Télématique
ISSN: 1856-4194
jlra431@gmail.com
Universidad Privada Dr. Rafael Belloso Chacín
Venezuela

Ortega, Dinarle; Guevara, María; Benavides, John
ELEMENTARY: UN FRAMEWORK DE PROGRAMACIÓN WEB
Télématique, vol. 15, núm. 2, julio-diciembre, 2016, pp. 144-171
Universidad Privada Dr. Rafael Belloso Chacín
Maracaibo, Venezuela

UNIVERSIDAD Privada DR. RAFAEL BELLOSO CHACÍN

Revista Electrónica de Estudios Telemáticos
TÉLÉMATIQUE

ELEMENTARY: UN FRAMEWORK DE PROGRAMACIÓN WEB

(Elementary: A Web Programming Framework)

Recibido: 07/09/2016 **Aceptado:** 17/05/2017

Ortega, Dinarle
Universidad de Carabobo, FACYT-UC, Venezuela.
dortega@uc.edu.ve

Guevara, María
Universidad de Carabobo, FACYT-UC, Venezuela.
mguevara@uc.eduve

Benavides, John
Universidad de Carabobo, FACYT-UC, Venezuela.
john.elric@gmail.com

RESUMEN

La complejidad del desarrollo de las aplicaciones web actuales, ha promovido el uso de estructuras de reutilización bien definidas conocidas como Frameworks de programación, muchos de ellos realizados en lenguaje PHP; la utilización de estos Frameworks, demanda una experticia adecuada del programador. Por ello, en este artículo se presenta Elementary, basado en la propuesta de patrón arquitectónico HGMVC (Hierarchical Generator-Model–View–Controller), dirigido a programadores inexpertos, ya que minimiza la complejidad de algunas actividades, tales como las configuraciones, instalación e instanciación para llevar a cabo un nuevo proyecto, proporcionando a la vez una documentación apropiada. Se realizaron varios tipos de evaluaciones de Elementary: pruebas unitarias para evaluar el desempeño, análisis de características basado en un conjunto de criterios de valoración y un caso de estudio donde se aplicó a desarrolladores un cuestionario para conocer la percepción de utilidad y facilidad de uso. Elementary es un aporte en el ámbito de las herramientas para construcción de aplicaciones web que se caracteriza por ser fácil de utilizar, bien documentado y diseñado para extenderlo con nuevos módulos o librerías, apropiado para desarrolladores con poca experticia, el cual establece un compromiso entre los principios de la Ingeniería del Software de generalidad, incrementalidad y la reutilización.

Palabras clave: framework web, PHP, HGMVC, documentación del framework.

ABSTRACT

The development complexity of current Web applications, has promoted the use of well-defined reuse structures known as programming Frameworks, many of them made in PHP; the use of these frameworks demand adequate programmer skills. Therefore, this article presents Elementary based on the proposed architectural pattern HGMVC (Hierarchical Generator-Model-View-Controller), aimed at inexperienced programmers

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

because it minimizes the complexity of some activities, such as configurations, installation and instantiation to carry out a new project, while providing appropriate documentation. Various types of assessment were performed to Elementary: unitary tests to evaluate the performance, analysis of characteristics based on a set of evaluation criteria and a case study in which one a questionnaire was applied to developers in order to determine the perception of usefulness and ease of use. Elementary is a contribution in the field of tools for building Web applications characterized by easy to use, well documented and designed for extending with new modules or libraries, suitable for developers with little expertise, which establishes a compromise between the principles of Software Engineering Generality, Incrementality and Reuse.

Keywords: framework web, PHP, HGMVC, framework documentation.

INTRODUCCIÓN

En la actualidad, muchos desarrolladores de software orientan sus aplicaciones como soluciones específicas a un problema, trayendo como consecuencia que mientras surgen nuevos requisitos, se debe realizar el re-trabajo correspondiente en las distintas etapas del desarrollo, contribuyendo con la duplicidad de código entre aplicaciones y el alto costo asociado al recurso de horas hombre necesarias para la construcción de las nuevas soluciones, sin aprovechar los esfuerzos de diseño realizados anteriormente.

