

**¨AÑO DE LA CONSOLIDACIÓN DEL MAR DE GRAU”**

**FACULTAD DE INGENIERIA**

* **INTEGRANTES:**

Herrera Fiestas Jocelyne Xiomara

Aguilar Ventura Kenneth Jesús

Ferrer Rubio Cristian Pedro

* **CICLO:**

III

* **PROFESOR:**

Coronel Castillo Eric Gustavo

* **TEMA**

Patrones MVC

2016

Lima-Perú

INDICE:

INTRODUCCION---------------------------------------------- PAG.3

RESUMEN------------------------------------------------------ PAG.4

HISTORIA----------------------------------------------------------------------- PAG.5

DESCRIPCION DEL PATRON------------------------------------------- PAG.6

INTERACCION DE LOS COMPONENTES

MVC Y BASE DE DATOS--------------------------------------- PAG.7

EJEMPLO--------------------------------------------------------- PAG.8

CONCLUSIONES------------------------------------------------ PAG.9

RECOMENDACIONES------------------------------------------------------- PAG.10

BIBLIOGRAFIA

ANEXOS-------------------------------------------------------------------------- PAG.11

INTRODUCCION

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.

Los patrones de diseño son soluciones al problema específico y común del diseño orientado a objetivos.

Uno de ellos es el patrón MVC, el cual separa el diseño de una solución en tres grandes componentes principales: Modelo, Vista, Controlador .estos manejan los datos de una aplicación, la interfaz de usuario y la lógica del control respectivamente.

En ingeniería los patrones de diseño son una herramienta fundamental que nos permite una solución general y reutilizable a problemas comunes en el diseño de software.

Los patrones de diseño son una plantilla o descripción que nos permite resolver un problema de muchas maneras diferentes. En un nivel superior existen los patrones de arquitectura que presentan un mayor alcance, ya que en estos describe un patrón general de todo un sistema.

RESUMEN

* El modelo-vista-controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de interfaz de usuario y el módulo encargado de gestionar eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura se software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.
* El modelo representa la información con la que trabaja la aplicación es decir, su lógica de negocio.
* La vista transforma el modelo en una página web que permita al usuario interactuar con ella.
* El controlador se encarga de procesar las interaccione del usuario y realiza los cambios.

Se basa en la separación de responsabilidades ya que cada una de las partes se encarga de un aspecto específico y no interfiere con el de las otras.

HISTORIA

El patrón MVC fue una de las primeras ideas en el campo de las interfaces graficas de usuario y uno de los primeros trabajos en describir e implementar aplicaciones software en términos de sus diferentes funciones.

MVC fue introducido por Trygve Reensauk en Smalltalk-76 durante su visita a Xerox Parc en los años 70 y, seguidamente, en los años 80, Jim Althoff y otros implementaron una versión de MVC para la biblioteca de clases de Smalltalk-80. Solo más tarde, en 1998, MVC se expresó como un concepto general en un artículo sobre Smalltalk-80.

En esta definición de MVC el controlador se definía como “el modulo que se ocupa de la entrada” (de forma similar a como la vista “se ocupa de la salida”). Esta definición no tiene cabida en las aplicaciones modernas en las que esta funcionalidad es asumida por una combinación de la ‘vista’ y algún framework moderno para desarrollo. El ‘controlador’, en las aplicaciones modernas de la década de 2000, es un módulo o una sección intermediaria de código, que hace de intermediario de la comunicación entre el ‘modelo’ y la ‘vista’, y unifica la validación.

Algunos aspectos del patrón MVC han evolucionado dando a lugar a ciertas variantes del concepto original ya que “las partes del MVC clásico realmente no tienen sentido para los clientes actuales.”

DESCRIPCION DEL PATRON

De manera genérica, los componentes de MVC se podrían definir como sigue:

* EL MODELO

Es la representación de la información con la cual el sistema opera, por lo tanto, gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de accesos que se hayan descrito en las especificaciones dela aplicación (lógica del negocio). Envía a la ‘vista’ aquella parte de la información que en cada momento se le solicita para que sea mostrada. Las peticiones de acceso o manipulación de información llegan al ‘modelo ‘a través del ‘controlador’.

* En otros puede ser un simple archivo, una hoja electrónica, entro otros.
* Además, representa la parte que implementa la “Lógica de negocio”.
* En términos de programación está dividida en  clases para hacer más fácil el programa.
* EL CONTROLADOR

Responde a eventos (usualmente acciones del usuario) e invoca peticiones al ‘modelo ‘cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comando a su ‘vista’ asociada si se solicita un cambio en la forma en que se presenta el ‘modelo’, por lo tanto se podría decir que el ‘controlador’ hace de intermediario entre la ‘vista’ y el ‘modelo’.

Es un intermediario entre las capas Modelo y Vista, el cual es responsable de actualizar el Modelo cuando el usuario manipula la Vista. También se puede entender que el Controlador lee o recolecta los datos desde la vista y se los envía al modelo. Por ejemplo, si el usuario hace clic en el botón del ratón o elige un elemento de menú, el controlador es responsable de determinar la forma en que la aplicación debe responder.

* LA VISTA

Presenta el ‘modelo’ (información y lógica del negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho ‘modelo’ la información que debe representar como salida. Es el responsable de mostrar toda o una porción de los datos de la aplicación, son los elementos que el usuario recibe y con los que puede interactuar.

INTERACCION DE LOS COMPONENTES

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo de control que 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 la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos.

3.- El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario.

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.

5.- La interfaz de usuario espera nuevas interacciones de usuario, comenzando el ciclo nuevamente.

