KnE Engineering

SIIPRIN Conference Proceedings
Simposio Iberoamericano en Programación Informática
(Ibero-American Symposium on Computer Programming)
SIIPRIN'2017

Knowledge E
Engaging minds

Conference Paper

Comparative Study of Performance and Productivity of MVC and MVVM design patterns

Gloria Arcos-Medina[1], Jorge Menéndez[2], and Javier Vallejo[3]

[1]Escuela Superior Politécnica de Chimborazo, Grupo de Investigación en Ingeniería de Software
[2]Escuela Superior Politécnica de Chimborazo, Grupo de Investigación Aplicaciones Informáticas para la Toma de Decisiones
[3]Escuela Superior Politécnica de Chimborazo, Facultad de Informática y Electrónica

Abstract

The research compares the patterns for user interfaces Model View Controller (MVC) and Model ViewView Model (MVVM) using.Net technologies for the system development for sales and purchasing management of a company. The inductive method is used to recognize, obtain and measure the results of performance and productivity variables based on the tests of two prototypes created for the MVC and MVVM pattern. For the performance tests we used: performance measurement tools Visual Studio 2012 Ultimate. The obtained results show that MVVM pattern presents better conditions of performance and productivity meeting the 95.1% of its performance levels with respect to the MVC pattern that fulfills the 41.6% of its performance levels in relation to the performance and productivity with a level of significance of 95% and a margin of error of 5%.

Keywords: Performance, Productivity, Model View Controller, Model View View Model

Corresponding Author:
Gloria Arcos-Medina
garcos@espoch.edu.ec

Received: 28 July 2017
Accepted: 5 September 2017
Published: 30 January 2018

Publishing services provided by Knowledge E

Selection and Peer-review under the responsibility of the SIIPRIN Conference Committee.

1. Introducción

En la actualidad existes varios patrones para el diseño de interfaces de usuario, cada uno con sus facilidades y dificultades. Para lograr identificar el patrón que nos presente mejores ventajas en el desarrollo de aplicaciones web profesionales se deberá realizar un estudio comparativo entre los patrones existentes, en el presente caso serán los patrones Modelo Vista Controlador (MVC) y Modelo Vista Vista Modelo (MVVM).

La complejidad de un sistema y los problemas recurrentes que contiene, require que los desarrolladores organicen su código de tal manera que sea fácil de comprender y solucionar problemas. Se disminuye el caos de un sistema complejo mediante la aplicación de patrones para el diseño de interfaces de usuarios, el uso de estos patrones

para la interfaz de usuario nos brindará una aplicación más estructurada, orgni-zada y escalable, es inevitable realizar un estudio de los patrones para interfaces de usuario y tomar la decisión más acertada sobre qué patrón es el más adecuado para el desarrollo de la aplicación.

En el artículo "Understanding the basics of MVVM design pattern", Shukkur, Shamlia afirma que el enfoque tradicional de los desarrolladores de crear la interfaz de usuario y agregarle toda la lógica de programación, "genera una fuerte dependencia entre la interfaz de usuario, el acceso a los datos y la lógica del negocio", lo que provoca que "no hay dos desarrolladores que pueden trabajar simultáneamente en la misma vista y los cambios de un desarrollador pueden dañar el código del otro. Así que, el que todo esté en un lugar, es una mala idea para la sostenibilidad, extensibilidad y posibilidad de prueba" [1], por lo que propone la utilización del patron de diseño MVVM.

Por otra parte, Md. Khaliluzzaman en su artículo "Pre and Post Controller based MVC Architecture for Web Application" afirma que el incrustar toda la lógica empresarial en las páginas HTML "es bueno para pequeñas empresas. Sin embargo, no es una buena solución para la organización empresarial de gran tamaño" por lo que "la arquitectura MVC (Model / View / Controller) se introduce debido a que la empresa necesita la nueva arquitectura para separar la lógica de negocio de la capa de presentación" la misma que "logra más popularidad debido a que aísla la lógica de negocio de la capa de presentación" [2].

