

# Marco de Trabajo para Seleccionar un Patrón Arquitectónico en el Desarrollo de Software

Juan Camilo Giraldo Mejía<sup>1</sup>, Fabio Alberto Vargas Agudelo<sup>2</sup>, Kelly Garzón Gil<sup>3</sup>

[jgiraldo1@tdea.edu.co](mailto:jgiraldo1@tdea.edu.co); [fvargas@tdea.edu.co](mailto:fvargas@tdea.edu.co); [kellyjoha-8@hotmail.com](mailto:kellyjoha-8@hotmail.com)

<sup>1</sup> Tecnológico de Antioquia, Robledo Medellín, 0500, Antioquia, Colombia.

<sup>2</sup> Tecnológico de Antioquia, Robledo Medellín, 0500, Antioquia, Colombia.

<sup>3</sup> Tecnológico de Antioquia, Robledo Medellín, 0500, Antioquia, Colombia.

**Pages: 568-581**

**Resumen:** Existen una serie de problemas de diseño de software que se reflejan en el desacoplamiento, en aspectos como la lógica empresarial, la interfaz de usuario, la navegación, y la arquitectura de la información; afectando principalmente la calidad del producto. Para resolver los problemas descritos, se propone un framework para seleccionar el patrón de arquitectura más adecuado según el contexto de la aplicación. Su diseño implicó la caracterización de patrones de software que relacionan soluciones probadas, desde arquitecturas de información y patrones de interoperabilidad, hasta patrones de navegación, interacción y visualización, todo soportado en tipos de desarrollo y variables identificadas con revisión de literatura, y consulta a expertos de algunas empresas. Se analizaron los patrones de mayor impacto en el sector productivo, y se identificó la taxonomía de proyectos de desarrollo según el contexto de aplicación, lo que permitió mejor selección del patrón de arquitectura durante el desarrollo de software.

**Palabras-clave:** Arquitectura de Software, Patrones de Arquitectura, Calidad, Framework en Arquitectura de Software.

## *Framework To Select An Architectural Pattern In Software Development*

**Abstract:** There are a number of software design problems that are reflected in decoupling, in aspects such as business logic, user interface, navigation, and information architecture; mainly affecting the quality of the product. To solve the problems described, a framework is proposed to select the most appropriate architecture pattern according to the context of the application. Its design involved the characterization of software patterns that relate proven solutions, from information architectures and interoperability patterns, to navigation, interaction and visualization patterns, all supported by types of development and variables identified with literature review, and expert consultation. of some companies.

The patterns with the greatest impact in the productive sector were analyzed, and the taxonomy of development projects was identified according to the application context, which allowed better selection of the architecture pattern during software development.

**Keywords:** Software Architecture, Architecture Patterns, Quality, Framework in Software Architecture.

## 1. Introducción

La llegada de nuevos tipos de aplicaciones que van desde servicios de presentación en pequeña escala, hasta servicios que soportan aplicaciones críticas a gran escala motiva la necesidad de tener conocimiento sobre arquitectura de software en un proyecto de desarrollo, donde no sólo se esperan entregas oportunas sino también calidad del producto resultante, el cual se ve afectado por el desconocimiento de patrones que son base para la estructuración de un proyecto de TI (Harper & Zheng, 2015). Sin embargo, la aplicación de la arquitectura de software depende de una gran cantidad de factores que generan impacto sobre el producto, reflejado en el rendimiento, mantenimiento y éxito general del entregable final (Harper & Dagnino, 2014).

En la actualidad la Arquitectura de Software facilita a un arquitecto tener a disposición una amplia gama de patrones, que superan problemas de adaptabilidad de requerimientos, rendimiento, modularidad y acoplamiento de los componentes, brindando calidad en menor tiempo (Ari & Mamatnazarova, 2014). Debido a la facilidad que otorgan dichos patrones, se presenta un marco de trabajo para aliviar problemas que afectan directamente la calidad, y proporcionar una guía que ayude al desarrollador a fortalecer conocimiento sobre arquitectura de Software.

El proporcionar un marco permite reducir tiempos, costos, mejoras en la calidad del producto de Software, y escalabilidad al tener una estructura definida permitiendo que cada capa este desacoplada aumentando la flexibilidad y mantenibilidad del producto.

