Darle estructura a nuestro proyecto

Programación web 2 - 1C-2023

2. Detectando smells

Analicemos una porción de código:

➔ **¿Qué nos huele feo?**

➔ **¿Qué principios viola?**

➔ **¿Éste problema se repite?**

Notas

- DRY: copy&paste
- Single responsability (HTML y PHP)
- SLAP: Niveles de abstracción
- Seguridad (SQL)
- Módulos/Funciones/Clases

Patrones de diseño

Basados en conceptos de arquitectura

Llevados a software por Gang of Four (GoF)

¿Qué es un patrón de diseño?

Los patrones de diseño (design patterns) son **soluciones habituales** a **problemas comunes** en el diseño de software.

Cada patrón es como un plano que se puede personalizar para resolver un **problema de diseño particular (contexto)** de tu código.

https://refactoring.guru/es/design-patterns

1. Clasificación de patrones

- **Los patrones creacionales** proporcionan mecanismos de creación de objetos que incrementan la flexibilidad y la reutilización de código existente.

- **Los patrones estructurales** explican cómo ensamblar objetos y clases en estructuras más grandes a la vez que se mantiene la flexibilidad y eficiencia de la estructura.

- **Los patrones de comportamiento** se encargan de una comunicación efectiva y la asignación de responsabilidades entre objetos.

¿Qué patrones conocemos?

