Ricardo Montalván Orellano

PATRÓN MVC



El MVC o Modelo-Vista-Controlador es un patrón de arquitectura de software que utilizando 3 componentes (View-Model-Controller) separa la lógica de la aplicación.

**DEDICATORIA**

**A Dios:** por permitirme tener la fuerza para terminar mi carrera.

**A mis padres:** por su esfuerzo en concedermela oportunidad de estudiar y por su constante apoyo a lo largo de mi vida.

**A mis parientes y amigos:** por sus consejos, paciencia y toda la ayuda que me brindaron para concluir mis estudios.

**ÍNDICE**

1. **Capítulo I. Resumen**

1.1Introducción

1. **Capítulo II. Desarrollo teórico**
   1. Ventajas y Desventajas
   2. Modelo
   3. Vista
   4. Controlador
2. **Capítulo III. La Solución**

3.1 Resumen

1. **Capítulo IV. Conclusiones**

4.1 Resumen

1. **Capítulo V. Recomendaciones**

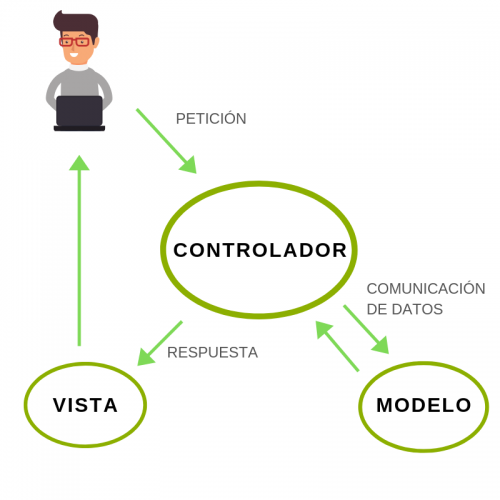
5.1 Resumen

1

**RESUMEN**

**1.1 Introducción**

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 los artefactos manejables es ayudar al usuario a construir modelos del sistema. Pero esto es imposible si el modelo mental no ha sido diseñado dentro del artefacto desde el principio. Intentar adicionar los modelos mentales del usuario cuando ya se a avanzado en el desarrollo puede ser imposible. A continuación un gráfico que resume el patrón.



2

**DESARROLLO TEÓRICO**

**2.1 Ventajas y Desventajas**

Se tienen muchas ventajas como:

* La implementación se realiza en forma modular.
* Sus vistas muestran información actualizada siempre. El programador no debe preocuparse de solicitar que las vistas se actualicen ya que este proceso es actualizado 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.
* Las modificaciones a las vistas no afectan al modelo de dominio, simplemente se domifica la representación de la información, no su tratamiento.
* MVC esta demostrado 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 requiere la existencia de una arquitectura inicial sobre la que se deben construir clases e interfaces para modificar y comunicar los módulos de una aplicación. Esta arquitectura inicial debe incluir, por lo menos, un mecanismo de eventos para poder proporcionar las notificaciones que genera el modelo de aplicación; una clase Modelo, otra clase Vista y una clase Controlador genéricas que realicen todas las tareas de comunicación, notificación y actualización que serán luego transparentes para el desarrollo de la aplicación.
* 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.

**2.2 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.

**2.3 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.

**2.4 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.

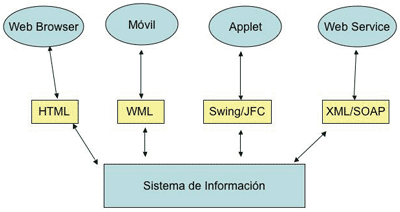
3

**LA SOLUCIÓN**

**3.1 Resumen**

El patrón MVC consiste en separar las diferentes responsabilidades en una aplicación, en elementos de tres tipos:

* Modelos que contengan el estado (datos) y los mecanismos para alterarlo.
* Vistas que desplieguen información o la comuniquen a otros sistemas.
* Controles que administren los eventos que afectan a los modelos y escojan las vistas adecuadas para desplegar los resultados.



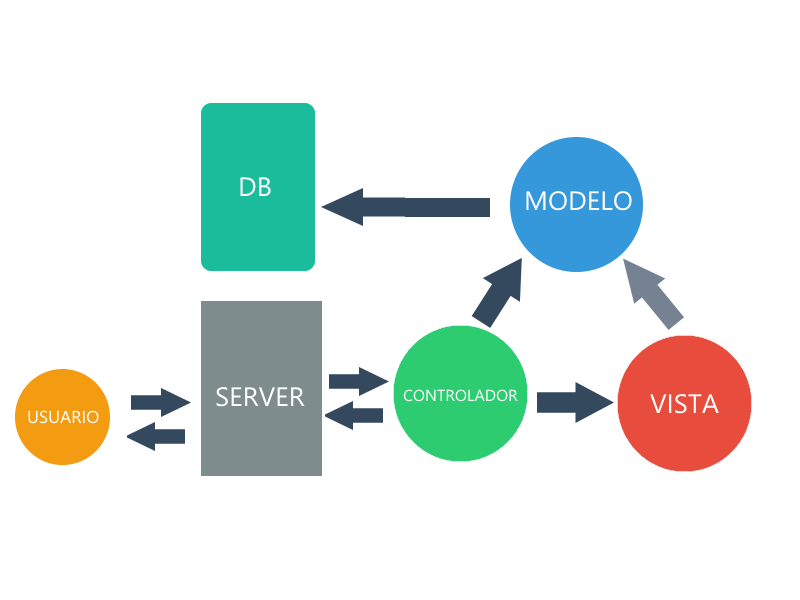
4

**CONCLUSIONES**

**4.1 Resumen**

Un patrón de diseño es una solución reutilizable en problemas con un contexto común en el diseño de software. Podemos conceptualizarlo como una receta, pasos concretos a seguir que nos ayudarán a resolver un problema. No cualquier cosa puede ser considerada un patrón de diseño, puesto que deben cumplir ciertos objetivos o tener ciertas características:

* Reutilizable en distintas circunstancias de diseño.
* Efectivo en problemas similares.
* Permite evitar riesgos que impidan reutilizar código.
* Simplifican la forma de documentar una aplicación.



5

**RECOMENDACIONES**

**5.1 Resumen**

En líneas generales, MVC es una propuesta de diseño de software utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario. Surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos.

MVC es un "invento" que ya tiene varias décadas y fue presentado incluso antes de la aparición de la Web. No obstante, en los últimos años ha ganado mucha fuerza y seguidores gracias a la aparición de numerosos frameworks de desarrollo web que utilizan el patrón MVC como modelo para la arquitectura de las aplicaciones web.

**Nota**: MVC es útil para cualquier desarrollo en el que intervengan interfaces de usuario.

