Ajuntament de Barcelona

Patrones de arquitectura y diseño Parte I

MVC, MVVM y Patron Repository

Abril 2019

¿Que son los patrones?

Hay 2 Tipos:

- Patrón de arquitectura: Parte alta del diseño referente a estructura y comunicación del sistema a crear (MVC, MVVM, Cliente-Servidor, Patrón capes...)

- Patrón de diseño: Se centra en partes específicas del programa centralizando la responsabilidad y función de esa pequeña parte (Patrón Repository, Patrón REST...)

Patrones de diseño en detalle (depende del area que afecta):
https://www.journaldev.com/1827/java-design-patterns-example-tutorial

¿Por que usar patrones?

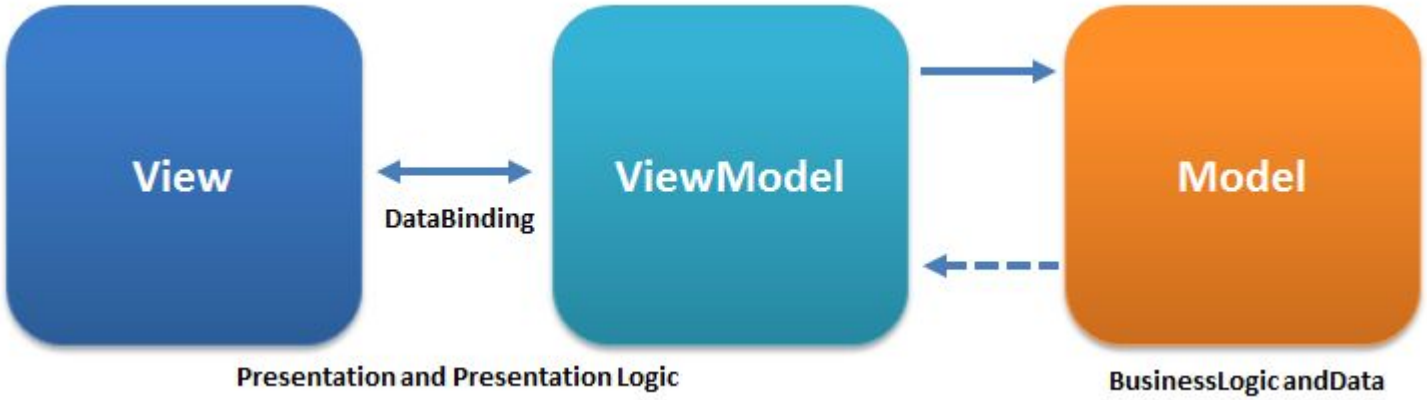
- Patrones de arquitectura:
  - Organización estructural unificado
  - Fácil definición de conceptos pre proyecto.

- Patrones de diseño:
  - Reutilización de componentes (código)
  - Reutilización de soluciones ya existentes (no reinventar la rueda)
  - Estandarización de funciones y vocabulario