En este sentido, la aparición de los Frameworks de desarrollo para aplicaciones web ha ayudado a mitigar esta situación, brindando al desarrollador una base para iniciar su trabajo, convirtiéndose en herramientas que facilitan la reutilización del código; sin embargo, surge una nueva problemática, la cual consiste en la utilización adecuada de las herramientas del Framework, especialmente en el caso de programadores no expertos.

Ahora bien, si un buen diseño e implementación de un Framework Web son requisitos necesarios para una reutilización exitosa, también una buena documentación es crucial (Aguiar y David, 2011). Por ello, la simplicidad de un Framework está asociada con la facilidad de instalarlo e instanciarlo, así como de proveer una documentación bien estructurada mediante la cual el desarrollador pueda aprender de forma sencilla a utilizarlo.

En esta investigación se aborda el desarrollo de un Framework en lenguaje PHP de nombre Elementary, siguiendo la metodología de Bosch y otros (1999) y lineamientos de Cwalina y Abrams (2006) con un diseño simple, incluyendo los módulos estrictamente necesarios, facilitando su instalación e instanciación, también se proporciona una documentación minimalista conforme a la propuesta de Aguiar y David (2000, 2011), dirigido a desarrolladores expertos e inexpertos.

En la evaluación de Elementary, se realizaron varios tipos de pruebas: pruebas unitarias para evaluar el desempeño, análisis de características basado en el modelo para comparar Frameworks Web Agiles de Fernández-Villamor y otros (2008) mediante un conjunto de criterios de evaluación y un caso de estudio de Yin (2002) en una población de programadores para determinar la aceptación del Framework.

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

Se ha estructurado el artículo en cinco secciones:

- Descripción del Problema: describe la evolución y situación actual de los Frameworks, adicionalmente resalta el valor, utilidad y aporte de Elementary.

- Fundamentos teóricos y tecnológicos: considerados para proponer a Elementary como una solución a la problemática planteada.

- Metodología: incluye una descripción del enfoque metodológico utilizado.

- Resultados: se indican los logros alcanzados en esta investigación.

- Conclusiones: se presentan Conclusiones y Recomendaciones para futuras mejoras del Framework y herramientas que facilitarían su uso.

DESCRIPCIÓN DEL PROBLEMA

La Ingeniería del Software asume la reutilización como un aspecto fundamental en la construcción sistemática de aplicaciones de software de calidad, porque brinda ventajas tales como la disminución de los costos, desarrollo rápido de aplicaciones, cumplimiento de estándares, entre otras (Sommerville, 2005). En este sentido, se favorece la reutilización con el uso de los Frameworks de Programación, conformados por una agrupación de Clases Concretas o Abstractas, diseñadas para trabajar juntas (Rogers, 1997).

Estos Frameworks han cambiado con el paso del tiempo, al principio eran principalmente verticales, diseñados para un tipo de aplicación muy particular, por ejemplo, una aplicación bancaria con funciones específicas para el cálculo de intereses. En contraste, los Frameworks actuales son más horizontales, porque se basan en funciones generales de la aplicación, como acceso a base de datos o manejo de plantillas para las vistas, entre otras. Ejemplo de estos Frameworks horizontales son: Structs (Apache Software Foundation) y Java Server Faces (Sun Microsystems) para Java, o Symfony (Symfony, 2012) y CakePHP (CakePHP) para PHP; donde el uso de estos, se ve limitado por su complejidad derivada del propósito de satisfacer todos los posibles escenarios, por lo que se necesita un conocimiento avanzado para hacer la configuración mínima requerida y obtener los beneficios de la reutilización y extensibilidad.

De modo que la simplicidad en los Frameworks horizontales es una característica muy valorada, para que el desarrollador dedique poco tiempo de aprendizaje al momento de instanciarlo o adaptarlo en la implementación de una solución (Cwalina y Abrams, 2006).