El artículo se estructura de la siguiente manera: en la sección II se presenta el estado de la cuestión alrededor de la temática planteada; en la sección III se establecen los materiales y métodos utilizados, así mismo, cada una de las fases que se llevaron a cabo para encontrar la solución al problema; en la sección IV se presentan los resultados obtenidos, y finalmente en la sección V las conclusiones y referencias obtenidas dentro de la investigación.

## 2. Estado de la Cuestión

Para el diseño, especificación y definición del estado de la cuestión se planteó inicialmente la identificación de la realidad y las consecuencias del problema con el fin de poder ahondar sobre las publicaciones realizadas por otros autores que pretendan solventar un problema similar, véase la figura 1.

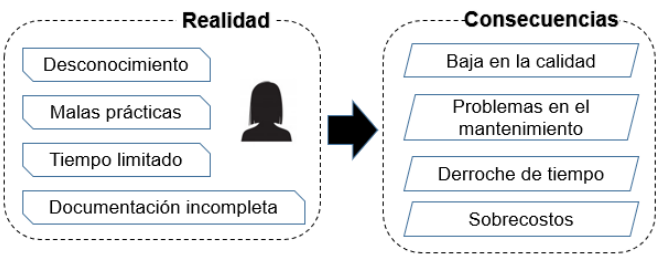


Figura 1 – El problema y como afecta la calidad  
Fuente (Elaboración propia)

Se hallaron algunos aportes de autores acerca de la arquitectura mediante marcos de trabajo que permitieran ayudar sobre el tema. En el trabajo de (Upadhyay,2016) se propone un marco para la toma de decisiones considerando factores de gestión y organización con parámetros de diseño que afectan la arquitectura de software, permitiendo evaluar y seleccionar software de calidad a través de estrategias propuestas en su investigación. (Guerrero & Londoño 2014) presentan un comparativo que busca identificar los mejores marcos de trabajo para utilizar el paradigma orientado a objetos estableciendo criterios y métricas para facilitar el comparativo y poder evidenciar ventajas en cada uno de los marcos de trabajo desarrollados en diferentes lenguajes de programación. En la propuesta de (Sabry,2015) se evalúan las mejoras tácticas de

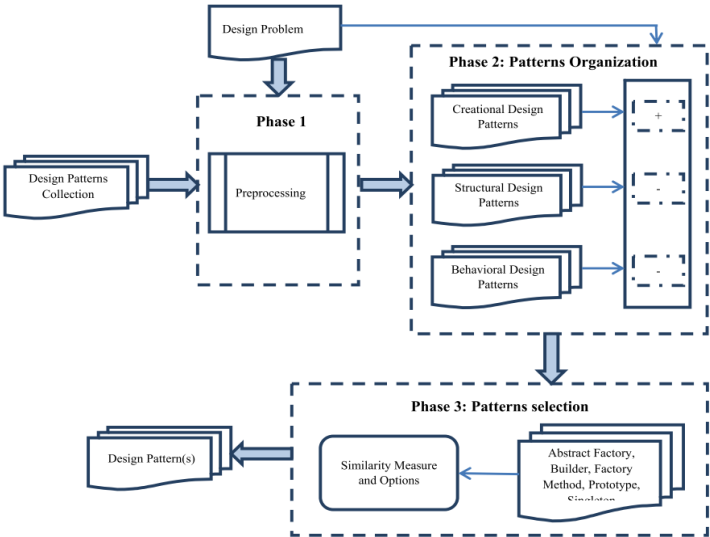


Figura 2 – Representación del framework propuesto Fuente (Hussain et al.,2019)

implementación de atributos de calidad que afectan la arquitectura y propiedades de las aplicaciones, reuniendo información aportante para la toma de decisiones. (Hussain et al., 2019) proponen un marco que utiliza un enfoque de categorización de texto para lograr tres objetivos, organizar los patrones en un número apropiado de clases de patrones, seleccionar el diseño apropiado de patrón para un problema de diseño y sugerir y seleccionar los patrones de diseño apropiados para un problema de diseño dado. Véase la figura 2.

En el trabajo de (Upadhyay, 2016) se describe un marco para la toma de decisiones en la arquitectura de software que tiene en cuenta factores de gestión, organizacionales, objetivos de desarrollo y parámetros de diseño que afectan la arquitectura de software. Lo anterior integrándolo con TOPSIS (Technique for Orden Preference by Similarity to Ideal Solution) para evaluar y seleccionar la calidad en la arquitectura de software.