Varias aplicaciones han sido desarrolladas utilizando patrones MVC y MVVM, por ejemplo [3], [4], [5], en los que se indica que estos patrones son útiles para el diseño de la arquitectura de software interactivo, permiten crear software de una manera mas organizada, entre otras ventajas.

El objetivo de este estudio es comparar el rendimiento, considerando los parámetros: indicador clave, tiempo de respuesta de la página, y controladores y agentes para decidir que parámetro presenta mejores condiciones de rendimiento. Así también se determinó la productividad, la misma que fue medida mediante los indicadores de: línea de código, complejidad y disponibilidad de información. Las pruebas fueron realizadas en dos prototipos desarrollados con los patrones MVC y MVVC, que son objeto de este estudio.

El resto del artículo está organizado en cuatro secciones. La Sección 2 describe los conceptos y características de: patrones de diseño, Modelo Vista Controlador y Modelo Vista Vista Modelo. La Sección 3 presenta la metodología utilizada para la realización del estudio comparativo, de describen los parámetros, indicadores y formas

de medición correspondientes a las variables rendimiento y productividad. La Sección 4 presenta los resultados del trabajo y la Sección 5 sus conclusiones.

2. Revisión de Literatura

2.1. Patrones de Diseño

Uno de los conceptos que describen de manera más efectiva qué es un patrón de diseño es el propuesto por Gamma et al. en su libro Design Patterns: "Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interfaces de usuario" [6]. Un patrón de diseño proporciona una solución a un problema que se presenta en la capa de presentación, específicamente en referentes a las interfaces de usuario.

Los patrones de diseño tuvieron relevancia en el desarrollo de software a partir de los primeros años de la década de 1990, cuando se publicó el libro "Design Patterns" escrito por el grupo "Gang of Four (GoF)" Banda de los cua-tro, compuesto por Erich Gamma, Richard Helm, Ralph Johnson y John Vlisides, en el que se recogían 23 patrones de diseño comunes [7].

Los patrones de diseño tienen como objetivo principales: a) Proporcionar catálogos de componentes reutilizables para el diseño de sistemas de software. b) Estandarizar una terminología común entre diseñadores. d) Estandarizar la manera en que se realiza el diseño. e) Facilitar el aprendizaje de futuros diseñadores integrando un conocimiento ya existente.

Para representar un patrón se utilizan plantillas, con el fin de representarlos de manera uniforme, de tal forma que se constituyan en un medio de comunicación efectivo y uniforme entre diseñadores. En general, la estructura general de un patrón tiene: nombre, tipo, objetivo, motivación, aplicabilidad, estructura, participantes, colaboraciones, consecuencias, implementación, código, usos conocidos y patrones relacionados.

La clasificación utilizada en este trabajo es la presentada por GoF en su libro "Design Patterns"[6], donde determina que los patrones se clasifican en: *Patrones Creacionales***:** abstraen el proceso de creación de instancias de objeto, ayudan hacer un sistema independiente de cómo se crean, se componen y se representan sus objetos; en esta categoría tenemos: Abstract Factory, Factory Method, Prototype, Singleton. *Patrones Estructurale,* relacionados con la composición de clases u objetos, se ocupan de cómo las clases y objetos son utilizados para componer estructuras de mayor tamaño; dentro de estos patrones tenemos: Adapter, Bridge, Composite, Decorator, Facade, Flyweight,

Proxy, Módulo. *Patrones de Comportamiento*, caracterizan el modo en que las clases y objetos interactúan y se reparten la responsabilidad, entre éstos se tienen: Chain of Responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template Method, Visitor.

2.2. Modelo Vista Controlador

El Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de presentación de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Con este fin, MVC propone la construcción de tres componentes 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 [8]. Este patrón de diseño se basa en las ideas de reutilización de código y la separación de conceptos, para facilitar el desarrollo de aplicaciones y su posterior mantenimiento [9].

El *controlador*, en las aplicaciones actuals, es un módulo o una sección intermedia de código, que hace de intermediario de la comunicación entre el 'modelo' y la 'vista', y unifica la validación (utilizando llamadas directas o el "observer" para desacoplar el 'modelo' de la 'vista' en el 'modelo' activo) [10].

