Evaluando la Facilidad de Aprendizaje de Frameworks MVC en el Desarrollo de Aplicaciones Web

Assessing the Ease of Learning of MVC Frameworks in the Development of Web Applications

[1]Libardo Pantoja, [2]César Pardo

Departamento de Sistemas, Facultad de Ingeniería Electrónica y Telecomunicaciones, Universidad del Cauca. Popayán, Colombia

[1]wpantoja@unicauca.edu.co
[2]cpardo@unicauca.edu.co

Recibido: 12/08/2015 • Aprobado: 15/11/2015

Resumen

Uno de los aspectos relevantes al momento de elegir un framework de desarrollo de software es determinar la curva de aprendizaje que requiere. En los proyectos de desarrollo de software, en general, los desarrolladores disponen de poco tiempo para entregar un producto estable y usualmente requieren de frameworks de desarrollo que tengan una curva de aprendizaje baja. Actualmente, existen diversos frameworks para soportar el desarrollo web; sin embargo, elegir el más adecuado puede ser una tarea compleja, debido a que los criterios de selección pueden ser diversos, poco claros e incluso inexistentes. En este sentido, en este artículo se lleva a cabo un análisis de diferentes frameworks de desarrollo MVC Web para determinar cuáles son los más convenientes basados en un criterio de selección, el cual tiene en cuenta el tiempo que requieren durante la curva de aprendizaje.

Palabras Claves: framework, MVC, desarrollo web, curva de aprendizaje.

Abstract

One of the relevant aspects at the time of choosing a software development framework is to determine the learning curve required. In software development projects, in general, developers have little time to deliver a stable product and usually require development frameworks that have a low learning curve. Currently, there are different frameworks to support web development; however, choose the most appropriate can be a complex task, since the selection criteria can be diverse, unclear and even non-existent. In this sense, in this article is carried out an analysis of different frameworks of MVC Web development to determine which are most suitable, based on selection criteria, which takes into account the time required for the learning curve.

Keywords: *framework, learning curve, MVC, web application development.*

I. Introducción

El desarrollo de aplicaciones de software es una labor que les exige a los equipos de desarrollo diferentes capacidades y conocimientos específicos a nivel de lenguajes de programación, entornos y/o plataformas, modelos de desarrollo, procesos, entre otros. Con relación al proceso, se debe considerar el tiempo y el esfuerzo que demanda tanto la obtención de requisitos, como las actividades relacionadas con el análisis, diseño, implementación y pruebas [1].

Para el caso de las aplicaciones web, el proceso de desarrollo es más complejo porque se deben considerar las características particulares que las diferencian de las aplicaciones de escritorio. Organizándolas en tres dimensiones, es posible nombrar las relacionadas con el producto, uso y desarrollo. Algunas de estas características según [2], son: la naturaleza no lineal del hipertexto. Si no se tienen en cuenta patrones de interacción adecuados, la interfaz web puede desorientar al usuario y generar sobrecarga cognitiva; presentan contenido (por medio de texto, tablas, imágenes y video) que debe permanecer actualizado; la presentación del contenido debe tener un diseño estético, lo cual exige un buen uso de HTML, hojas de estilo en cascada y habilidades desde el *front-end*; requieren un despliegue multiplataforma; la globalidad y amplia disponibilidad generan requisitos no funcionales de seguridad y desempeño difíciles de tratar y las aplicaciones web se deben adaptar a los cambios continuos de tecnologías y estándares.

Las características mencionadas anteriormente son la razón por la cual diversos conceptos, métodos, técnicas y herramientas de las aplicaciones tradicionales han tenido que ser adaptadas a las aplicaciones web, por lo que se habla de un área multidisciplinar llamada: *Ingeniería Web* [2]. Además, toda esta situación conlleva a que desarrollar una aplicación web requiera más trabajo que una aplicación tradicional.

A pesar de la complejidad de las aplicaciones web, estudios empíricos muestran que los tiempos de desarrollo de las mismas son extremadamente cortos; por lo general, no duran más de seis meses y, en promedio, es menor a tres meses [3][4]. Por ello, si se usan procesos iterativos, las iteraciones son igualmente cortas (de días o semanas). Por lo tanto, elegir un *framework* que facilite el desarrollo de la aplicación web, puede ser un elemento determinante de éxito o fracaso.

Las ventajas de usar un *framework* a la hora de realizar un proyecto son diversas, entre otras, se disminuye el tiempo de creación de las aplicaciones, facilita el mantenimiento del código y hace uso de patrones. El patrón más utilizado por casi todos los *frameworks* es el conocido como Modelo Vista Controlador (MVC).

El patrón MVC puede implementarse sin la necesidad de utilizar un *framework*; no obstante, y a diferencia de aplicarlo de forma manual, el *framework* obliga al desarrollador a utilizarlo, creando de esta forma un código mucho más robusto [5]. De esta manera, se evita el "código spaghetti", el cual se caracteriza por agregar funcionalidades en capas que no corresponden [6][7][8], lo que incide negativamente sobre el código fuente y el cumplimiento de las características de calidad del producto en relación con la mantenibilidad, especialmente sobre algunas de sus sub-características como: analizabilidad, modificabilidad y testeabilidad [9].