### **3. Materiales y Métodos**

Se estableció un conjunto de fases o etapas para el desarrollo del marco de trabajo propuesto, las cuales son descritas en esta sección. Todas las fases están orientados a estructurar el marco de trabajo, para permitir un tratamiento único que indique el patrón más adecuado, a partir del juicio de expertos, que contribuirá a seleccionar una correcta arquitectura para un proyecto de Software antes de iniciar las actividades de codificación, evitando reprocesos y poca mantenibilidad del producto durante o al final del ciclo de Desarrollo.

#### **Fase 1- Población - expertos**

Se enfoca principalmente en arquitectos y desarrolladores de software, teniendo en cuenta que algunas compañías del sector productivo no cuentan con una figura de arquitecto, pasando a ser responsabilidad del desarrollador para cumplir con esta fase del ciclo de vida del desarrollo de software.

#### **Fase 2- Recolección**

Se realiza la decantación de la información en fuentes bibliográficas y en el universo de información referente al centro de esta investigación. Además se diseñan encuestas que permiten a través de una herramienta en línea conocer la opinión del grupo de expertos acerca de su experiencia con el desarrollo de software en el sector.

#### **Fase 3- Recuento**

La información recolectada a través de las diferentes fuentes bibliográficas y expertos, se procesa, y se obtienen resultados que sirven de insumo para el diseño del marco de trabajo, y la construcción del prototipo. El recuento y análisis de la información se realizó de acuerdo con los puntajes obtenidos con las encuestas y las valoraciones entregadas por los diferentes expertos consultados.

#### Fase 4- Presentación

Se define un esquema para representar el marco de trabajo, y será la ruta para la elaboración de un prototipo que pruebe la funcionalidad del marco.

#### Fase 5- Síntesis y análisis.

Construido el prototipo se analiza su funcionalidad, a partir del marco de trabajo propuesto y su comparativa con otras metodologías revisadas, y la validación de un experto que en la fase inicial fue encuestado. Los métodos empleados y la recolección de la información parten de las siguientes actividades:

### 3.1. Caracterización de los patrones de arquitectura

Se caracterizó un conjunto de patrones de arquitectura de software a través de revisión de literatura, y el contexto empresarial. Se clasificaron 8 patrones, entre ellos MVC, que es una arquitectura de tres capas. La primera está relacionada con la lógica de entrada del usuario, la segunda capa está relacionada con la lógica de negocios y la tercera capa se usa para implementar la lógica de la interfaz de usuario, que proporciona un acoplamiento entre estas tres capas (Abdul & Ibtisam, 2018); MVP por su parte, es un patrón de diseño reconocido para los desarrolladores de Android, el cual permite desacoplar la lógica de negocios (Modelo) de la lógica de vista mediante la introducción de un intermediario llamado Presentador (Li et al., 2015). MVVM se basa en el hecho de la Vista y el estado de la Vista en los enfoques anteriores (MVC / MVP), donde todavía están interconectados con el Modelo en un grado en el que las pruebas individuales son difíciles de lograr. Este enlace interfiere con el principio general de programación modular (Yang et al., 2018); Los Microservicios a su vez son un enfoque para desarrollar una aplicación de software como una serie de pequeños servicios, cada uno ejecutándose de forma autónoma y comunicándose entre sí, por ejemplo, a través de peticiones HTTP a sus API. Con respecto a manejar una arquitectura en la nube, éste, incorpora una tecnología de virtualización que permite que la infraestructura física sea alquilada. Además, cambia la ecuación proporcionando herramientas de plataforma específicas para acelerar el desarrollo (Lou, 2016) como arquitectura por capas y SOA; Cada uno de los patrones mencionados presentan diferentes variables que aportan una estructura definida y desacoplan los diferentes componentes de cada proyecto de software según sus necesidades.

### 3.2. Clasificación de las características de cada patrón

La figura 3 expone como algunos autores clasifican las características más sobresalientes de cada patrón.

### 3.3. Aplicación de las encuestas a público experto

Se construyó una encuesta dirigida a un grupo de expertos que permitió seleccionar los 4 patrones de más impacto. Los resultados se presentan en la figura 4.