El *modelo* es el conjunto de clases encargado de representar la información con que trabaja el usuario. Estas clases pueden ser clases de dominio, DTO (Data Transfer Object) o ViewModels (Entendiendo como ViewModel una clase que almacena los datos que representa una determinada vista, no un ViewModel [11].

Las *vistas* son las delegadas de representar gráficamente el modelo y de brindar las operaciones de los controladores para que el usuario pueda interactuar con el sistema diseñado. En el caso de una aplicación web la vista es una página HTML con contenido dinámico sobre la cual se puede realizar operaciones.

El *controlador* organiza la interacción entre las vistas y el modelo. Recibe las peticiones del usuario, interactúa con el modelo realizando consultas y modificaciones a este, decide qué vista se muestra como respuesta y le proporciona los datos requeridos para su renderizado, o delega la respuesta a otra acción de otro controlador.

2.3. Modelo Vista Modelo de Vista

El patron Modelo Vista Vista Modelo o en inglés Model View ViewModel, es una variación del MVC que está diseñado para plataformas de desarrollo de interfaz de

usuario modernas donde la vista es responsabilidad de un diseñador en lugar de un desarrollador. MVVM se basa en un mecanismo general de enlace de datos que facilita el desarrollo de la capa de separación de vista desde el resto del patrón mediante la eliminación de todo el código subyacente de la capa de la vista.

El *modelo*, encapsula la lógica de negocio y datos. La lógica empresarial se define como cualquier lógica de aplicación que se ocupa de la recuperación y gestión de datos de la aplicación y de asegurar que las reglas del negocio que garanticen la coherencia de los datos y la validez se cumplan. Para maximizar las oportunidades de reutilización, los modelos no deben contener ningún caso de uso o conducta específica del usuario o lógica de la aplicación.

La *vista*, encapsula la interfaz de usuario y la lógica de la interfaz de usuario. Las vistas definen la interfaz de usuario específica para una parte de la aplicación, son normalmente los controles que se han diseñado para funcionar bien cuando se une a un ViewModel.

El *viewmodel*, encapsula la lógica de presentación y estado. Lógica de presentación se define como la lógica de la aplicación que se ocupa de casos de uso de la aplicación (o casos de usuario, tareas de usuario, flujo de trabajo, etc.) y define el comportamiento lógico y la estructura de la aplicación. Para maximizar las oportunidades de reutilización, el ViewModel no debe tener ninguna referencia a las clases específicas de interfaz de usuario, elementos, controles o comportamiento.

3. Metodología

Considerando que el propósito del presente estudio es determinar cuál es el patrón de interfaces de usuario más adecuado con respecto a productividad y rendimiento, se elaboran dos prototipos de aplicaciones web, las que usarán el patrón Model View Controler (MVC) y Model View ViewModel (MVVM).

Prototipo 1: implementado con el patrón MVC con la tecnología ASP.net MVC 4.

Prototipo 2: implementado con el patrón MVVM con la tecnología Light Switch y Silverlight.

Para los dos prototipos se implementaron los siguientes módulos: módulo de autenticación. módulo de gestión de empleados, módulo de gestión de clientes, módulo de gestión de proveedores, módulo de gestión de productos, módulo de gestión de compras, módulo de gestión de ventas.

3.1. Variables de estudio

El estudio comparativo de los patrones MVC y MVVM se realizó en función de dos variables: rendimiento y productividad. Uno de los puntos más importantes en el desarrollo web es el rendimiento de la página web, una página web veloz y optimizada mejora considerablemente la experiencia web del usuario. La productividad otra característica importante para poder seleccionar el mejor patrón para el desarrollo de interfaces de usuario.

Cada una de las variables de estudio tuvo un peso establedido a través de sus respectivos indicadores; el rendimiento tuvo un peso del 70% y la productividad del 30%.

3.1.1. Variable Rendimiento

Para realizar correctamente las pruebas de rendimiento se estudiaron los paráme-tros especificados en la Tabla 1 con sus respectivos indicadores y pesos, los mismos que se presentan en la Tabla 2.