Como ya se ha explicado, el uso de *frameworks* de desarrollo trae grandes beneficios; además, existe un gran número de *frameworks* que pueden utilizarse. El abanico de opciones es tan grande que en ocasiones es bastante complejo decidir cuál es el más adecuado de acuerdo con el tipo de proyecto o producto que se desea desarrollar. Elegir un *framework* de desarrollo de software no es una tarea sencilla. Se deben considerar diferentes variables. Entre estas, la más importante, es el lenguaje de

programación por utilizar, pues a partir de este, es posible tener en cuenta otras variables como: características del proyecto, conocimientos previos del equipo en el uso de la tecnología, existencia de proyectos que hayan sido desarrollados con el *framework* por elegir, nivel de madurez del *framework*, soporte, curva de aprendizaje, tipo de licencia (libre o de pago), entre otras [5].

Entre las variables mencionadas, la curva de aprendizaje resulta una de las más importantes, en especial, porque la gran mayoría de desarrolladores pueden desistir del uso de un *framework* cuando esta no es lo suficientemente corta para el desarrollo de sus proyectos [5]. La facilidad de aprendizaje de un *framework* puede incidir positiva o negativamente sobre otras variables relacionadas con el tiempo, esfuerzo y dinero, variables cruciales para la entrega temprana y oportuna de los proyectos. En este sentido, y dada la importancia del nivel de complejidad en el aprendizaje de los *frameworks* de desarrollo, se presentan en este artículo los resultados obtenidos al llevar a cabo un experimento empírico que permitió evaluar en total 20, teniendo en cuenta el patrón de diseño MVC. La pregunta de investigación que intenta resolver este estudio de caso es: *¿qué frameworks de desarrollo web MVC son los más convenientes teniendo en cuenta la variable del tiempo o curva de aprendizaje como criterio para su selección?*

Además de la presente introducción, este artículo está organizado de la siguiente manera. La sección 2 presenta los trabajos relacionados a través de un análisis del estado del arte; la sección 3 presenta la metodología de investigación utilizada; en la sección 4 se muestra de forma detallada el estudio de caso llevado a cabo para evaluar el tiempo de aprendizaje en diferentes *frameworks* de desarrollo con relación al patrón de diseño MVC y, finalmente, la sección 5 presenta las conclusiones y trabajos futuros.

II. Trabajos Relacionados

Después de llevar a cabo un análisis detallado del estado del arte, ha sido posible identificar algunos trabajos en los que se comparan diferentes *frameworks*. La tabla 1 describe de forma resumida algunos de estos:

TABLA I

Principales Características de Trabajos Relacionados

NO.	FRAMEWORKS	PRINCIPALES CARACTERÍSTICAS	REF.
1	HTML5, CSS y *JavaScript*	Se lleva a cabo una comparación de las características, ventajas, desventajas y limitaciones para los desarrolladores que tienen los *frameworks* analizados. Este trabajo permite a los desarrolladores facilitar la elección de una herramienta de trabajo que cumpla con los requerimientos del cliente y que optimice el trabajo de construcción de la aplicación web. Este trabajo se diferencia del proyecto planteado, en que no analiza *frameworks* de desarrollo MVC, sino *frameworks* que facilitan la escritura del código *front-end*.	[10]
2	*Tapestry y Wicket*	Se plantea un estudio comparativo para el desarrollo de aplicaciones web. Es un caso práctico del Instituto Particular San Gabriel de la ciudad de Riobamba de la provincia de Chimborazo (Ecuador). Se desarrollaron dos prototipos en cada uno de estos *frameworks* de estudio y se estableció una comparación entre estos. Este proyecto tiene en común que compara la curva de aprendizaje y la productividad, pero se diferencia, en que únicamente contrasta dos *frameworks* de desarrollo.	[11]
3	*Frameworks* MVC de Java	Se analizaron *frameworks* MVC de java para el desarrollo de aplicaciones web empresariales. Para el desarrollo de esta investigación se utilizó el método científico y el método descriptivo. Se escogieron parámetros como: producto, rendimiento, desarrollo, patrón de diseño, seguridad, cada uno con sus respectivas variables y gráficos estadísticos. Se diferencia del proyecto planteado, en que no analiza la facilidad de aprendizaje, sino que aborda otros parámetros. Además, se centra en *frameworks* de java.	[12]

NO.	FRAMEWORKS	PRINCIPALES CARACTERÍSTICAS	REF.
4	*Spring* MVC, *Struts2*, JSF, *Tapestry* y *Cocoon*	Se llevó a cabo un análisis mediante una encuesta en una página web dirigida a desarrolladores web. Con dicho análisis previo se buscó obtener los dos mejores *frameworks* que reflejen facilidad y suficientes características para uso efectivo, para luego terminar con un análisis más profundo entre *Spring* MVC y *Struts* 2 con los parámetros de manejo del Patrón MVC, otras características y, lo más importante, tiempo de desarrollo, que ayudó a la comprobación de la hipótesis. Observándose que *Spring* MVC obtuvo una calificación del 90% y *Struts*2 78.75%, se determinó que *Spring* MVC es el más adecuado para reducir el tiempo de desarrollo de aplicaciones Web. Nuevamente, se diferencia del proyecto planteado porque solo compara dos *frameworks* de Java.	[13]
5	*Frameworks* de PHP: Kumbia, Cake y Seagull	En este estudio de los *frameworks* de PHP, el resultado cuantitativo obtenido mediante la comparación de parámetros como, acceso a datos, seguridad, ajax, *skins*/temas y *plug-ins* para evaluar la eficiencia en el desarrollo fue: 68,40%(Regular) para *Kumbia*; 56,90%(Regular) para *Cake*; y 86,50%(Muy Bueno) para *Seagull*. Se concluye que el *framework* apropiado para trabajar es *Seagull* junto al *IDE Zend Studio*. La investigación se basó en el Método Científico General. Este proyecto tiene en cuenta varios aspectos y no se centra en la facilidad de aprendizaje.	[14]
6	Análisis de *Python* en *Django* y *Ruby on Rails*	Realiza el análisis de dos de las tecnologías de *framework* de software libre más populares en el mercado, con el propósito de determinar qué *framework* permite mejorar la productividad de desarrollo ágil de aplicaciones web. El análisis comparativo se basó en el Método Científico General, con base en el cual se efectuó la investigación. Ambas tecnologías fueron adaptadas a los mismos ambientes de desarrollo y pruebas, con técnicas de observación directa, de laboratorio y experimento. Los resultados cuantitativos que se obtuvieron mediante la comparación de parámetros como: patrón MVC, reutilización, seguridad de aplicación, madurez de producto e instalación, permitieron evaluar la productividad de desarrollo, cuyo resultado fue: *Django* 91,82% (Muy Bueno) y *Ruby on Rails* con 84,61% (Muy Bueno).	[15]