De esta manera, para contribuir con la solución de la problemática planteada se propuso Elementary, el cual es un Framework basado en PHP 5.4 (PHP, 2012) bien diseñado siguiendo la metodología de Bosch y otros (1999) los lineamientos establecidos por Cwalina y Abrams (2006) y el enfoque de documentación propuesto por Aguiar y David (2000, 2011), promoviendo su evolución, integridad y consistencia, para brindar apoyo al desarrollador al facilitar la concentración de su esfuerzo en los objetivos del

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

proyecto de software y no en los detalles de bajo nivel, tales como, la configuración y mantenimiento de su entorno de desarrollo y prueba.

FUNDAMENTOS TEÓRICOS Y TECNOLÓGICOS

Esta sección incluye aspectos importantes para el diseño del Framework Elementary como son: principios en Ingeniería del Software, Frameworks de programación y el patrón arquitectónico **HGMVC** propuesto durante el desarrollo de Elementary.

PRINCIPIOS EN INGENIERÍA DEL SOFTWARE

Los mismos constituyen razones fundamentales sobre las cuales se discurre en el diseño de software, determinando lo que se aprende de la experiencia y cómo se comprenden las relaciones causa - efecto (Ghezzi y otros, 2002). En el caso particular de Elementary, la consideración de los principios de generalidad, incrementalidad y reutilización sugirió aspectos de diseño del Framework como, la identificación de los componentes y sus relaciones. La generalidad, se puede definir como la capacidad para dar respuesta a diversos problemas similares, con una misma solución; en este caso, se aplica un enfoque que consiste en observar y examinar algunos Frameworks ampliamente utilizados, para identificar aquellos aspectos comunes y otros que resaltan por su funcionalidad particular, considerada de gran valor, en el diseño de Elementary. La incrementalidad, se realiza mediante un diseño de clases, herencia y ligadura simple y de fácil comprensión que permite identificar donde adicionar componentes y código, facilitando el mantenimiento y evolución de Elementary. Para finalizar, el principio de reutilización fue la base para establecer el conjunto de clases abstractas, concretas y sus interfaces que constituyen el Framework las cuales son instanciadas en el desarrollo de aplicaciones específicas.

FRAMEWORKS DE PROGRAMACIÓN

En cuanto a la definición utilizada en esta investigación, se tiene que es un esquema de reutilización del software conformado por componentes y relaciones entre estos, por ejemplo: la abstracción de clases, objetos o componentes que la conforman; además, provee diferentes componentes de conexión a base de datos, como controladores para conexión directa (MySQL, SQL Server, Oracle) o de manera general, mediante el estándar ODBC (Open DataBase Connectivity) (Zabala y Ochoa, 2008).

Existen varios tipos de Frameworks, usualmente los enfocados en el desarrollo Web, ofrecen una capa de controladores de acuerdo con el patrón MVC (Model View Controller) (Cui y otros, 2009) facilitando así la integración con otras herramientas para la implementación de las capas de negocio y presentación. En la

Tabla 1 se presenta una clasificación.

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

Tabla 1. Clasificación de Frameworks según su Facilidad para hacer cambios

Tipo de Framework	Descripción
Caja Blanca	Poseen características de la orientación a objetos como herencia o ligadura dinámica, de modo que se extienden por herencia de sus clases base o redefiniendo sus métodos. También definen interfaces para componentes que pueden integrarse mediante composición de objetos. Requieren de un profundo conocimiento y comprensión de las clases a extender, así mismo, la dependencia entre métodos por causa de la herencia en el caso de redefinir una operación puede necesitar redefinir otra y así sucesivamente.
Caja Negra o Dirigidos por Datos	Se estructuran utilizando composición de objetos y delegación en lugar de herencia, componiendo objetos existentes de diferentes formas para reflejar el comportamiento de la aplicación.
Caja Gris	Es una mezcla de los dos tipos anteriores, permiten la extensión mediante la herencia y la ligadura dinámica, además tienen la propiedad de ocultar información innecesaria a los desarrolladores.