Tabla 1: Parámetros de Rendimiento.

Parámetro	Descripción
Indicadores claves	Contadores que describen los aspectos básicos de la realización de la prueba.
Tiempo de Respuesta de la Página	El tiempo de medio de la respuesta de las páginas web que se accede durante las pruebas de carga.
Controladores y Agentes	Información sobre los equipos en los que se ejecutan las pruebas de carga. Esto incluye datos sobre el uso de la memoria, el procesador, el disco físico, los procesos.

Para la determinación del rendimiento se aplicó la guía de Microsoft "Performance Testing Guidance for Web Applications" [12], que consiste de las siguientes actividades: 1) Identificar el entorno de las pruebas 2) Identificar los criterios de aceptación del rendimiento 3) Planificar y diseñar las pruebas 4) Configurar el entorno de prueba 5) Ejecutar la prueba 6) Analizar los resultados, realizar un informe y repetirlo.

Para la prueba de rendimiento se ejecutaron las siguientes acciones en cada uno de los prototipos diseñados: a) seleccionar cliente, b) seleccionar empleado, c) seleccionar fecha, d) añadir número de factura, e) seleccionar el estado de la factura, d) seleccionar 5 productos, e) seleccionar los productos con las diferentes unidades de venta que existen como son: unidad, decena, ciento y millar; y f) guardar la factura.

Tabla 2: Indicadores de los parámetros de rendimiento.

Parámetro	Indicador	Descripción	Peso
Indicadores claves	Promedio de tiempo de descarga de la página	Tiempo medio para descargar las páginas y todas sus solicitudes.	15
	Páginas por segundo	Número de veces que se está accediendo a la memoria virtual.	10
Tiempo de respuesta de la página	Solicitudes/segundo	El número de solicitudes ejecutado por segundo. Esto representa el rendimiento actual de la aplicación. Bajo carga constante, este número debe permanecer dentro de un cierto rango.	10
	Promedio tiempo de respuesta	Muestran una relación entre el tiempo total transcurrido del intervalo de muestra con el número de procesos u operaciones realizadas durante ese tiempo.	15
Controladores y agentes	Porcentaje del tiempo del procesador	Porcentaje del CPU utilizado.	10
	Mbyte disponibles	Memoria disponible para nuevos procesos	10
Total			70

El análisis de resultados se realizó en base a la estadísticica descriptive de los datos obtenidos, con el objetivo de describir su comportamiento Además se aplicó la prueba U de Mann-Whitney, prueba no paramétrica para muestras independientes, que permitió determinar la existencia de diferencias significativas entre los indicadores de rendimiento motivo de estudio.

3.1.2. Variable Productividad

La productividad se determinó a través del nivel de sencillez y aprendizaje que poseen los patrones MVC y MVVM a la hora de instruirse e implementarlos en el desarrollo de aplicaciones web. Para realizar una correcta medición de la productividad de las aplicaciones web se estudiaron los indicadores que se presentan en la Tabla 3 con sus respectivos pesos.

3.2. Herramientas para pruebas de rendimiento y productividad

Las herramientas que se utilizaron para las medir los indicadores de las pruebas de rendimiento fueron: *Herramientas de Análisis de Código de Visual Studio 2012*, que proporcionan un conjunto de medidas de software que proporcionan a los programadores una mejor visión del código que están desarrollando y *Herramientas de Prueba*

Tabla 3: Indicadores de los parámetros de productividad.

Parámetro	Indicadores	Descripción	Peso
Productividad	Líneas de Código	Permite medir la cantidad de líneas de código para la realización de una consulta en específico.	10
	Complejidad	Mide la facilidad o complejidad de entendimiento del código.	15
	Disponibilidad de información	linformación disponible de cada una de las herramientas es importante para aquellas personas que necesitan realizar diseños profesionales o tener conocimiento más avanzado. Medido en número de fuentes	5
Total			30

y Rendimiento Web de Visual Studio 2012, para determinar lo bien que res-ponde su programa a diferentes niveles de uso por medio de pruebas de carga. Una prueba de carga modela el uso previsto de un programa de software simulando varios usuarios que obtienen acceso al programa al mismo tiempo.