Por otro lado, en la web también se encuentra información acerca del *ranking* de algunos *frameworks* de desarrollo Web. Por ejemplo, en www.catswhocode.com es posible observar la clasificación de algunos *frameworks* por nombre y un resumen de sus principales características. Sin embargo, no se describe la manera como se obtuvo dicha clasificación, razón por la cual, este tipo de información no es muy confiable.

III. Metodología de Investigación

De acuerdo con [16], existen diversos tipos de metodologías de la investigación para la ingeniería de software, tales como: encuestas [17], experimentos controlados, quasi-experimentos, investigación-acción [17] y estudios de caso [18]. Por las características del trabajo realizado, se llevó a cabo un estudio de caso y la aplicación de encuestas, cuyo objetivo principal fue el de determinar qué *frameworks* son los más convenientes teniendo en cuenta, como criterio para su selección, la variable del tiempo o curva de aprendizaje de los individuos. Por las características de las encuestas, el estudio que se realizó es cuantitativo dado que se pudo obtener la opinión y el nivel de satisfacción de los individuos encuestados por medio de diferentes alternativas.

Teniendo en cuenta los dos enfoques de investigación elegidos, se establecieron, para su aplicación, un conjunto de actividades por seguir:

- Formulación de la pregunta de investigación.
- Descripción del estudio de caso.
- Diseño de la actividad académica.
- Unidades de análisis.
- Procedimiento de campo y la recolección de información.
- Ejecución de la actividad académica.
- Intervención en el estudio de caso: aplicación de una encuesta.
- Análisis de los resultados.
- Plan de validación y limitaciones del estudio de caso.

IV. Estudio de Caso

En esta sección se describe la experiencia obtenida durante el estudio de caso llevado a cabo para evaluar la facilidad de aprendizaje de los *frameworks* MVC más populares en el mercado.

A. Formulación de la pregunta de investigación

Para formular la pregunta de investigación se hizo una revisión bibliográfica buscando proyectos relacionados con *frameworks* MVC. Finalmente, después de revisar varios proyectos, y acorde con los objetivos de la asignatura, se decidió plantear la siguiente pregunta:

¿Qué *frameworks* MVC para desarrollo web existentes son los más convenientes bajo el criterio del tiempo o curva de aprendizaje?

B. Descripción del estudio de caso

La experiencia se llevó a cabo en un curso de Ingeniería Web con 33 estudiantes del programa de Ingeniería de Sistema de VII semestre de la Universidad del Cauca, durante el segundo semestre de 2014.

La asignatura de Ingeniería Web hace parte del plan de estudios del programa de Ingeniería de Sistemas de la Universidad. En ella se estudian conceptos, métodos, técnicas y herramientas para medir el costo efectivo del análisis, diseño, implementación, pruebas, operación y mantenimiento de aplicaciones Web de alta calidad. En este curso, los estudiantes ya cuentan con bases sólidas de construcción de software dado que, como pre-requisito, han cursado cursos de programación, bases de datos, sistemas operativos, estructuras de datos y cursos de ingeniería de software, entre otros.

Teniendo en cuenta las características mencionadas, en el marco de esta asignatura se abrió un espacio investigativo para analizar la facilidad de aprendizaje de varios *frameworks* de desarrollo MVC.

C. Diseño de la actividad académica

Uno de los objetivos de la asignatura de Ingeniería Web es trabajar un proyecto final en equipos de 4 o 5 personas. Para esto, cada equipo debe elegir un *framework* MVC para su proyecto. El docente en la primera semana de clase asigna una primera actividad exploratoria, en la cual se tienen en cuenta los siguientes aspectos:

Cada estudiante debe estudiar un *Framework* MVC y elaborar un tutorial de aprendizaje del mismo (instalación, configuración, algunos aspectos generales), en el que, además, se guíe al usuario, paso a paso, para que construya una sencilla aplicación que implemente un sencillo CRUD (Crear, Leer, Actualizar y Borrar del original en inglés*: Create, Read, Update and Delete*) de las siguientes tablas:

- **Empleado:** código, nombres, fecha de nacimiento, código del departamento donde labora, salario.
- **Departamento:** código, nombre del departamento. Los entregables son:
- Prototipo funcional del CRUD empleados.
- Tutorial paso a paso de instalación y creación del CRUD. Debe tener portada, introducción, tutorial, conclusiones o lecciones aprendidas y referencias.
- Breve socialización (10 minutos), en la que se muestre la aplicación y una breve explicación general de la guía.

Para ejecutar esta actividad exploratoria, los estudiantes contaron con 5 semanas, con una disponibilidad de 5 horas semanales. Es la misma actividad para todos, pero cada estudiante trabaja con un *framework* diferente.