MVC Y BASE DE DATOS

Muchos sistemas informativos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos que debe utilizar la aplicación; en líneas generales del MVC dicha gestión corresponde al modelo. La unión entre capa de presentación y capa de negocio conocido en el paradigma de la Programación por capas representaría la integración entre la ‘vista’ y su correspondiente ‘controlador’ de eventos y accesos de datos, MVC no pretende discriminar entre capa de negocio y capa de presentación pero si pretende separar la capa visual grafica de su correspondiente programación y acceso a datos, algo que mejora el desarrollo y mantenimiento de la ‘vista’ y el ‘controlador’ en paralelo, ya que ambos cumplen ciclos de vida muy distintos entre sí.

* EJEMPLO



Tal como se observa en esta imagen, el ‘Controlador’ (mesero) y el ‘Modelo’ (cocinero) esperan algún evento por parte de la ‘Vista’ (cliente). La misión en este caso, es dar un servicio de alimentación al cliente. Por supuesto el que inicia este ciclo es el cliente que solicita al mesero un plato y bebida específica. Luego el mesero al capturar la orden de comida del cliente, le solicita al cocinero prepararla. Tanto el cliente como el mesero esperan el resultado de esta orden.



El cocinero interpreta y prepara lo solicitado por parte del mesero. Pasando un tiempo, el cocinero devuelve al mesero el plato solicitado (datos). Y el mesero finalmente hace la entrega del plato al cliente.

CONCLUSIONES

* La estructura MVC es un paradigma utilizado en el desarrollo de diversos software, 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 patrón MVC es muy utilizado hoy en día, es un diseño que ofrece consistencia y baja complejidad en el desarrollo de software y podrán encontrar muchos framework que implementan esta solución.
* El patrón MVC en su implementación embebe diferentes patrones dependiendo de la naturaleza de la aplicación que se está diseñando.
* MVC es un patrón de diseño enfocado a separar las responsabilidades dentro de nuestra aplicación y es muy utilizado en la web por su enfoque y las ventajas que ofrece con respecto a algunas otras formas o patrones de desarrollo de aplicaciones web.
* El patrón modelo vista controlador (MVC) ha facilitado la creación de aplicaciones Web escalables y flexibles desde los orígenes de las mismas, sin embargo, la adaptación de este patrón depende de las tecnologías utilizadas.
* El patrón MVC se adapta muy fácilmente a la arquitectura Cliente-Servidor, sin importar la tecnología que se use en el desarrollo de las aplicaciones. Sin embargo, es necesario mantener una disciplina estricta que impida la propagación de incoherencias que destruyan el patrón y no se logren los cometidos del mismo.

RECOMENDACIONES

* Usa la mejor herramienta para tu trabajo:
* Java es apropiado para crear lógica de negocio.
* Los lenguajes de plantilla y etiquetado son más apropiados para crear LOYOUTS HTML.
* utilizar una capa de abstracción de base de datos para separar la lógica de negocio y la interacción con el modelo de datos.
* Separar el código, de tal manera que si necesitamos hacer un cambio en la base de datos, esto no afecte a la lógica del programa.
* Separar las funciones de la aplicación en modelos, vistas y controladores hace que la aplicación sea muy ligera.
* Con MVC la aplicación se puede desarrollar rápidamente, de forma modular y mantenible.

BIBLIOGRAFIA

* **González, Y. D., & Romero, Y. F. (2012). Patrón Modelo-Vista-Controlador.Revista Telem@ tica, 11(1), 47-57.**
* [**Applications Programming in Smalltalk-80(TM)**](http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html)
* **Bascón Pantoja, E. (2004). El patrón de diseño Modelo-Vista-Controlador (MVC) y su implementación en Java Swing. línea].[Recuperado el 15 de noviembre del 2013] de: http://www. ucbcba. edu. bo/Publicaciones/revistas/actanova/documentos/v2 n4/v2, (4).**
* **Anastasio Velásquez, Miguel M. *"Model View Controller (MVC)"***[**http://www.informatizate.net/articulos/model\_view\_controller\_mvc\_20040324.html**](http://www.informatizate.net/articulos/model_view_controller_mvc_20040324.html)**, 18 de febrero de 2006.  
    
  Leer más:**[**http://www.monografias.com/trabajos43/patron-modelo-vista/patron-modelo-vista2.shtml#ixzz4LsxzkJLE**](http://www.monografias.com/trabajos43/patron-modelo-vista/patron-modelo-vista2.shtml#ixzz4LsxzkJLE)

**ANEXOS:**

* [**https://www.google.com.pe/search?q=MVC&espv=2&biw=911&bih=420&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiOxOTY97rPAhVBmh4KHQqfDWYQ\_AUIBigB#tbm=isch&q=modelo+vista+controlador**](https://www.google.com.pe/search?q=MVC&espv=2&biw=911&bih=420&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiOxOTY97rPAhVBmh4KHQqfDWYQ_AUIBigB#tbm=isch&q=modelo+vista+controlador)**.**
* [**http://image.slidesharecdn.com/poo-introduccionalpatronmvc-reneemoralescalhua-130820142953-phpapp01/95/poo-introduccion-al-patron-mvc-renee-morales-calhua-13-638.jpg?cb=1377009040**](http://image.slidesharecdn.com/poo-introduccionalpatronmvc-reneemoralescalhua-130820142953-phpapp01/95/poo-introduccion-al-patron-mvc-renee-morales-calhua-13-638.jpg?cb=1377009040)**.**
* [**http://image.slidesharecdn.com/mvcii-160403165451/95/mvc-2-638.jpg?cb=1459702528**](http://image.slidesharecdn.com/mvcii-160403165451/95/mvc-2-638.jpg?cb=1459702528)**.**
* **http://eltamiz.com/elcedazo/wp-content/uploads/2010/06/3capas.gif.**