Autor	Patrón			
	MVP	MVC	Microservicios	Arquitectura en la nube
[12]	MOD	MEM		
	REN			
	TES			
[13]		MAN		
		REN		
[19]		VE		
[14][15]	MAN			
	MODU			
	TES			
	FLE			
[16][17][18][19]			REN	
			MAN	
			SEG	
			FLE	
[20]			No TES	
[16]				SEG
				FLE

MEM	Memoria
MOD	Modificabilidad
REN	Rendimiento
TEST	Testeabilidad
MAN	Mantenibilidad
VE	Velocidad
MODU	Modularidad
FLE	Flexibilidad
SEG	Seguridad

Figura 3 – Características por cada patrón  
Fuente (Elaboración propia)

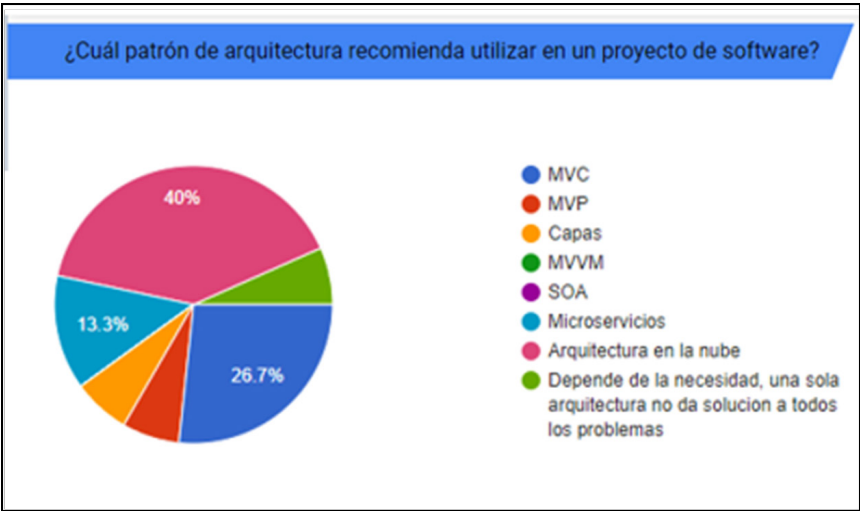


Figura 4 – Resultados encuesta a grupo de expertos.  
Fuente (Elaboración propia)

Se clasificaron 4 patrones para proyectos de software, MVC, microservicios, arquitectura en la nube y MVP. En la figura 5 se observan los patrones seleccionados y su relación con las variables de acuerdo con el análisis realizado y validado.

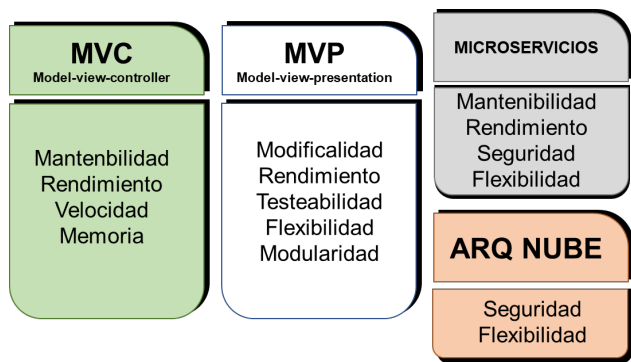


Figura 5 – Patrones clasificados y relación con variables según literatura y expertos  
Fuente (Elaboración propia)

3.4.Caracterización de los tipos de desarrollo

Identificación de los tipos de desarrollos existentes. Los tipos de desarrollo más utilizados en la actualidad se dividen en aplicaciones móviles. Estos son programas diseñados para ser ejecutados en celulares, tablets y otros dispositivos móviles, y su desarrollo es similar al desarrollo web (Richards, 2016); las aplicaciones web son una aplicación informática accesible desde cualquier navegador, ya sea a través de internet o a través de una red local y se pueden clasificar en webforms, aplicaciones Silverlight u otra aplicación web desarrollada en los diferentes lenguajes tales como JavaScript (Camargo, 2016); Por su parte las aplicaciones de escritorio pueden dividirse en wpf y windows forms y permiten crear interfaces de usuario para aplicaciones de escritorio (Younas, Jawawi, Mahmood & Ahm, 2019). Una vez identificados los tipos de desarrollo se relacionan con los patrones definidos.

3.5. Clasificación de los patrones con sus respectivos tipos de desarrollo

La figura 6 muestra la relación de patrones con los tipos de desarrollo, vista desde el concepto de algunos autores.