Para elegir el *framework*, se creó un foro en una plataforma virtual, y cada estudiante eligió el suyo. Los estudiantes exploraron en internet qué *frameworks* existen, sus características, ventajas, desventajas, y una vez elegido, reportaron su elección en el foro. Un *framework* ya elegido no puede ser estudiado por otro estudiante Fig. 1.

Fig. 1 Foro para elegir el framework por estudiar

Los *frameworks* estudiados, con sus versiones, se aprecian en la Tabla 2.

TABLA II
Frameworks MVC Seleccionados

NO.	FRAMEWORKS	LENGUAJE	VERSIÓN
1	*Asp.NetMVC*	C#	3
2	*Cake*	PHP	2.4.5
3	*CodeIgniter*	PHP	2.2.0
4	*Grails*	Java	2.4.2
5	*JSF*	Java	2.0
6	*Kohana*	PHP	3.3.2
7	*Kumbia*	PHP	1.0
8	*Laravel*	PHP	4.1
9	*Nette*	PHP	2.2.3
10	*Phalcon*	PHP	1.3.2
11	*Pyramid*	Python	1.4.6
12	*RubyOnRails*	Ruby	4.1.6
13	*Sails.js*	JavaScript	0.10
14	*Slim*	PHP	2.3.0
15	*Spring*	Java	3.6.1
16	*Symfony2*	PHP	2.5.5
17	*Tapestry*	Java	5.3.7
18	*Web2Py*	Python	2.9.10
19	*Yii*	PHP	1.1.15
20	*Zan*	PHP	2.6.6
21	*Zend*	PHP	2.3.3

D. Unidades de análisis

Las unidades de análisis en este estudio de caso son los *frameworks* MVC más populares en el mercado (Tabla 2). Estos fueron evaluados con relación a factores como el esfuerzo de aprendizaje, productividad y tamaño (referente a la cantidad de líneas de código producidas).

E. Procedimiento de campo y la recolección de información

El procedimiento de campo y la recolección de la información en el estudio de caso, se llevó a cabo mediante la aplicación de encuestas.

F. Ejecución de la actividad académica

Una vez elegido el *framework*, los estudiantes comenzaron el proceso de aprendizaje, valiéndose de la documentación oficial, videos, blogs, foros, etc. De antemano, se les solicitó que llevaran el control de las horas que le dedicarían a esta actividad. Al final, los estudiantes debían entregar el manual solicitado, el prototipo funcional y la socialización de 10 minutos.

G. Intervención en el estudio de caso: aplicación de las encuestas

Para medir el tiempo de aprendizaje de cada *framework*, se diseñó y aplicó una encuesta a cada estudiante. La encuesta fue el instrumento principal de recolección de información y su finalidad era tomar los datos relacionados con la facilidad de aprendizaje de cada *framework*.

La encuesta estuvo conformada de varias partes (encuestas):

- Primera parte: datos básicos del estudiante y datos genéricos del *framework* Fig. 2.

Encuesta

La presente encuesta tiene como finalidad estudiar las características relacionadas con la facilidad de aprendizaje de los fraworwks MVC para desarrollo web. Gracias por sus aportes.

Nombres y Apellidos:	
Fecha de la encuesta (dd-mm-aaa):	
Framework MVC:	
Lenguaje de Programación (php, java, python)	
Versión del framework	
Fecha de liberación de esta versión (dd-mm-aaa)	
Sobre qué sistema operativo trabajó (Window 7, 8, Ubuntu, etc)	
Documentación en español? Si/No	
Sitio web del Framework:	
Resumen del framework: (Máximo cinco lineas)	

Fig. 2 Encuesta - Datos básicos del estudiante y del framework

- Segunda parte: las preguntas de la 1 hasta la 6, y tiene por finalidad recolectar tiempos, tamaños y logros alcanzados Fig. 3.

1. ¿Aproximadamente cuántas horas de estudio le dedicó al framework (instalación, configuración, hacer correr los ejemplos básicos)? _____
2. ¿Aproximadamente cuántas horas le dedicó a la creación del CRUD de empleados/departamentos? _____
3. ¿Aproximadamente cuántas horas le dedicó a la creación de la guía? _____
4. ¿Es el primer framework que aprende? Si ___ No ____
5. ¿Qué dificultades considerables tuvo con el aprendizaje del framework y cómo las solucionó? Describa brevemente cada una (puede aumentar las filas que requiera):

   _____
   _____
   _____
   _____
   _____
   _____
   _____

6. De las funcionalidades del CRUD empleados evalúe el % de logro (0 a 100):
   a) Agregar empleados (incluido el combo departamentos y validaciones): ___ %
   b) Modificar empleados (incluido el combo departamentos y validaciones): ____ %
   c) Eliminar empleados (incluida la confirmación): ____%
   d) Buscar empleados: ___%
   e) Paginación: ___ %
   f) Plantilla HTML y CSS personalizada: ___ %
7. ¿Cuantas líneas de código fuente produjo su aplicación (únicamente modelos y controladores)?
   Modelos: _____
   Controladores: _____
   ***Nota**: Por favor, no contar comentarios, espacios ni llaves. Por ejemplo, para el siguiente código, se contarían 4 lineas de código fuente (las que aparecen en negrita):*

Fig. 3 Encuesta - Datos de tiempos, tamaños y logros alcanzados

- Tercera parte: hace una valoración de las características más relevantes del *framework* en una escala de Excelente, Bueno, Neutro, Regular y Deficiente Fig. 4.

7. ¿Evalúe las siguientes características de su framework (Marque con una X)?