Patrón de arquitectura

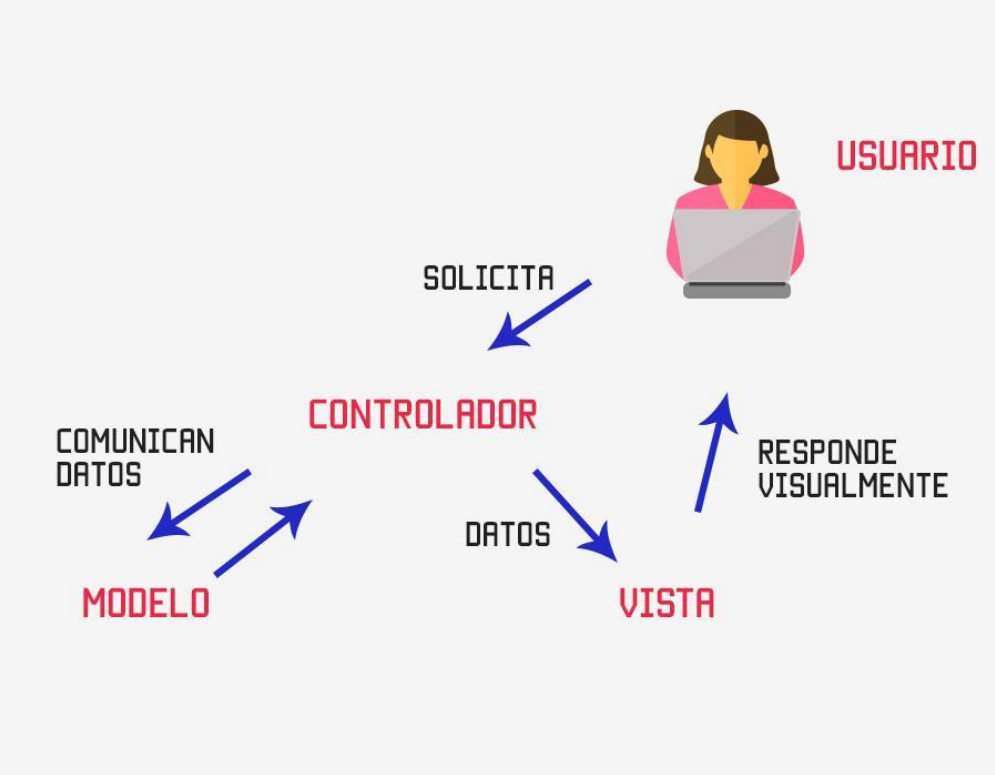
MVC y MVVM

MVVM (Modelo-Vista-Vista-Modelo)



- Uso en aumento (Angular 2 y Microsoft por ejemplo)
- Característico por desacoplar la interfaz de usuario con la lógica mediante utilización de “Bindings”
- Consta de tres partes: Modelo de datos, la vista y el modelo de vista (“un controller” con muchas comitas. Enlaces directos)

MVC (Modelo Vista Controlador)



- 3 componentes separan la lógica de la aplicación de la lógica de la vista.
- Ampliamente usada en todos los entornos (tanto empresarial, componentes gráficos, incluso en videojuegos)
- Java, .Net y AngularJS son típicos lenguajes que usan el MVC.

Una pequeña analogía ~~(de internet)~~

En tu televisión puedes ver distintos canales distribuidos por tu proveedor de cable o televisión (que representa al modelo), todos los canales que puedes **ver** son la vista, y tú cambiando de canal, **controlando qué ves** representas al controlador.

Sus componentes

- Modelo: Se encarga de los datos consultando a la base de datos (¡No siempre!) mediante un CRUD o parte de el.

- Controlador: Recibe ordenes del usuario (vista), solicita datos al modelo y comunicarlo a la vista…. Es un controlador….

- Vistas: Es la representación visual de los datos e interfaz gráfica. Único responsable de la visualización de datos y forma de comunicación del usuario.

En ningún momento hay que saltarse la comunicación.