Fuente: Johnson y Foote (1988) y Venkateswara Rao y otros (2011).

En particular, Elementary es un Framework Caja Gris, ya que combina las ventajas de Caja Blanca (flexibilidad, fácil modificación) y Caja Negra (facilidad de uso, abstracción de clases ocultando código innecesario) brindando al usuario las características adecuadas para un desarrollo ágil y mantenible.

PATRÓN ARQUITECTÓNICO HGMVC

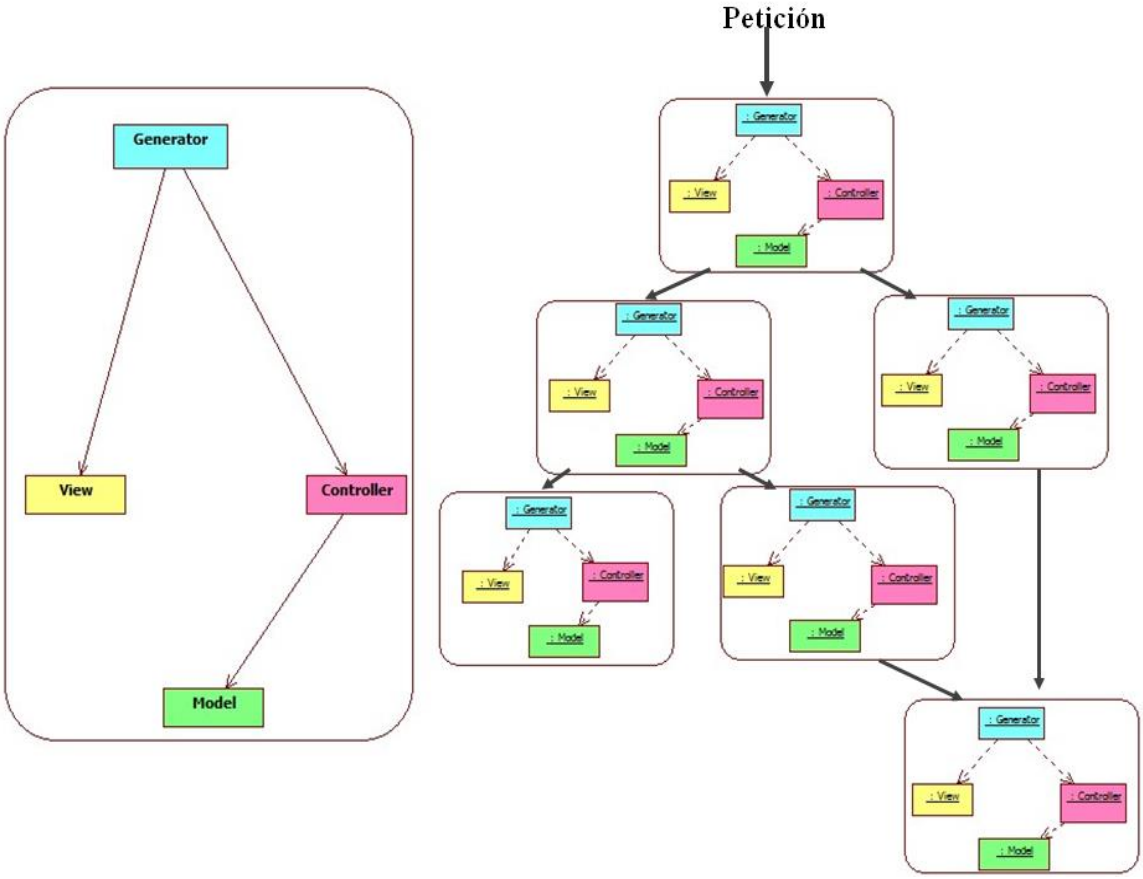
Entre tanto, como respuesta a la evolución de la complejidad de los sistemas, el patrón arquitectónico MVC (Model–View–Controller) altamente usado en los Frameworks de programación actuales por su versatilidad y fácil implementación, ha evolucionado a un nuevo diseño, HMVC (Hierarchical Model–View–Controller) (Rizvi y Hassan, 2009) y (Cogan, 2010) que al primero, permitiendo el uso recursivo de su estructura. Adicionalmente, como un aporte de esta investigación, se propone la incorporación de una cuarta capa a HMVC, denominada Generadora, encargada de recibir las peticiones del cliente e intermediaria entre las capas Vista y Controlador, es decir, la capa Controlador se dedica solo a la lógica del negocio. A esta propuesta, se le denomina **Patrón Arquitectónico HGMVC (Hierarchical Generator-Model–View–Controller)**.