Notas

-

-

-

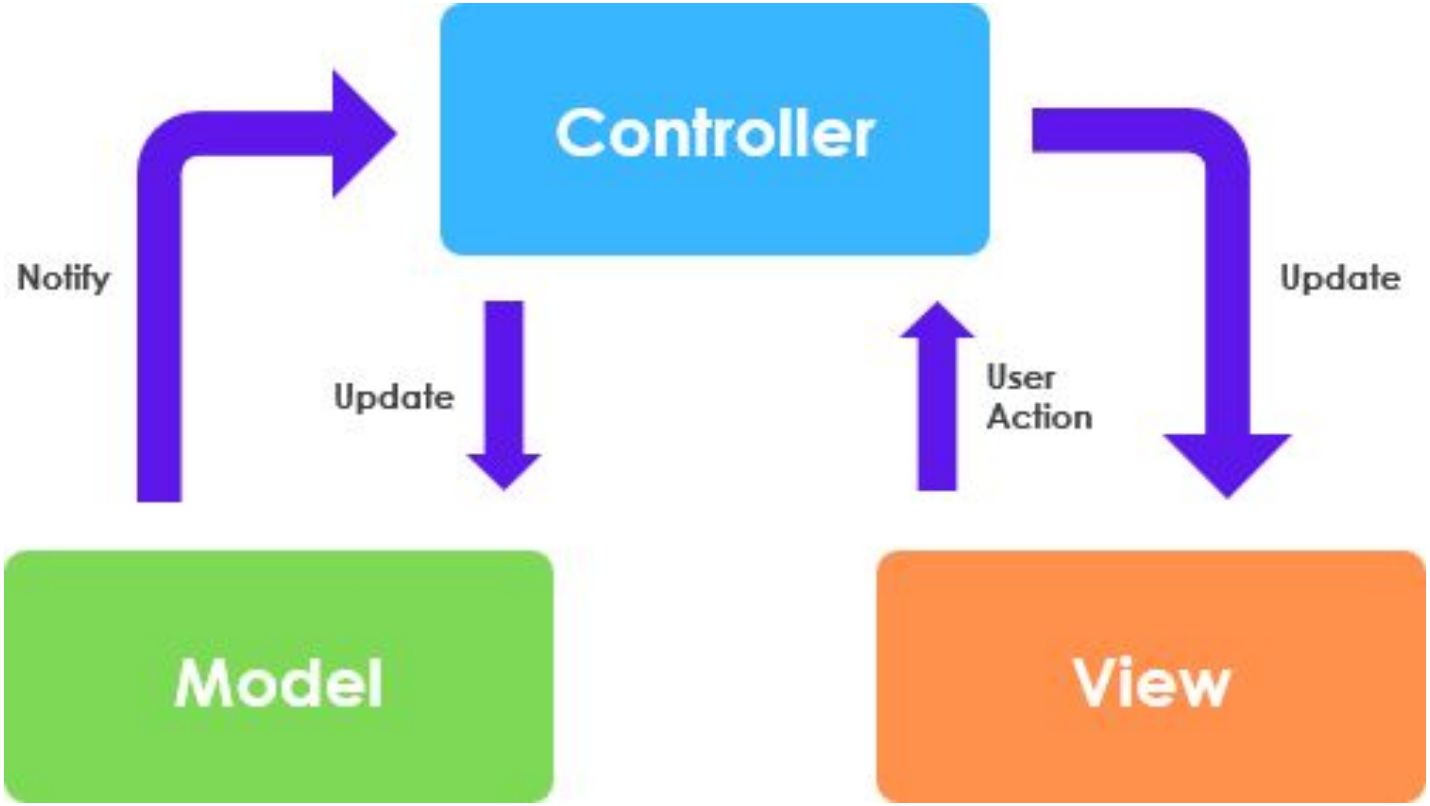
-

¿MVC es el único patrón estructural para organizar el proyecto?

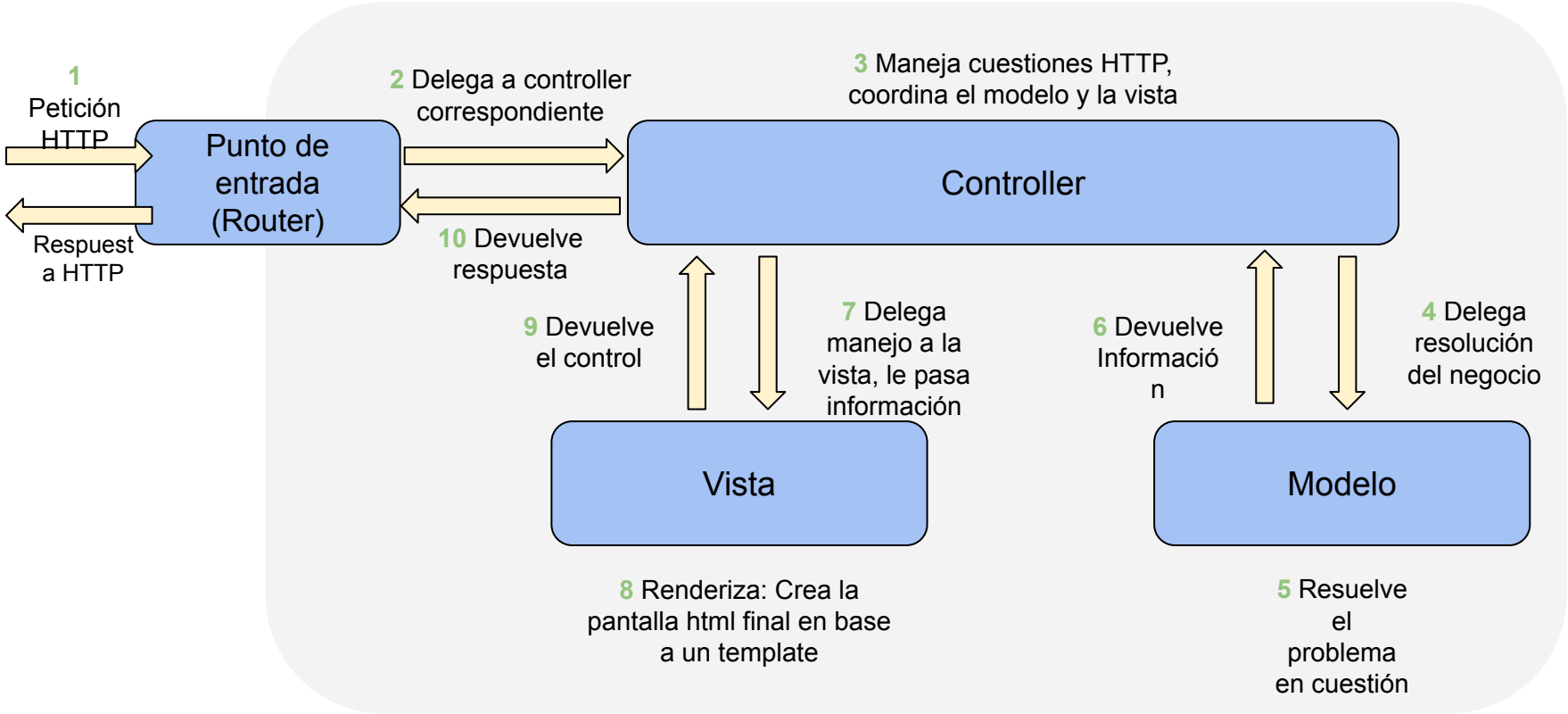
¡No! Existen varias alternativas

Comprender una nos hará más fácil entender el resto: MVP, MVVM

¿Qué nos cuentan de MVC?



