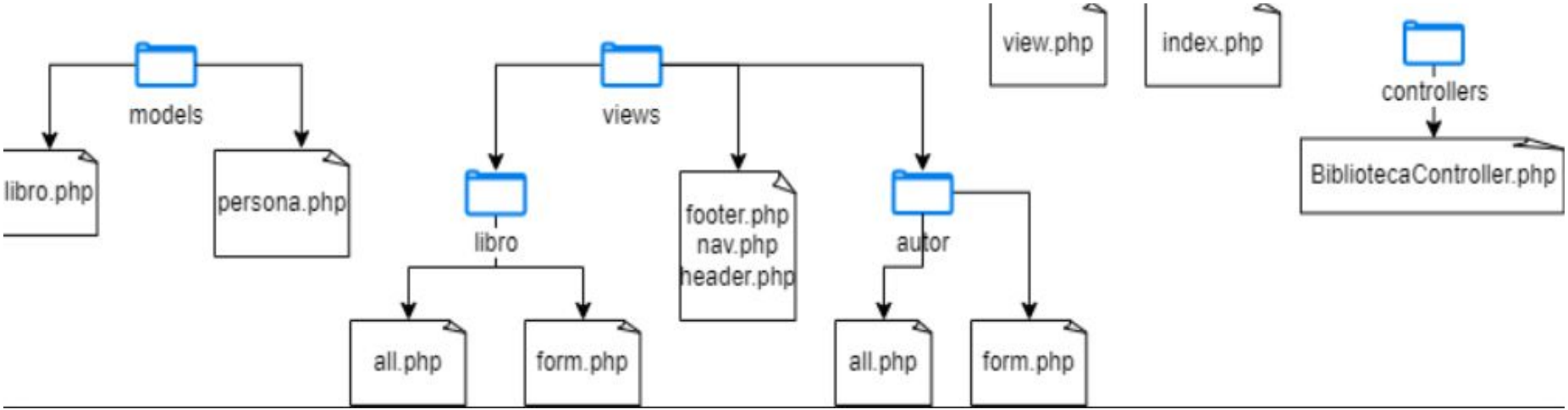
- Desarrollar en paralelo cada capa (mayor rapidez de desarrollo).
- Aplicaciones más robustas gracias al encapsulamiento. ¿Te suena? ¡Programación orientada a objetos! Cada capa se implementa en una clase, y cada clase hace su trabajo sin importunar a las demás y sin preocuparse por cómo funcionan las otras internamente.
- El matenimiento es más sencillo.
- Más flexibilidad para añadir módulos.
- Más seguridad, al poder aislar (relativamente) cada capa del resto.
- Mejor escalabilidad: es más fácil hacer crecer al sistema.
- Mejor rendimiento (aunque esto podría discutirse: puedes hacer un sistema multicapa con un rendimiento desastroso y un sistema monolítico que vaya como un tiro. Pero, en general, es más fácil mejorar el rendimiento trabajando en cada capa por separado).
- Es más fácil hacer el control de calidad, incluyendo la fase de pruebas.

Modelo Vista Controlador

El Modelo Vista Controlador (Model View Controller) - también conocido por sus siglas MVC - es un patrón de diseño de software que separa una aplicación en tres grandes bloques: el acceso a datos, la interfaz de usuario y la lógica de negocio.

La manera de implementar este patrón no es única. Hay muchas aproximaciones posibles, se pueden encontrar implementaciones diferentes en otros manuales.


DATABASE
LAYOUT
MODEL
VIEW
CONTROLLER
WEB BROWSER


models
libro.php
persona.php
views
libro
all.php
form.php
footer.php
nav.php
header.php
autor
all.php
form.php
view.php
index.php
controllers
BibliotecaController.php