Autor	Patrón			
	MVP	MVC	Microservicios	Arquitectura en la nube
[24]	DM	DM		
[25]		DM		
		AW		
		AE		
[26] [21]		AE		
		AW		
[27][28]		DM		
[29]		AE		
[30][31]	AW			
[26][30]			AW	AW

DM	Dispositivos móviles
AW	Aplicaciones web
AE	Aplicaciones de escritorio

Figura 6 – Clasificación de los patrones y su relación con los tipos de desarrollo.  
Fuente (Elaboración propia)

En la figura 7 se representa la relación entre los patrones y los tipos de desarrollo de acuerdo con lo expuesto por los autores.

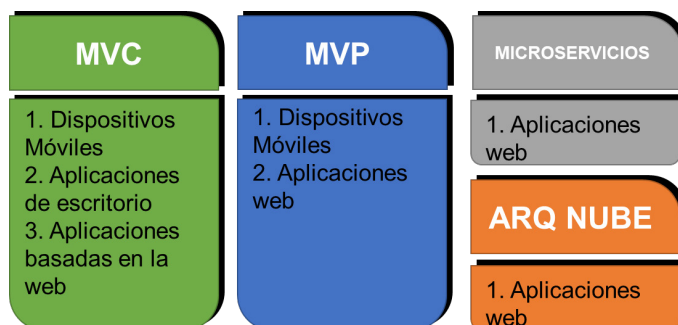


Figura 7 – Tipos de desarrollo y su relación con patrones clasificados.  
Fuente (Elaboración propia)

## 4. Resultados

### 4.1. Estructura y esquematización del marco de trabajo

Basado en la literatura y en el desarrollo de cada una de las fases previas, se construyó un conjunto de reglas que permiten estructurar el marco de trabajo:

Regla1: Si Tipo de desarrollo seleccionado es web, y características seleccionadas son mantenibilidad, rendimiento y flexibilidad, el patrón recomendado es MVC.

Regla2: Si Tipo de desarrollo seleccionado es web, y las características son mantenibilidad, rendimiento y seguridad, el patrón recomendado es MicroServicios.

Regla3: Si Tipo de desarrollo seleccionado es móvil, y las características son modificabilidad, rendimiento y testeabilidad el patrón recomendado es MVP.

Regla 4: Si Tipo de desarrollo seleccionado es escritorio, y las características son rendimiento, mantenibilidad y velocidad el patrón recomendado es MVC.

Regla 5: Si Tipo de desarrollo seleccionado es servicios web, y las características son rendimiento, mantenibilidad y seguridad el patrón recomendado es Microservicios.

Regla 6: Si Tipo de desarrollo seleccionado es servicios web, y las características son seguridad, y flexibilidad el patrón recomendado es en una arquitectura en la nube.

De acuerdo con las reglas planteadas, la figura 8 representa los pasos que el usuario podrá seguir en el prototipo final, para que se le indique el patrón de arquitectura adecuado para el proyecto de software a desarrollar.



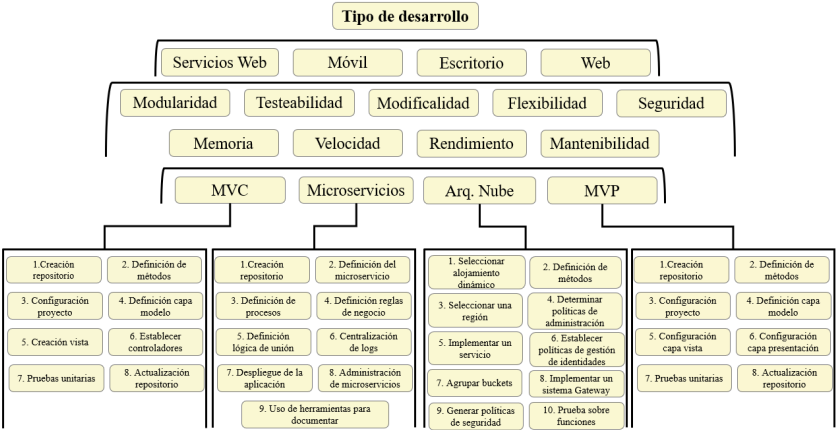


Figura 8 – Marco de trabajo  
Fuente (Elaboración propia)

5. Aplicación del marco de Trabajo- caso de estudio

Para la prueba del marco propuesto, se tomó un caso práctico del sector productivo, el cual utiliza el prototipo diseñado para seleccionar la arquitectura adecuada de acuerdo con el requerimiento base, el cual consiste en el desarrollo de una aplicación Web que permita ser accedida desde cualquier navegador, que tenga una página de validación de usuarios, y presentar el catálogo de productos de la compañía. Debe tener buen rendimiento, ser veloz y mantenible, debido al acceso masivo de usuarios.