	Excelente	Bueno	Neutro	Regular	Deficiente
Facilidad de aprendizaje					
Documentación					
Instalación					
Legibilidad del código fuente					
Herramientas de generación de código					
Ejemplos que vienen en la documentación					
Comunidad de soporte					
Herramientas de depuración					
Facilidad del uso del patrón MVC					

Fig. 4 Encuesta - Evaluando las características del framework

- Cuarta parte: en las preguntas 8 y 9 se evalúa de manera cuantitativa y general el *framework*, y las preguntas 10 y 11 tienen por finalidad evaluar las habilidades de aprendizaje del estudiante Fig. 5.

8. Recomienda el uso de este framework, Si __ No ___ ¿Por qué?

_____

_____

_____

9. Globalmente en una escala entre 1 y 100, ¿Cómo evalúa el framework? _____

10. ¿Cómo considera que es su habilidad para aprender nuevas tecnologías en software (lenguajes de programación, entornos de desarrollo, plataformas) (Marque con una X)?

Muy Alta	Alta	Media	Baja	Muy Baja

11. ¿Qué nivel de disciplina tiene Usted a la hora de aprender nuevas tecnologías en software (lenguajes de programación, entornos de desarrollo, plataformas) (Marque con una X)?

Muy Alta	Alta	Media	Baja	Muy Baja

Fig. 5 Encuesta - Habilidades del estudiante

Las encuestas fueron enviadas a cada estudiante y posteriormente recolectadas mediante correo electrónico.

H. Análisis de resultados

Una vez aplicadas las encuestas, se procesaron los datos, lo cual se llevó a cabo teniendo en cuenta que agrupando varios conjuntos de preguntas se podrían analizar diversos aspectos. A continuación se muestran los resultados obtenidos:

***1)* Análisis de esfuerzo:** las preguntas 1, 2, 12, 13 y 14 dan un indicio del esfuerzo dedicado al aprendizaje del *framework*, lo cual involucra estudiar la documentación del mismo y realizar una aplicación:

P1: ¿Aproximadamente cuántas horas de estudio le dedicó al *framework* (instalación, configuración, compilar correctamente los ejemplos básicos)?

P2: ¿Aproximadamente cuántas horas le dedicó a la creación del CRUD de empleados/departamentos?

P11: ¿Cómo considera que es su habilidad para aprender nuevas tecnologías en software (lenguajes de programación, entornos de desarrollo, plataformas)?

P12: ¿Qué nivel de disciplina tiene usted a la hora de aprender nuevas tecnologías en software (lenguajes de programación, entornos de desarrollo, plataformas)?

P13: ¿Cómo es su habilidad para leer documentación en inglés?

Los resultados de estas preguntas se muestran en la Tabla 3.

TABLA III
Evaluación del Esfuerzo: Horas de Estudio

NO.	*FRAMEWORK*	P1	P2	TOTH	P11	P12	P13
1	*Symfony2*	3	2	5	Alta	Media	Media
2	*Spring*	3	4	7	Alta	Alta	Media
3	*Phalcon*	4	4	8	Muy Alta	Muy Alta	Alta
4	*JSF*	8	2	10	Alta	Alta	Alta
5	*Cake*	8	3	11	Alta	Media	Baja
6	*Grails*	12	1	13	Alta	Alta	Alta
7	*Web2Py*	5	10	15	Media	Alta	Muy Alta
8	*Zend*	12	3	15	Alta	Media	Media
9	*Yii*	12	4	16	Muy Alta	Muy Alta	Muy Alta
10	*CodeIgniter*	7	12	19	Alta	Alta	Baja
11	*Asp.NetMVC*	8	12	20	Alta	Alta	Baja
12	*Pyramid*	13	7	20	Alta	Alta	Baja
13	*RubyOnRails*	20	2	22	Alta	Media	Baja
14	*Kumbia*	8	15	23	Alta	Alta	Baja
15	*Nette*	21	15	36	Alta	Media	Muy Baja
16	*Laravel*	24	16	40	Alta	Alta	Alta
17	*Kohana*	15	25	40	Alta	Alta	Media
18	*Zan*	24	20	44	Alta	Alta	Media
19	*Tapestry*	50	20	70	Media	Alta	Media
20	*Slim*	72	28	100	Alta	Alta	Media

Según la Tabla 3, los cinco *frameworks* que tienen menor tiempo de aprendizaje son*: Symfony2*, *Spring*, *Falcon*, JSF y *Cake*. Sin embargo, se debe considerar las habilidades del estudiante, pues un estudiante con una capacidad muy alta de aprendizaje en tecnologías, puede emplear mucho menos tiempo que otro con una capacidad menor. Las preguntas P11, P12 y P13 permiten detectar dichas habilidades. Por otro lado, se puede apreciar que *Tapestry* y *Slim* son los que tienen mayor tiempo de aprendizaje.

***2)* Productividad:** la pregunta 6 indaga qué tan productivo es el uso del *framework* en la implementación de una aplicación web:

De las funcionalidades del CRUD empleado evalúe el porcentaje (%) de logro (0 a 100).

a) Agregar empleados (incluido los combos departamentos y validaciones).
b) Modificar empleados (incluido los combos departamentos y validaciones).
c) Eliminar empleados (incluida la confirmación).
d) Buscar empleados.
e) Paginación.

La Tabla 4 muestra los *frameworks* ordenados por productividad, así como, las preguntas 11, 12 y 13 muestran las habilidades del estudiante. Se puede apreciar que los *frameworks* Yii, Web2Py, Asp.NetMVC y Kumbia tuvieron una productividad del 100% (se implementaron todas las funcionalidades requeridas). En segundo lugar, están los *frameworks* Laravel, *Zend*, y *Cake* que permitieron implementar el 92% de las funcionalidades.

