**Patrón Modelo-Vista-Controlador**

El patrón Modelo-Vista Controlador (MVC) es un patrón de arquitectura software que separa los datos y la lógica de negocio (Def. “parte de un sistema que se encarga de codificar las reglas de negocio del mundo real que determinan cómo la información puede ser creada, almacenada y cambiada”) de una aplicación, su representación y el módulo encargado de gestionar los eventos y las comunicaciones. Tiene tres componentes principales:

* **Modelo**: contiene la funcionalidad básica y los datos.
* **Vista**: muestran y recogen información del usuario (la información puede recógela el controlador).
* **Controlador**: media (actúa como puente) entre la vista y el modelo.

Este patrón se basa en ideas como la **reutilización del código** y la **separación de conceptos**. Este patrón surgió a finales de los años 70 y principios de los 80 y en gran medida pretendía aplicar el **Principio de Responsabilidad Única**, es decir, que cada clase o módulo se haga cargo de únicamente una parte de la funcionalidad y que esta responsabilidad esté encapsulada. Con esto conseguimos **desvincular la presentación del resto de la aplicación**. En cuanto a la reutilización del código, una misma vista puede adaptarse mediante una refactorización a otro tipo de datos. El motivo final por el cuál aplicamos este patrón es para facilitar el desarrollo de las aplicaciones. Profundizando algo más en las atribuciones de cada módulo tenemos que:

El **modelo** se encarga de la representación de la información y de gestionar los accesos a dicha información (consultas, actualizaciones). Envía a la vista la información para ser mostrada. Estas peticiones se realizan a través del controlador.

El **controlador** responde a eventos (acciones del usuario) e invoca peticiones al modelo. También puede enviar comandos a la vista asociada si se solicita un cambio en la forma en que se presenta el modelo. Principalmente el controlador funciona de intermediario entre la vista y el controlador. Puede diferir el número de controladores en una aplicación: desde uno por evento (Action Listener) hasta uno por aplicación.

La **vista** se encarga de presentar los datos y la información del modelo en un formato adecuado (fichero de texto, HTML, interfaz con el usuario…).

En los siguientes diagramas se puede ver…

**Tipos modelo pasivo vs activo** (la diferencia es quien actualiza la vista)

Preferimos el modelo pasivo ya que para actualizar la vista en una aplicación web que es una de las grandes utilidades del modelo-vista-controlador, no tiene sentido hacerlo desde el modelo. Es preferible hacerlo desde el controlador una vez el modelo le ha devuelto la información.

(Saca el código del action listener, es decir, los action listener de los botones no tienen el código de qué hacer cuando son pulsados, sino que llaman al controlador que es el que sabe cómo responder.)

**Ventajas:**

* El modelo es independiente del comportamiento de la entrada y de la representen de sus datos en la salida.
* Múltiples vistas para un mismo modelo
* Los cambios o modificaciones de código se pueden hacer de manera independiente en la interfaz y la lógica del programa.
* Centraliza la interacción en las propias vistas y la vista y el modelo.
* Desarrollo simultáneo de la aplicación
* Alta cohesión
* Bajo acoplamiento

**Desventajas:**

* Complejidad
* Navegación dentro del código debido a las numerosas capas
* Excesivamente repetitiva, si se practica mal el mvc podemos la aplicación puede estar sobrecargada en el modelo y el controlador y las vistas pueden tener código repetido o estar completarse simplemente para satisfacer las exigencias del patrón.
* Curva de aprendizaje lenta