**Patron Modelo-Vista-Controlador ( MVC )**

**Concepto**

El patrón Modelo – Vista – Controlador fue inventado en el contexto de Smalltak para realizar una separación entre la interfaz gráfica y el código del funcionamiento de una aplicación. Esta idea teórica afectó, de forma importante, a gran parte del código de Smalltalk y fue posteriormente aplicada a los lenguajes que están basados en objetos.

En el paradigma MVC, las entradas del usuario, los modelos del mundo exterior y la retroalimentación visual son explícitamente separados y manejados por tres tipos de objetos, cada uno especializado para un conjunto de tareas específicas.

El objetivo primordial del patrón es dar soporte a los modelos funcionales y mapas mentales de la información relevante para los usuarios, permitiendo un modelo que facilite la consulta y manejo de los mismos. La única manera de construir artefactos manejables es ayudar al usuario a construir modelos del sistema lo cual a la fecha no se ha logrado.

### Modelo

El modelo es un conjunto de clases que representan la información del mundo real que el sistema debe reflejar. Es la parte encargada de representar la lógica de negocio de una aplicación. Así, por ejemplo, un sistema de administración de datos geográficos tendrá un modelo que representara la altura, coordenadas de posición, distancia, etc. sin tomar en cuenta ni la forma en la que esa información va a ser mostrada ni los mecanismos que hacen que esos datos estén dentro del modelo, es decir, sin tener relación con ninguna otra entidad dentro de la aplicación.

#### Modelo de Dominio

Se entiende por modelo de dominio al conjunto de clases definidas a través del análisis de la situación real del problema que queremos solucionar. Si seguimos el ejemplo anterior, sobre datos geográficos, pertenecerían a este dominio las clases; Rio, Montaña , Mar, etc...El modelo del dominio no debería tener relación con nada externo a la información que contiene.

#### Modelo de la aplicación

El modelo de la aplicación es un conjunto de clases que sirven de puente en la relación de las vistas con el modelo de dominio. Tienen conocimiento de las vistas e implementan los mecanismos necesarios para notificar a éstas los cambios que se pudieren dar en el modelo del dominio. El modelo de la aplicación es llamado también coordinador de la aplicacion.

### Vista

Las vistas son las encargadas de la representación de los datos, contenidos en el modelo, al usuario. La relación entre las vistas y el modelo son de muchas a uno, es decir cada vista se asocia a un modelo, pero pueden existir muchas vistas asociadas al mismo modelo. De esta manera, por ejemplo, se puede tener una vista mostrando la hora del sistema como un reloj analógico y otra vista mostrando la misma información como un reloj digital.

La vista solo necesita la información requerida del modelo para realizar un despliegue. Cada vez que se realiza una actuación, que implica una modificación del modelo de dominio, la vista cambia a través de notificaciones generadas por el modelo de la aplicación. Sencillamente, es la representación visual del modelo que redibuja las partes necesarias cuando se produce una modificación del mismo.

### Controlador

El controlador es el encargado de interpretar y dar sentido a las instrucciones que realiza el usuario, realizando actuaciones sobre el modelo. Si se realiza algún cambio, comienza a actuar, tanto si la modificación se produce en una vista o en el modelo. Interactúa con el Modelo a través de una referencia al propio Modelo.

### El patrón MVC y el lenguaje JAVA

El lenguaje de programación Java proporciona soporte para la arquitectura MVC. Provee de dos clases que son las encargadas de realizar las notificaciones de cambios en los estados de los objetos. Se definen a continuación los objetos:

* Observer: Es cualquier objeto que desee ser notificado cuando el estado de otro objeto sea alterado.
* Observable: Es cualquier objeto cuyo estado puede representar interés y sobre el cual otro objeto ha demostrado ese interés.

Estas dos clases no sólo se utilizan en la aplicación del patrón MVC, tienen una utilidad mayor dentro del lenguaje. Serán útiles en cualquier sistema en el que se necesite que algunos objetos sean notificados cuando ocurran cambios en otros objetos.

**Ventajas y desventajas del uso del patrón**

Se tienen muchas ventajas como:

* Sus vistas muestran información actualizada siempre. El programador no debe preocuparse de solicitar que las vistas se actualicen, ya que este proceso es realizado automáticamente por el modelo de la aplicación.
* Cualquier modificación que afecte al dominio, como aumentar métodos o datos contenidos, implica una modificación sólo en el modelo y las interfaces del mismo con las vistas, no todo el mecanismo de comunicación y de actualización entre modelos.
* MVC esta demostrando ser un patrón de diseño bien elaborado pues las aplicaciones que lo implementan presentan una extensibilidad y una mantenibilidad únicas comparadas con otras aplicaciones basadas en otros patrones.

Como desventajas tenemos:

* Para desarrollar una aplicación bajo el patrón de diseño MVC es necesario una mayor dedicación en los tiempos iniciales del desarrollo. Normalmente el patrón exige al programador desarrollar un mayor número de clases que, en otros entornos de desarrollo, no son necesarias. Sin embargo, esta desventaja es muy relativa ya que posteriormente, en la etapa de mantenimiento de la aplicación, una aplicación MVC es mucho más mantenible, extensible y modificable que una aplicación que no lo implementa.
* MVC es un patrón de diseño orientado a objetos por lo que su implementación es sumamente costosa y difícil en lenguajes que no siguen este paradigma.