TABLA IV
Evaluación de la Productividad

NO.	FRAMEWORK	P6	P11	P12	P13
1	Yii	100 %	Muy Alta	Muy Alta	Muy Alta
2	Web2Py	100 %	Media	Alta	Muy Alta
3	Asp.NetMVC	100 %	Alta	Alta	Baja
4	Kumbia	100 %	Alta	Alta	Baja
5	Laravel	92 %	Alta	Alta	Alta
6	Zend	92 %	Alta	Media	Media
7	Cake	92 %	Alta	Media	Baja
8	Phalcon	88 %	Muy Alta	Muy Alta	Alta
9	Sails.js	82 %	Muy Alta	Alta	Media
10	Grails	80 %	Alta	Alta	Alta
11	Symfony2	80 %	Alta	Media	Media
12	Nette	76 %	Alta	Media	Muy Baja
13	Pyramid	76 %	Alta	Alta	Baja
14	Kohana	74 %	Alta	Alta	Media
15	JSF	68 %	Alta	Alta	Alta
16	Tapestry	68 %	Media	Alta	Media
17	RubyOnRails	68 %	Alta	Media	Baja
18	Zan	60 %	Alta	Alta	Media
19	Spring	57 %	Alta	Alta	Media
20	CodeIgniter	54 %	Alta	Alta	Baja
21	Slim	50 %	Alta	Alta	Media

***3)* Tamaño:** la pregunta 7 indaga el tamaño de la aplicación. Se podría suponer que entre menos líneas de código requiere el *framework* es más fácil de aprender. La Tabla 5 muestra los *frameworks* ordenados por tamaño.

TABLA V
Evaluación del Tamaño

NO.	FRAMEWORK	P6:PRODUCTIVIDAD	P7:TAMAÑO
1	*Zan*	60 %	19
2	*Grails*	80 %	40
3	*Kohana*	74 %	49
4	*Sails.js*	82 %	54
5	*Slim*	50 %	88
6	*Kumbia*	100 %	92
7	*Nette*	76 %	100
8	*Symfony2*	80 %	100
9	*Laravel*	92 %	102
10	*RubyOnRails*	68 %	107
11	*CodeIgniter*	54 %	116
12	*Cake*	92 %	135
13	*Pyramid*	76 %	138
14	*Asp.NetMVC*	100 %	149
15	*Yii*	100 %	183
16	*Spring*	57 %	223
17	*Zend*	92 %	250
18	*Phalcon*	88 %	263
19	*JSF*	68 %	688
20	*Web2Py*	100 %	852
21	*Tapestry*	68 %	

Según la Tabla 5, los cinco *frameworks* con menor tamaño en líneas de código son: *Zan, Grails, Kohana, Sails.js, y Slim*. Sin embargo, se debe tener en cuenta que no todos los *frameworks* alcanzaron la misma productividad (% de implementación de todas las funcionalidades). Por lo tanto, tomando únicamente los que alcanzaron el 100 % de la implementación de las funcionalidades, se observa que los mejores serían Kumbia, ASP.NET MVC y Yii, los cuales produjeron un tamaño relativamente igual (entre 92 y 182 líneas de código). Además, se puede apreciar que el *framework* que mayor número de líneas de código produjo (a pesar que solo implementó el 68 % de las funcionalidades), es Tapestry con 1800.

4)* Valoración general:** la pregunta 10, Globalmente en una escala entre 1 y 100, *¿cómo evalúa el framework?,** es interesante pues permite tener un concepto global de la experiencia entre el estudiante y el *framework*. La Tabla 6 muestra los *frameworks* ordenados según este criterio. Se puede apreciar que los mejores *frameworks,* con una valoración de 100 puntos, son: Yii, Web2Py y Asp.NetMVC. Luego siguen con valores igualmente altos (entre 90 y 95 puntos): JSF, Kumbia, RubyOnRails, Zan, CodeIgniter y Symfony2.

TABLA VI
Evaluación Global

NO.	*FRAMEWORK*	P10
1	*Yii*	100 %
2	*Web2Py*	100 %
3	*Asp.NetMVC*	100 %
4	*JSF*	95 %
5	*Kumbia*	93 %
6	*RubyOnRails*	90 %
7	*Zan*	90 %
8	*CodeIgniter*	90 %
9	*Symfony2*	90 %
10	*Phalcon*	85 %
11	*Cake*	81 %
12	*Laravel*	80 %
13	*Grails*	80 %
14	*Spring*	80 %
15	*Kohana*	80 %
16	*Pyramid*	75 %
17	*Zend*	70 %
18	*Sails.js*	70 %
19	*Slim*	60 %
20	*Tapestry*	60 %
21	*Nette*	60 %

***5)* Características generales:** la pregunta 8 evalúa características generales de los *frameworks*: facilidad de aprendizaje (APREN), documentación (DOC), instalación (INST), legibilidad del código fuente (LEGI), herramientas de generación de código (HERRAM), ejemplos que vienen en la documentación (EJEM), comunidad de soporte (COMUN), herramientas de depuración (DEBUG), facilidad del uso del patrón MVC.

Cada criterio se valora en una escala de: Excelente, Bueno, Neutro, Regular y Deficiente. Se ha valorado el conjunto de estas características dando un valor de Excelente=4, Bueno=3, Neutro=2, Regular=1 y Deficiente=0, sacando en una columna la sumatoria respectiva (TOT). En la Tabla 7 se presentan los resultados.