4. Resultados

4.1. Variable Rendimiento

Los resultados obtenidos de la medición de los indicadores de la variable rendimiento se presentan en las Tablas 4 y 5.

El P-valor (significancia asintótica) para la prueba U de Mann-Whitney como se observa en la Tabla 5, para cada uno de los indicadores es menor al valor alfa que es del 5% (<0,05), es decir que: "Existe una diferencia significativa entre la media del promedio calculado en cada indicador para el patrón MVC con respecto al patrón MVVM". Los resultados evidencian que para todos los indicadores de rendimiento, excepto *Mbyte disponibles*, el patrón MVVM presenta major comportamiento que el patrón MVC.

Para la calificación de los indicadores de la variable rendimiento se utilizaron los pesos establecidos en la Tabla 2. donde el valor obtenido representa el 70% de la calificación total. La calificación de cada indicador se presenta en la Tabla 5. donde se observa que los resultados obtenidos para el parámetro MVVM es de 69.7%, mientras que para el patrón MVC el resultado es de 27.94%.

Tabla 4: Estadística descriptiva promedio de tiempo de descarga de la página.

	Frecuencia	Promedio	Error Relativo	Desviación estándar
Promedio de tiempo de descarga de la página				
MVVM	241	0.07052574	0.020537639	0.318829891
MVC	241	1.433100848	0.46068736	7.151791065
Páginas por Segundo				
MVVM	241	314.1613732	59.39171856	0,04606
MVC	241	394.6475189	58.84198053	0,06815
Solicitudes por Segundo				
MVVM	241	11.70788382	0.326928659	5.075297619
MVC	241	678.7236515	13.87582041	215.4106601
Promedio de tiempo de respuesta				
MVVM	241	0.070525753	0.020537639	0.31882989
MVC	241	0.289595568	0.114490168	1.777365376
Porcentaje de tiempo de procesador				
MVVM	241	33.70982817	0.488277434	7.564361484
MVC	241	62.1948585	0.657010877	10.17836873
Mbyte disponible				
MVVM	241	2136.394191	15.53211857	241.1233221
MVC	241	2202.556017	13.57566871	210.7510527

Tabla 5: Estadísticos de contraste.

U de Mann-Whitney	W de Wilcoxon	Z	Sig. asintót. (bilateral)
Promedio de tiempo de descarga de la página			
13157.500	42318.500	-10.388	2.803E-25
Páginas por Segundo			
15327.000	44488.000	-8.969	2.989E-19
Solicitudes por Segundo			
2322.500	31483.500	-17.475	2.212E-68
Promedio de tiempo de respuesta			
8931.000	38092.000	-13.152	1.647E-39
Porcentaje de tiempo de procesador			
1494.000	30414.000	-17.971	3.305E-72
Mbyte disponible			
1494.000	30414.000	-17.971	2.685E-16

4.2. Variable Productividad

Los indicadores de la variable productividad: líneas de código y complejidad se medieron en base al módulo de gestión de compras. Los resultados se muestran en la Tabla 6.

Tabla 6: Calificación de indicadores.

Parámetro	Indicador	MVVM	MVC
Indicadores claves	Promedio de tiempo de página	15	0.73818
	Páginas por segundo	10	7.96056
	Solicitudes/segundo	10	0.172498539
Tiempo de respuesta de la pagina	Promedio de tiempo de respuesta	15	3.65298
Controladores y agentes	Porcentaje del tiempo del procesador	10	5.42003
	Mbyte disponible	9.69961	10
Total		69.69961	27.5878685

Tabla 7: Valores calculados de los indicadores de la variable Productividad.

Indicador	MVVM	MVC
Líneas de código	68	223
Complejidad	3	8
Disponibilidad de información	680600	8869000

Para la calificación de los indicadores de la variable productividad, se utilizaron los pesos establecidos en la Tabla 3. El total de la calificación obtenida representa el 30% de la calificación total. La calificación de cada indicador se presenta en la Tabla 7. Se puede oservar que los resultados obtenidos para el parámetro MVVM es de 25.38%, mientras que para el patrón MVC el resultado es de 13.67%.