Vista <-> Controlador <-> Modelo

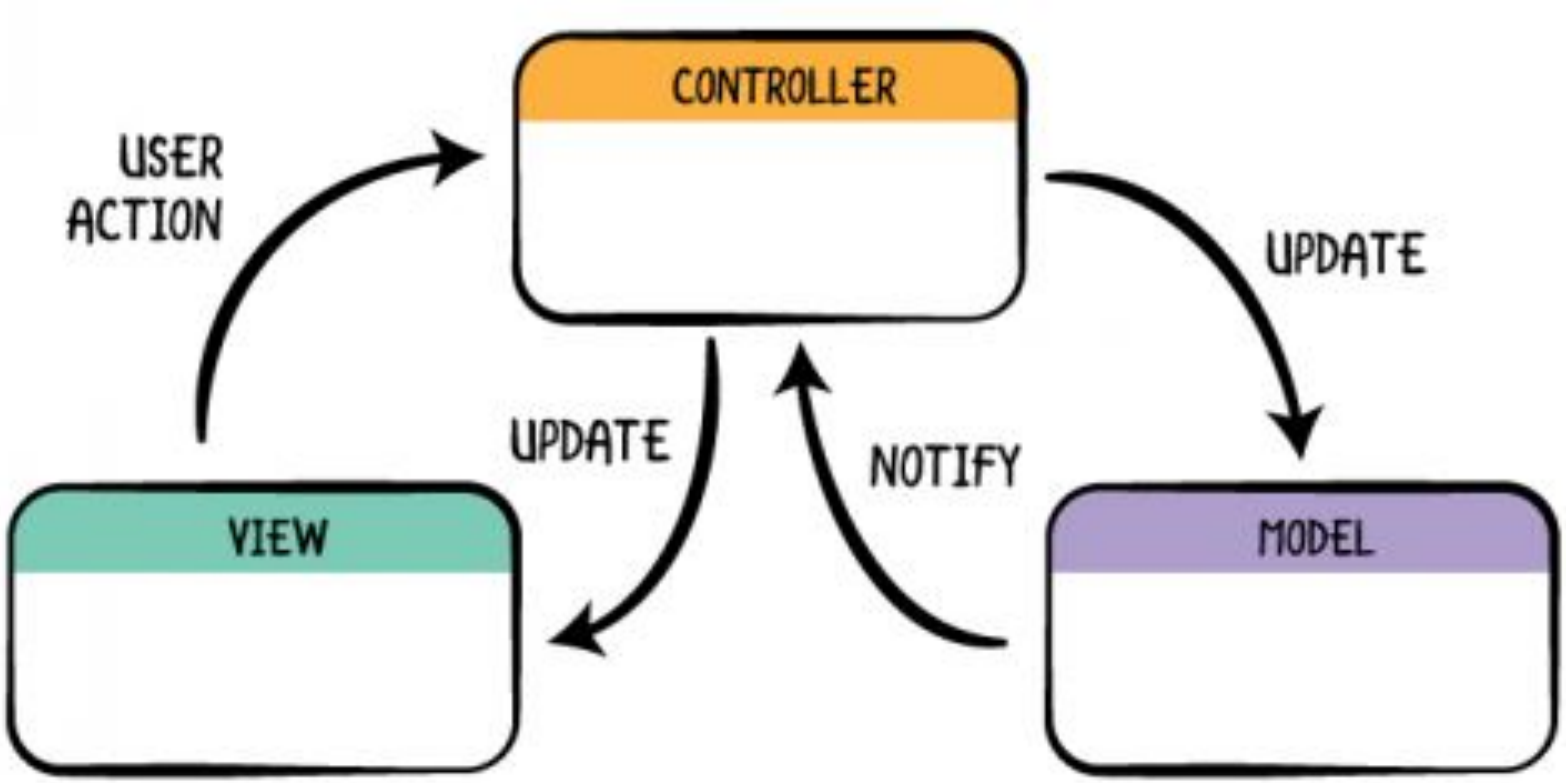
¿Por qué MVC?

- Separar los componentes de la aplicación dependiendo de la responsabilidad
    - Ejemplo: Modificamos la Base de Datos solo debemos modificar el Modelo **que se encarga de los datos.**

- Respeta el principio de responsabilidad única
    - Una parte del código solo debe saber que hace el, no toda la aplicación

Reusabilidad y escalabilidad al alcance de todos.