5.1. Navegación del usuario con el prototipo del marco

Como primer paso el usuario debe loguearse para validar su cuenta de usuario, e ingresar a la interfaz. Luego la aplicación lo guiará por cada uno de los pasos para indicarle el patrón más adecuado según el requerimiento base:

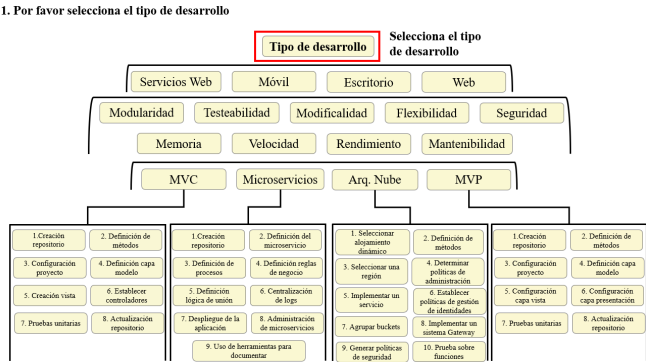


Figura 9 – Inicio proceso de selección  
Fuente (Elaboración propia)

Partiendo del requerimiento base propuesto en la simulación de caso, el usuario selecciona la opción web, de las 4 propuestas por el prototipo.

Figura 10 – Tipo de desarrollo  
Fuente (Elaboración propia)

Luego de seleccionar el tipo de desarrollo, el software le resalta indicando los pasos que debe seguir y los pendientes para obtener el resultado.

## 2. Ir al módulo para seleccionar las variables

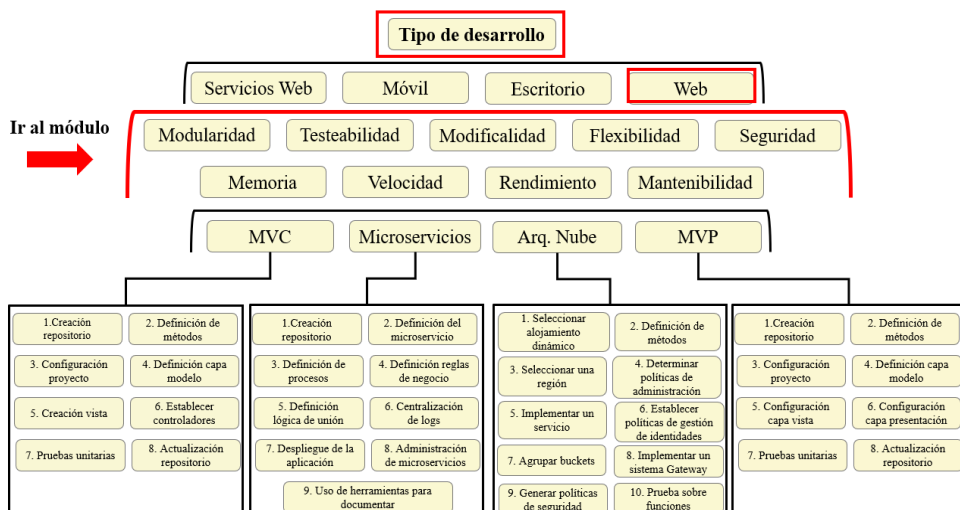


Figura 11 – Módulo marco de trabajo paso 1  
Fuente (Elaboración propia)

En este paso, el usuario selecciona como mínimo 3 variables, que el considere importantes en su implementación.