Cómo aplicar MVC en arquitectura web



Cómo aplicar MVC en arquitectura web

Aclaraciones

Controller:

- Session
- Get
- Redirect

Modelo:

- Procesar datos
- Tomar decisiones de negocio
- Acceder a los datos

Vista:

- Solamente mostrar datos
- Tener flags para saber cuando mostrarlos
- Idealmente, no conocer de lenguaje backend (php)

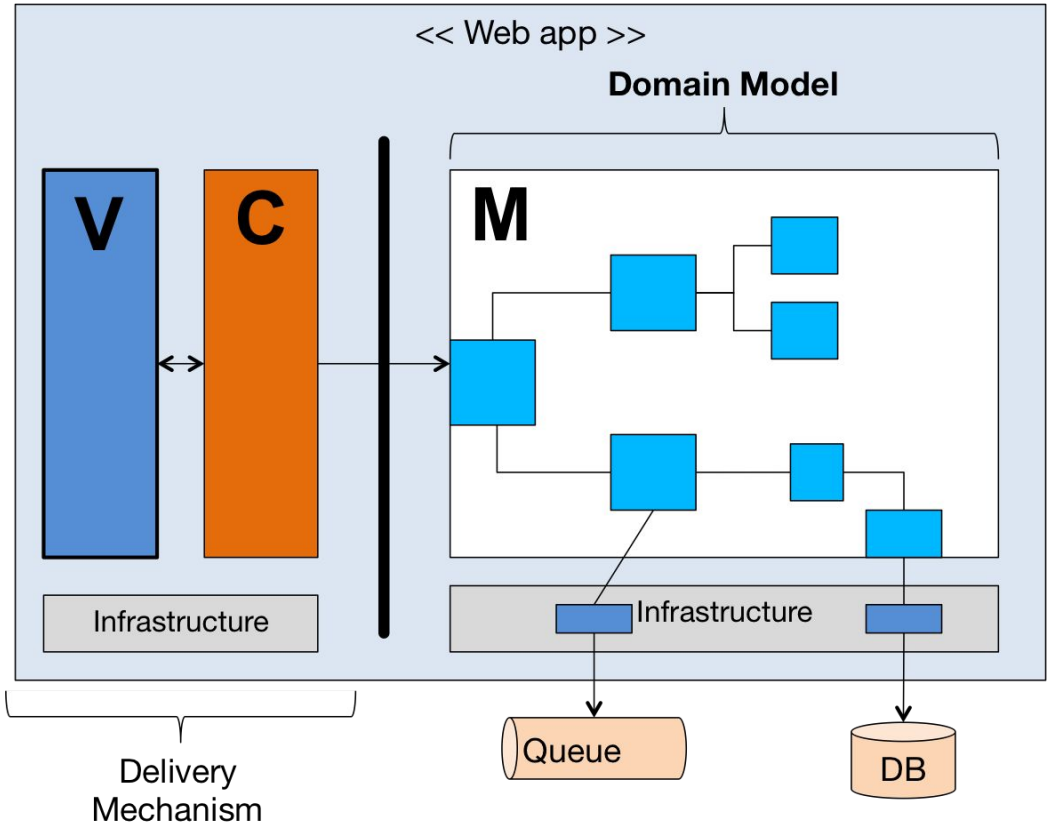
¿MVC resuelve todo?

MVC es sólo un **mecanismo de delivery** para organizar nuestro proyecto

El modelo es quien nos da de comer: Dentro de nuestro modelo, debemos organizar nuestro proyecto en sí.

Antipatrones: Modelo no es una tabla de base de datos, y repositorio no es el modelo

Link de referencia



¡Manos a la obra!

Paso a paso

Etapa 1

- ➔ Header y footer
- ➔ Base de datos: Acceso a base de datos (getAll, llevar a objeto)
- ➔ Config: Configuraciones en código (ini)
- ➔ Todo al index: Switch donde sólo varía el contenido

Paso a paso

Etapa 2

- MVC: Ponerle nombre a las cosas
- Configuration: Instanciar todo en un sólo lugar
- Packaging/Scaffolding: En criollo carpetas o estructura
- Mustache: Vista no tiene que conocer php
- Router: evitar switch enorme

- .htaccess: Urls no son claras (problemas de permisos y acceso a archivos)
- Otros helpers: Logs, parser-url, router, database, render, etc

Preguntas

¡Gracias!