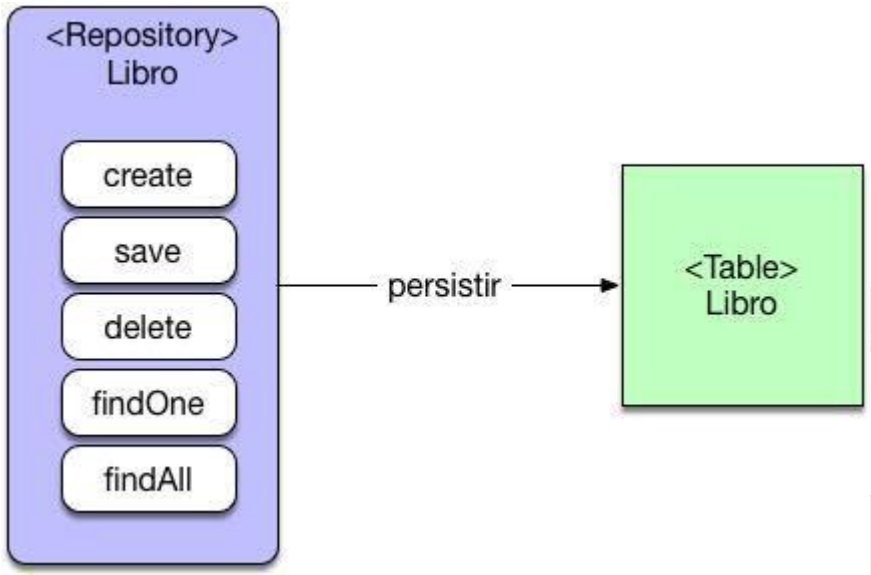
Ejemplo ~~improvisado~~



Patrón de diseño

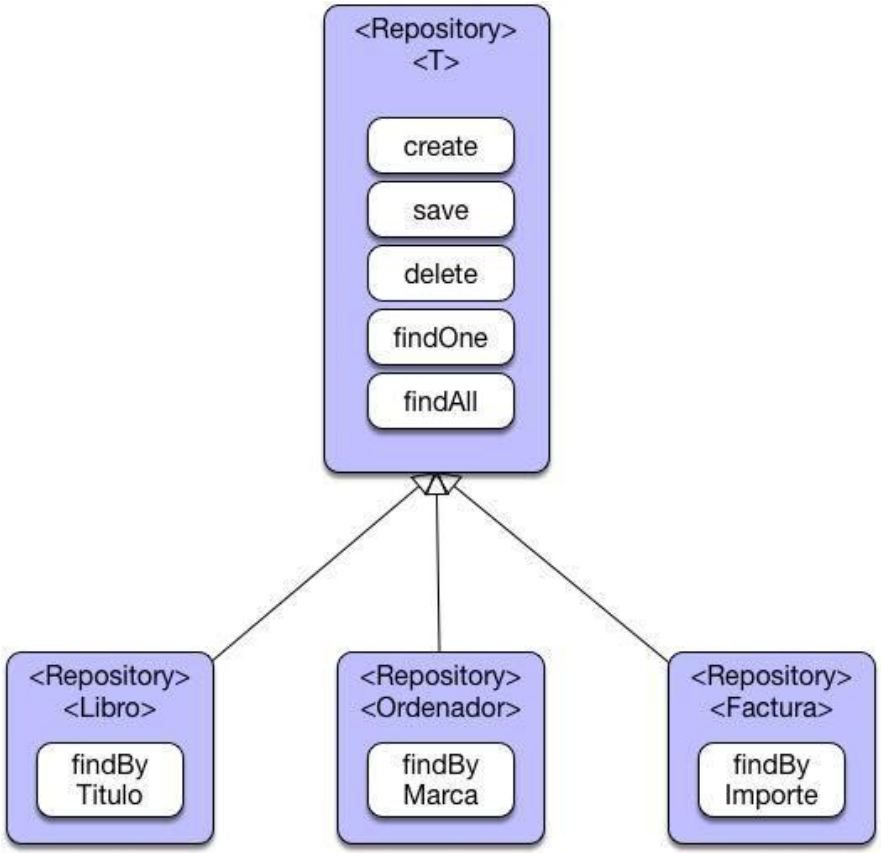
Patrón Repository

Patrón repository (repositorio)



- Su función es especificar como persistir datos en la base de datos (mediante una clase repository)
- Tiene variaciones según necesidades y **entorno/lenguaje**
- Diferentes patrones se pueden aplicar a la vez (mientras no solapen responsabilidad)

Patrón repository generico



- Muchos lenguajes soportan los tipos **Genéricos**
- Reducen la cantidad de código (Serás mas feliz)
- Todos los repositorios extenderán (heredaran) del repositorio Genérico añadiendo las operaciones específicas.

Extra

- Los patrones de diseño se separan según la zona que afectan.
- Existen algunos patrones extendidos (como el repository) pero cada gran proyecto tiene sus propios patrones y puntos clave.
- Por ello hay muchos y de muchos tipos (creativos, estructurales, runtime etc…)

Conclusión

Programad en inglés

Barcelona Activa

barcelona.cat/barcelonactiva

Unió Europea
Fons Europeu
de Desenvolupament Regional