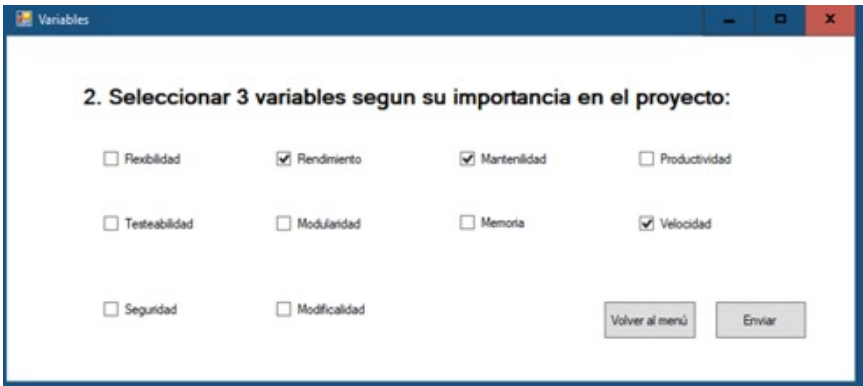


Figura 12 – Selección de las variables que requiere para la selección.  
Fuente (Elaboración propia)

Luego de enviar podrá ver los resultados basado en la recolección de información.

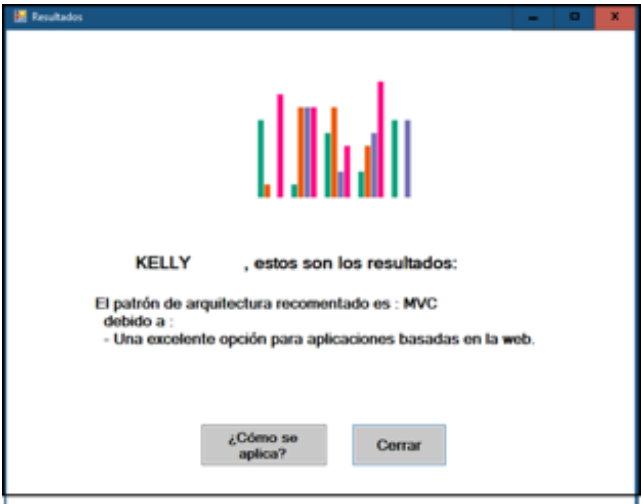


Figura 13 – Resultados basados en los datos ingresados.  
Fuente (Elaboración propia)

Posteriormente debe seleccionar la opción “Cómo se aplica” la cual le permitirá ver la figura 14 que a continuación se presenta:

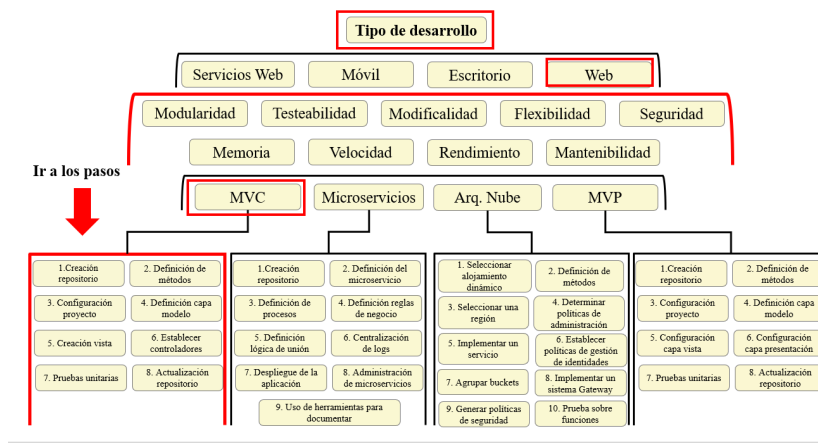


Figura 14 – Módulo del marco de trabajo paso 2  
Fuente (Elaboración propia)

De acuerdo con la figura 14 hay un paso opcional, y corresponde a las recomendaciones para la aplicación del patrón, dichos pasos incluyen buenas prácticas y como se podría desarrollar ese patrón sin especificar lenguajes o tecnologías. Si se selecciona la opción “ver los pasos” se indicaran los pasos generales, y el icono o imagen para acceder a cada uno de ellos. De igual forma dispone de la opción de imprimir las plantillas o información que requiera de los pasos recomendados. Véase figura 15.



Figura 15 – Pasos MVC  
Fuente (Elaboración propia)

## 6. Conclusiones

Actualmente hay muchas compañías que se centran en generar un producto funcional sin tener en cuenta calidad, mantenibilidad o las consecuencias que puede traer si un proyecto no se construye con buenas bases; además, muchos de los desarrolladores no cuentan con el tiempo para realizar una investigación a partir de los requerimientos, y menos cuando se cuenta con poco tiempo para desarrollarlo, se tiende a empezar a codificar sin saber el diseño ni que arquitectura es la correcta para manejar.