En particular, la capa Generadora está encargada de manejar las peticiones del cliente, además de permitir la llamada a diversos controladores (dada la complejidad que

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

puede alcanzar en algunos casos), minimizando la funcionalidad que generalmente realiza la capa Controladora como se aprecia en la Figura 1.

Figura 1. Patrón Arquitectónico Hgmvc (Hierarchical Generator-Model–View–Controller)



Fuente: elaboración propia.

METODOLOGÍA DE INVESTIGACIÓN

La metodología utilizada es Investigación Acción (IA) (Baskerville, 1999) la cual propone 5 fases iterativas: Diagnóstico, Plan de Acción, Tomar Acción, Evaluación y Especificación de Conocimiento, las cuales se explican a continuación.

1. Diagnóstico: Identifica la evolución y situación actual de los Frameworks, su complejidad que puede limitar su uso. Una solución a esta problemática la constituye un Framework bien diseñado, usando los principios de la Ingeniería de Software que mantenga su simplicidad, y además que esté bien documentado.

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

2. Planificación de la acción: establece el desarrollo sistemático de Elementary a partir del estudio de la metodología para el desarrollo de Frameworks propuesta por Bosch y otros (1999).

3. Tomar la acción: en esta fase se aplica la metodología de Bosch y otros (1999). Asimismo, en la actividad 6 de esta metodología, referente a la Documentación, se realiza con el enfoque para documentación de Frameworks de Aguiar y David (2000, 2011).

4. Evaluación: se realizan las pruebas y validaciones de Elementary.

5. Especificación del conocimiento: se elaboran conclusiones y recomendaciones, dejando abierta la posibilidad de otro ciclo de IA.

Asimismo, la propuesta de Bosch aplicada en la fase 2 de IA, incorpora los Diagrama de Clases y de Secuencia de UML (Miles y Hamilton, 2006) y comprende las siguientes actividades:

1. Análisis del Dominio: se identifican los requisitos necesarios según el dominio del Framework. El resultado de esta actividad es un Modelo de Análisis de Dominio, con los requisitos, conceptos del dominio y sus relaciones.

2. Diseño Arquitectónico: se considera el Análisis del dominio como entrada para decidir cuál estilo de arquitectura se adecúa para el desarrollo. Esto se denomina el diseño de alto nivel en el cual se basa el Framework. La salida de esta actividad consta de un diagrama de Estados y Transiciones.

3. Diseño del Framework: Se refina el diseño de alto nivel del Framework obtenido en la actividad anterior y se diseñan las clases adicionales necesarias, los resultados son: el alcance de las funcionalidades del Framework, las consideraciones y lineamientos de diseño basados en la arquitectura usada, y el diagrama de clases del Framework.

4. Implementación del Framework: codificación.

5. Prueba: se instancia el Framework para determinar si provee las funcionalidades propuestas, además se evalúa las funcionalidades del mismo. Se realizan pruebas unitarias, análisis de características y un estudio de caso de Yin. En esta investigación la fase de Evaluación de IA se corresponde con esta actividad.

6. Documentación: en este paso, se utiliza el enfoque de Aguiar y David (2000, 2011) para indicar como se estructura la documentación del API y el Cookbook, que consiste en un manual de instalación, configuración y algunos ejemplos de las funcionalidades básicas del Framework.