Según los datos de la Tabla 7, los cinco mejores *frameworks* (sobre 30 puntos) son: *Web2Py, Cake, Asp.NetMVC, JSF* y *Symfony2*. Sin embargo, con valores cercanos a 30 puntos, están los *frameworks: Kumbia,Yii y Kohana.*

TABLA VII

Evaluación de las Características Generales

FRAMEWORK	APREN	DOC	INST	LEGI	HERRAM	EJEM	COMUN	DEBUG	MVC	TOT
Web2Py	Excelente	Excelente	Excelente	Neutro	Excelente	Excelente	Excelente	Bueno	Excelente	32
Cake	Bueno	Excelente	Excelente	Bueno	Excelente	Bueno	Bueno	Bueno	Excelente	31
Asp.NetMVC	Excelente	Excelente	Bueno	Bueno	Bueno	Bueno	Bueno	Bueno	Excelente	30
JSF	Excelente	Excelente	Excelente	Regular	Excelente	Bueno	Excelente	Bueno	Bueno	30
Symfony2	Bueno	Bueno	Excelente	Bueno	Excelente	Bueno	Bueno	Excelente	Bueno	30
Kumbia	Excelente	Excelente	Excelente	Excelente	Deficiente	Excelente	Excelente	Neutro	Excelente	29
Yii	Excelente	Excelente	Excelente	Bueno	Excelente	Excelente	Bueno	Regular	Neutro	28
Kohana	Excelente	Bueno	Excelente	Bueno	Neutro	Neutro	Bueno	Excelente	Excelente	27
Phalcon	Excelente	Bueno	Bueno	Excelente	Excelente	Bueno	Neutro	Deficiente	Excelente	26
Grails	Regular	Excelente	Neutro	Bueno	Excelente	Excelente	Neutro	Bueno	Excelente	25
RubyOnRails	Bueno	Bueno	Neutro	Bueno	Excelente	Bueno	Bueno	Neutro	Bueno	24
Zan	Excelente	Excelente	Excelente	Bueno	Neutro	Neutro	Bueno	Neutro	Bueno	24
Nette	Bueno	Neutro	Bueno	Bueno	Bueno	Bueno	Bueno	Bueno	Neutro	23
Pyramid	Bueno	Bueno	Excelente	Bueno	Neutro	Bueno	Bueno	Neutro	Neutro	22
Laravel	Bueno	Bueno	Neutro	Bueno	Regular	Regular	Excelente	Neutro	Excelente	21
CodeIgniter	Bueno	Neutro	Bueno	Bueno	Neutro	Neutro	Neutro	Bueno	Excelente	20
Zend	Bueno	Regular	Bueno	Bueno	Bueno	Neutro	Regular	Neutro	Bueno	19
Spring	Bueno	Deficinte	Excelente	Bueno	Neutro	Regular	Neutro	Neutro	Bueno	17
Tapestry	Neutro	Neutro	Deficiente	Neutro	Bueno	Bueno	Bueno	Bueno	Neutro	16
Sails.js	Neutro	Neutro	Bueno	Bueno	Neutro	Regular	Neutro	Bueno	Neutro	15
Slim	Bueno	Neutro	Bueno	Bueno	Deficiente	Bueno	Regular	Deficiente	Deficiente	14

1. Plan de validación y limitaciones del estudio de caso

Con el objetivo de hacer frente a posibles amenazas y permitir un plan de validación lo mejor posible, se han considerado varios factores:

- El diseño del estudio de caso y el plan de recolección de los datos fueron comparados con la lista de chequeo para estudios de caso en la ingeniería de software propuesto por [19].

- Por lo que respecta a la validez del constructo, se utilizaron múltiples fuentes de evidencia, incluyendo encuestas, la observación del participante y archivos de documentación. La evidencia se obtuvo de las actividades realizadas, por lo que cada uno de los participantes desempeñó las funciones específicas que se les asignó.

- En cuanto a la validez interna, ha sido posible determinar que la aplicación de las encuestas en el estudio de casos ha permitido satisfacer las necesidades identificadas. También se han tenido en cuenta los beneficios reportados por el estudio de caso.

Para la validez externa, el estudio de caso fue apoyado por un asesor. Las observaciones y lecciones aprendidas fueron recogidas con el objetivo de perfeccionar el protocolo y procedimiento de campo, con el fin de ser capaces de realizar una replicación en estudios de casos futuros.

Las limitaciones consideradas en el estudio de caso fueron:

- Dado que la población fue muy pequeña, no es una muestra representativa y de alguna manera afecta el poder generalizar los resultados. Por lo tanto, se hace necesario replicar el estudio de caso en un número de estudios de casos más grande que permita asegurar un análisis adecuado y subsecuentemente llevar a cabo la generalización de los resultados.

- El sesgo en el estudio de caso con relación a la falta de conocimientos en el uso de los *frameworks* analizados y la realización de las actividades, se pudo ver afectado por la observación y monitoreo continuo. La observación de los participantes puede resultar ventajosa cuando se trata de obtener cierta información que sería imposible de conseguir de otra manera. Sin embargo, el principal problema con esto, es la posibilidad de sesgo producido.

V. Conclusiones y Trabajo Futuro

En la actualidad, los proyectos de desarrollo Web tienen ciclos de desarrollo muy cortos; por esta razón, elegir un *framework* MVC que tenga un tiempo de aprendizaje bajo, es uno de los factores claves para abordar con éxito un proyecto. Elegir un *framework* MVC con baja curva de aprendizaje no es una tarea tan simple, debido a la cantidad de *frameworks* disponibles en la Web y a la cantidad de experiencias reportadas de manera informal por programadores que pueden dificultar dicha elección.