Luego de una extensa revisión de la literatura y encuestas a un grupo de expertos de las diferentes compañías se concluyó que los patrones de arquitectura más usados actualmente en los proyectos de software son: Arquitectura en la nube, MVC, Microservicios y MVP, por ende, se convirtieron en el soporte para proponer el marco.

Basado en la información recolectada en cada una de esas etapas y fases que se realizaron durante la investigación, se establecieron unas reglas que son el insumo para la funcionalidad y toma de decisión del marco de trabajo a través del prototipo propuesto. Estas reglas le permitieron al usuario a través del requerimiento base, contar con una guía y obtener recomendación del patrón de arquitectura que mejor se ajusta con los requisitos del negocio y del sistema.

El desarrollo del marco de trabajo, le permite al desarrollador o arquitecto de software obtener la estructura correcta antes de la codificación del proyecto y de esta manera reducir tiempos, evitar reprocesos y asegurar la mantenibilidad, escalabilidad y calidad del producto a desarrollar.

Este trabajo es una solución a la ausencia de arquitectura de software, al dispersamiento de información o documentación no muy clara sobre la misma, está orientada en apoyar a los arquitectos o desarrolladores de software para conocer la dirección, y la estructura de cómo debe manejar su proyecto. El marco de trabajo aporta a la población que relaciona el problema, permitiendo reducir tiempos, costos y reprocesamientos que son el reflejo de con bases débiles y poco mantenibles.

## Referencias

- Abdul, M. and Ibtisam, R.(2018). "MVC Architecture: A detailed insight to the modern web applications development" *Semantic scholar*, vol. 1, pp. 1-7.
- Ari, N. and Mamatzarova, N. (2014). "Programming languages," 11th International Conference on Electronics, Computer and Computation (ICECCO), Abuja, 2014, pp. 1-8.
- Camargo, A.d. (2016). "An architecture to automate performance tests on microservices" *ACM*, vol. 1, pp. 422-429.
- Guerrero, C. A. and Londoño, J. M. (2014). "Estudio comparativo de marcos de trabajo para el desarrollo software orientado a aspectos," *SciELO*, vol. 8, pp. 13-29.
- Harper, K. E. and Dagnino, A. (2014). "Agile Software Architecture in Advanced Data Analytics," IEEE/IFIP Conference on Software Architecture, Sydney, NSW, 2014, pp. 243-246.

- Harper, K. E. and Zheng, J. (2015). “Exploring Software Architecture Context,” 12th Working IEEE/IFIP Conference on Software Architecture, Montreal, QC, 2015, pp. 123-126.
- Hussain, S. and Keung, J. and Sohail, M. and Khan, A. & Ilahi, M.(2019). “Automated framework for classification and selection of software design patterns”. *Applied Soft Computing*, vol 75, pp. 1-20.
- Li, X. and Chang, Pen, D. H. and Zhang, X. and Liu, Y. and Yao, Y. (2015). “Application of MVVM design pattern in MES,” *IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, Shenyang, pp. 1374-1378.
- Lou, T.(2016). “A comparison of Android Native App Architecture MVC, MVP and MVVM” *Semantic scholar*, vol. 1, pp 1-7.
- Richards, M.(2016). “Microservices vs. Service-Oriented Architecture “, *IEEE*, vol 1, pp. 16.
- Sabry, A.E. (2015). *Decision Model for Software Architectural Tactics Selecton Based on Quality Attributes requirenments*, *Science direct*, vol. 65, pp. 422-431.
- Upadhyay, N. (2016). “SDMF: Systematic Decision-making Framework for Evaluation of Software Architecture,” *Science direct*, vol. 91,pp.599-608.
- Upadhyay, N.(2016). “SDMF: Systematic decision-making framework for evaluation of software architecture”. *Procedia computer science*, vol 91, pp. 599-608.
- Yang, D. and Wei, H. and Zhu, Y. and Li, P. and Tan, J.-C. (2018). “Virtual Private Cloud Based Power-Dispatching Automation System—Architecture and Application,” *IEEE*, vol. 1, pp. 2,3.
- Younas, M. and Jawawi, D. N. A., and A Mahmood, A.K., and Ahm, M.N. (2019). “Agile Software Development Using Cloud Computing: A Case Study,” *IEEE*, vol. 8, pp. 4475 – 4484.