

**República de Panamá**

**Universidad Interamericana de Panamá**

**Carrera:** Licenciatura en Ingeniería Industrial y de Sistemas

**Materia:** Programación II

**Grupo/ Horario**: Jueves – Jueves 5:45PM – 8:05PM

**Estudiante / Cedula:**

Massiel Vergara 8-896-1960

**Profesor:**

Abdel Martínez

**Investigación**

**Fecha de Entrega:**

14 de Abril de 2016

**Introducción**

Hoy día en cualquier lugar del mundo los que construyen aplicaciones informáticas centran su atención en dos aspectos fundamentales: cómo lograr construir mejores aplicaciones en menos tiempo, y cómo utilizar mayor cantidad de estándares en el diseño de las aplicaciones que permitan mayor reutilización del código y mejores mantenimientos de los sistemas desarrollados.

Teniendo en cuenta el creciente uso de la programación orientada a objeto en la concepción e implementación de este tipo de aplicaciones, y la gran actualidad que tiene el uso de patrones internacionalmente aceptados, se propone en este artículo un análisis somero del patrón Modelo-VistaControlador (MVC).

**Modelo-Vista-Controlador**

* **Modelo**

Contiene el núcleo de la funcionalidad (dominio) de la aplicación.

Encapsula el estado de la aplicación.

No sabe nada / independiente del Controlador y la Vista.

* **Vista**

Es la presentación del Modelo.

Puede acceder al Modelo pero nunca cambiar su estado.

Puede ser notificada cuando hay un cambio de estado en el Modelo.

* **Controlador**

Reacciona a la petición del Cliente, ejecutando la acción adecuada y creando el modelo pertinente

Para entender cómo funciona nuestro patrón Modelo vista controlador, se debe entender la división a través del conjunto de estos tres elementos y como estos componentes se comunican unos con los otros y con otras vistas y controladores externos a el modelo principal. Para ello, es importante saber que el controlador interpreta las entradas del usuario (tanto teclado como el ratón), enviado el mensaje de acción al modelo y a la vista para que se proceda con los cambios que se consideren adecuados

***Comunicación***

El modelo, la vista y el controlador deben comunicarse de una manera estable los unos con los otros, de manera que sea coherente con las iteraciones que el usuario realizara. Como es lógico la comunicación entre la vista y el controlador es bastante básica pues están diseñados para operar juntos, pero los modelos se comunican de una manera diferente, un poco más sutil

***Modelo pasivo***

No es necesario para el modelo hacer ninguna tener alguna disposición a él, simplemente basta con tener en cuenta su existencia. El modelo no tiene ninguna responsabilidad para comunicar los cambios a la vista porque ocurren solo por orden del usuario, por lo que esta función la llevara a cabo el controlador porque será el que interprete las ordenes de este usuario debido a que solo debe comunicar que algo ha cambiado. Por esto, el modelo es se encuentra en modo inconsciente y su participación en este caso es irrisoria.

***Unión del modelo con la vista y el controlador***

Como no todos los modelos pueden ser pasivos, necesitamos algo que comunique al controlador y a la vista, por lo que en este caso, si que necesitamos el modelo, ya que solo este puede llevar a cabo los cambios necesarios al estado actual en el que estos se encuentran.

Al contrario que el modelo, que puede ser asociado a múltiples asociaciones con otras vistas y controladores, cada vista solo puede ser asociada a un único controlador, por lo que han de tener una variable de tipo controler que notificara a la vista cuál es su controlador o modelo asignado. De igual manera, el controlador tiene una variable llamada View que apunta a la vista. De esta manera, pueden enviarse mensajes directos el uno al otro y al mismo tiempo, a su modelo.

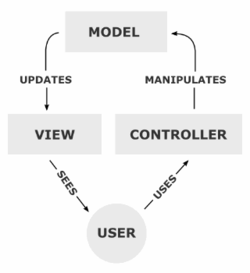
Al final, la vista es quien lleva la responsabilidad de establecer la comunicación entre los elementos de nuestro patrón MVC. Cuando la vista recibe un mensaje que concierne al modelo o al controlador, lo deja registrado como el modelo con el cual se comunicara y apunta con la variable controller al controlador asignado, enviándole al mismo su identificación para que el controlador establezca en su variable view el identificador de la vista y así puedan operar conjuntamente. El responsable de deshacer estas conexiones, seguirá siendo la vista, quitándose a sí misma como dependiente del modelo y liberando al controlador.

**Interacción de los componentes**

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo de control que se sigue generalmente es el siguiente:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se reflejan los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, se podría utilizar el patrón Observador para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. Este uso del patrón Observador no es posible en las aplicaciones Web puesto que las clases de la vista están desconectadas del modelo y del controlador. En general el controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice. Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista. Por ejemplo en el MVC usado por Apple en su framework Cocoa. Suele citarse como Modelo-Interface-Control, una variación del MVC más puro
5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente....

**Uso en aplicaciones Web**

Aunque originalmente MVC fue desarrollado para aplicaciones de escritorio, ha sido ampliamente adaptado como arquitectura para diseñar e implementar aplicaciones web en los principales lenguajes de programación. Se han desarrollado multitud de frameworks, comerciales y no comerciales, que implementan este patrón (ver apartado siguiente "Frameworks MVC"); estos frameworks se diferencian básicamente en la interpretación de como las funciones MVC se dividen entre cliente y servidor.

Los primeros frameworks MVC para desarrollo web planteaban un enfoque de cliente ligero en el que casi todas las funciones, tanto de la vista, el modelo y el controlador recaían en el servidor. En este enfoque, el cliente manda la petición de cualquier hiperenlace o formulario al controlador y después recibe de la vista una página completa y actualizada (u otro documento); tanto el modelo como el controlador (y buena parte de la vista) están completamente alojados en el servidor. Como las tecnologías web han madurado, ahora existen frameworks como JavaScriptMVC, Backbone o jQuery que permiten que ciertos componentes MVC se ejecuten parcial o totalmente en el cliente.

**Conclusión**

Conforme se incrementan las necesidades de cualquier aplicación, la modificación al código existente se hace inminente, y si no existe una clara división de uso, el código no sólo se torna indescifrable, sino en ocasiones impredecible, debido a la mezcla de funcionalidades que pueden surgir.

La estructura MVC ("Model-View-Controller") es un paradigma utilizado en el desarrollo de diversossoftware, a través de este patrón se logra una división de las diferentes partes que conforman una aplicación, permitiendo la actualización y mantenimiento del software de una forma sencilla y en un reducido espacio de tiempo.

El uso de los frameworks basados en el patrón MVC permite tener una separación lógica y física de los componentes de la aplicación, permitiendo a su vez, una mayor especialización de los desarrolladores y diseñadores de la aplicación, además de contribuir a una elevada organización en el trabajo.