A lo largo de este artículo, se ha planteado una experiencia académica para probar distintos *frameworks* y determinar cuáles son los de menor tiempo de aprendizaje. En dicha experiencia se han reportado varias resultados por medio de tablas en las que se clasifican los mejores, teniendo en cuenta el análisis de esfuerzo, productividad, tamaño y algunas características particulares. Con base en los resultados obtenidos, se hace una contribución a la industria de software facilitando algunos criterios y resultados concretos al momento de elegir un *framework* de desarrollo.

Por otro lado, se aporta a la comunidad científica una experiencia y metodología detallada práctica para probar *frameworks* de desarrollo, experiencia que abarca la formulación de la pregunta de investigación, la planificación de la experiencia, seguida

de su ejecución, recolección de datos y análisis de los mismos. La experiencia, además, aprovecha el talento humano valioso de los estudiantes y de las aulas de clase, evitando el uso de recursos económicos para su ejecución. Por otro lado, se puede concluir y demostrar que las aulas de clase son un buen medio y laboratorio para llevar a cabo experiencias y realizar aportes investigativos y académicos importantes.

A futuro se puede llevar a cabo la misma experiencia con otros grupos de estudiantes, de tal forma, que se puedan contrastar los resultados obtenidos. La finalidad sería tener un listado de los *frameworks* MVC de desarrollo Web que involucren los más bajos tiempos de aprendizaje. Asimismo, sería conveniente extender la experiencia para probar atributos como la seguridad, el desempeño y la modificabilidad.

Agradecimientos

Este artículo fue elaborado gracias a la valiosa colaboración de los estudiantes del programa de Ingeniería de Sistemas, pertenecientes al curso de Ingeniería de Software 3, 2014-II de la Universidad del Cauca. También se contó con el apoyo y asesoría del PhD. Julio A. Hurtado Alegría. Adicionalmente, los profesores Libardo Pantoja y César Pardo agradecen la contribución de la Universidad del Cauca, donde trabajan como profesor titular y profesor asistente, respectivamente.

Referencias

[1] G. M. Villalobos, G. D. C. Sánchez, and D. A. B. Gutiérrez, "Diseño de framework web para el desarrollo dinámico de aplicaciones," Scientia et Technica, vol. 1, no. 44, pp. 178–183, 2010.

[2] G. Kappel, B. Proll, S. Reich, and W. Retschitzegger, "Web Engineering: The Discipline of Systematic Development of Web Applications". Wiley, 2006.

[3] McDonald and R. Welland, "A survey of web engineering in practice, department of computing sciencetechnical report r-2001-79," University of Glasgow, Scotland, 2001.

[4] A. McDonald and R. Welland, "Web engineering in practice," Proc. Of the 4th Workshop on Web Engineering (held in conjunction with the 10th international conference on WWW), Hong Kong, May 2001.

[5] A. Technologies, "Framework para el desarrollo ágil de aplicaciones," tech. rep., Acens Technologies, 2014.

[6] D. Riehle, "Framework design: A role modeling approach.," Softwaretechnik-Trends, vol. 20, no. 4, 2000.

[7] G. Krasner and S. Pope, "A description of the model-view-controller user interface paradigm in the smalltalk-80 system," Journal of Object Oriented Programming, vol. 1, no. 3, pp. 26–49, 1988.

[8] J. S. C. Garrido, "Arquitectura y diseño de sistemas web modernos," InforMAS, Revista de Ingeniería Informática del CIIRM, no. 1, 2004.

[9] ISO, "Systems and software engineering – systems and software quality requirements and evaluation (square) – system and software quality models:iso/iec 25010:2011," tech. rep., International Standards Organization, 2011.

[10] A. M. Valbuena Aponte, "Guía comparativa de frameworks para los lenguajes HTML 5, CSS y Javascript para el desarrollo de aplicaciones web," 2014.

[11] E. M. Jara Izurieta, "Estudio comparativo de los frameworks tapestry y wicket para el desarrollo de aplicaciones web. caso práctico: Instituto particular San Gabriel," 2014.

[12] T. P. Aguirre Buenaño and A. I. Moncayo Alvarez, "Análisis de frameworks MVC de Java para el desarrollo de aplicaciones Web empresariales. caso práctico: Sistema de bienestar politécnico," 2013.

[13] M. E. Morán Tapia, M. Saltos, and X. Fernanda, "Análisis de frameworks de presentación para el desarrollo de aplicaciones Web en Java, caso práctico: Gadpch," 2013.

[14] X. L. Zabala Hidalgo and C. L. Ochoa Iglesias, "Estudio de frameworks para php e integraci´on a una herramienta ide: Aplicado al portal web de la comunidad linux de la espoch," 2010.

[15] P. D. Cumba Armijos and B. A. Barreno Pilco, "Análisis de Python con Django frente a Ruby on rails para desarrollo ágil de aplicaciones Web. caso práctico: Dech," 2013.

[16] P. Runeson, and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," Empirical Software Engineering, vol. 14, no. 2, pp. 131–164, 2009.

[17] C. Robson, "Real World Research - A Resource for Social Scientists and Practitioner- Researchers," Blackwell Publishing, Malden, second edition, 2002.

[18] R. K. Yin, "Case Study Research: Design and Methods," Sage Publications Ltd, Newbury Park. 2003.

[19] M. Höst, and P. Runeson, "Checklists for software engineering case study research," In Proceedings of the First International Symposium on Empirical Software Engineering and Measurement (ESEM'07), pp. 479–481, Madrid, Spain. IEEE Computer Society. 2007.