Tabla 8: Calificación de indicadores de la variable productividad.

Indicador	MVVM	MVC
Líneas de código	10	3.049327354
Complejidad	15	5.625
Disponibilidad de información	0.38369602	5
Total	25.38369602	13.674327354

En la Tabla 8 se muestran, en resumen, las calificaciones de cada variable de comparación, la misma que incluye el porcentaje de evaluación obtenido, y el peso asignado a cada variable. Aquí se evidencia que el patron MVVM es superior en rendimiento y productividad al patron MVC.

Tabla 9: Resumen de resultados.

Variable	MVC	MVVM	Peso
Rendimiento	27.94%	69.7%	70%
Productividad	13.67%.	25.38%	30%
Total	41.61%	95.08%	100%

5. Conclusiones

El estudio comparativo entre los patrones MVVM y MVC ha permitido determinar que el patrón MVVM presenta mejores condiciones de rendimiento y productividad con respecto al patrón MVC ambos patrones desarrollados con tecnología.Net.

El modelo y la vista cumplen las mismas funciones tanto para el patrón MVVM como para el patrón MVC; la diferencia más distintiva del patrón MVC la acción del usuario afecta directamente al controlador mientras que en el patrón MVVM la acción afecta directamente a la vista.

El patrón MVVM presenta un cumplimiento de las variables analizadas del 95.1% mientras que el patrón MVC presenta un cumplimiento del 41,6%. El patrón MVVM presenta un 53.5% en superioridad de niveles de rendimiento y productividad con respecto al patrón MVC.

Referencias

[1] Microsoft, S.M.: Understanding the basics of MVVM design pattern, https://blogs .msdn.microsoft.com/msgulfcommunity/2013/03/13/understanding-the-basics-of-mvvm-design-pattern/.

[2] Khaliluzzaman, M., Chowdhury, I.I.: Pre and post controller based MVC architecture for web application. In: 2016 5th International Conference on Informatics, Electronics and Vision (ICIEV). pp. 552–557 (2016).

[3] Leff, A., Rayfield, J.T.: Web-application development using the Model/View/Controller design pattern. In: Proceedings Fifth IEEE International Enterprise Distributed Object Computing Conference. pp. 118–127 (2001).

[4] Sorensen, E., Mikailesc, M.: Model-View-ViewModel (MVVM) Design Pattern using Windows Presentation Foundation (WPF) Technology. MegaByte J. 9, 1–19 (2010).

[5] Cortez, R., Vazhenin, A.: Developing re-usable components based on the virtual-mvc design pattern. In: International Workshop on Databases in Networked Information Systems. pp. 132–149. Springer (2013).

[6] Gamma, E., Johnson, R., Vlissides, J., Helm, R.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Proffesional Computing Series, USA (2005).

[7] Sensagent: Patrón de diseño?: definición de Patrón de diseño y sinónimos de Patrón de diseño (español),

http://diccionario.sensagent.com/Patr%C3%B3n%20de%20dise%C3%B10/es-es/.

[8] Reenskaug, T., Coplien, J.O.: The DCI Architecture: A New Vision of Object-Oriented Programming, http://www.artima.com/articles/dci_vision.html.

[9] Yiiframework: Fundamentals: Model-View-Controller (MVC) | The Definitive Guide to Yii | Yii PHP Framework, http://www.yiiframework.com/doc/guide/1.1/en/basics.mvc.

[10] Microsoft: Model-View-Controller, https://msdn.microsoft.com/en-us/library/ff649643.aspx.

[11] Estruch: MVC – De todo un poco..., https://jjestruch.wordpress.com/category/asp-net/mvc/, (2012).

[12] Meier, J.D., Farre, C., Bansode, P., Barber, S., Rea, D.: Performance Testing Guidance for Web Applications, https://msdn.microsoft.com/en-us/library/bb924375.aspx.