En conformidad con la aplicación de la metodología de Bosch y otros (1999) se presenta a continuación los resultados obtenidos.

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

ANÁLISIS DE DOMINIO

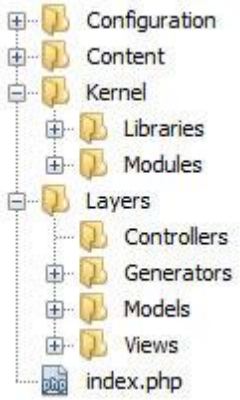
Como producto final de esta actividad, se identifican los componentes de Elementary y las relaciones entre ellos, obteniendo una estructura lógica inicial. Para lograr la modularidad del Framework y de esta manera favorecer su extensibilidad y reutilización, es necesaria la separación en componentes individuales débilmente acoplados. Es decir, un diseño de clases, herencia y ligadura simple y de fácil comprensión que permite identificar donde adicionar componentes y códigos. En consecuencia, Elementary está conformado por:

- Generator, el cual recibe la petición del cliente y la comunica al Controlador y la Vista;
- Controller, encargado de la lógica de negocio del sistema;
- Model, es el conjunto de clases las cuales son mapeadas por el ORM (De Troyer y otros, 2005) del Framework, el cual utiliza los componentes del Modelo para acceder a los datos de la Base de Datos y Sistema de gestión de Datos y de esta manera, permite al Controlador acceder a los mismos;
- View, muestra al usuario la información obtenida mediante las capas anteriores; y
- Kernel, es el núcleo del Framework, encargado de las funcionalidades básicas.

DISEÑO ARQUITECTÓNICO

El diseño de un Framework también necesita una estructura física, constituida por Carpetas y Archivos como se muestra en la Figura 2.

Figura 2. Estructura Física De Framework



Fuente: elaboración propia.

Seguidamente, se halla la explicación de los componentes del Diseño Arquitectónico:

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

- Configuration, carpeta que contiene los archivos de configuración necesarios para el funcionamiento de una instancia del Framework, por ejemplo: configuración de la Base de Datos, configuración inicial del Framework para la sesión, entre otros. En particular, Database.json.php contiene la información de la Base de datos (Gestor, Usuario, Contraseña, Schema), Web.json.php guarda las configuraciones de sesión, caché y configuración para el funcionamiento del sistema, Languages.json.php en el caso de poseer soporte multi-idiomas y el idioma por defecto se encuentra en este archivo.

- Content, contiene el contenido estático del sistema: Imágenes, Hojas de Estilo, JavaScript y Archivos de Idiomas, en las sub-carpetas Images, Style, Script y Languages, respectivamente.

- Kernel, es el núcleo del Framework, contiene las Interfaces y clases maestras. Está conformado por las sub-carpetas: Libraries, son las clases mínimas necesarias para el adecuado funcionamiento del Framework y Modules, todas las extensiones y componentes no fundamentales.

- Layers, las clases creadas por el desarrollador se encuentran en esta carpeta distribuidas en subcarpetas correspondientes a cada capa. La sub-carpeta Generators, guarda los archivos de la Capa generadora, el nombre del archivo generador es utilizado para navegar por el sistema y el nombre de la clase que contiene el archivo debe ser Nombre_del_Generador+Generator, por ejemplo http://dominio/Home accederá al generador cuyo nombre de Archivo es Home y el nombre de la clase será HomeGenerator; Controllers, incluye las clases de la capa Controlador del modelo **HGMVC**, el nombre del archivo será igual al generador y el nombre de la clase será Nombre_del_Controlador+Controller; Models, incluye las clases que son mapeadas a tablas de la base de datos; Views contiene una carpeta shared en la cual están todas aquellas vistas utilizadas por más de un generador y una carpeta por cada generador, en la cual se encuentra cada una de las vistas que usa dicho generador.

Los archivos de Configuración e Idiomas se proporcionan como archivos de configuración .json, ya que PHP posee un analizador sintáctico para este tipo de archivos, además de ser fáciles de leer para el programador.

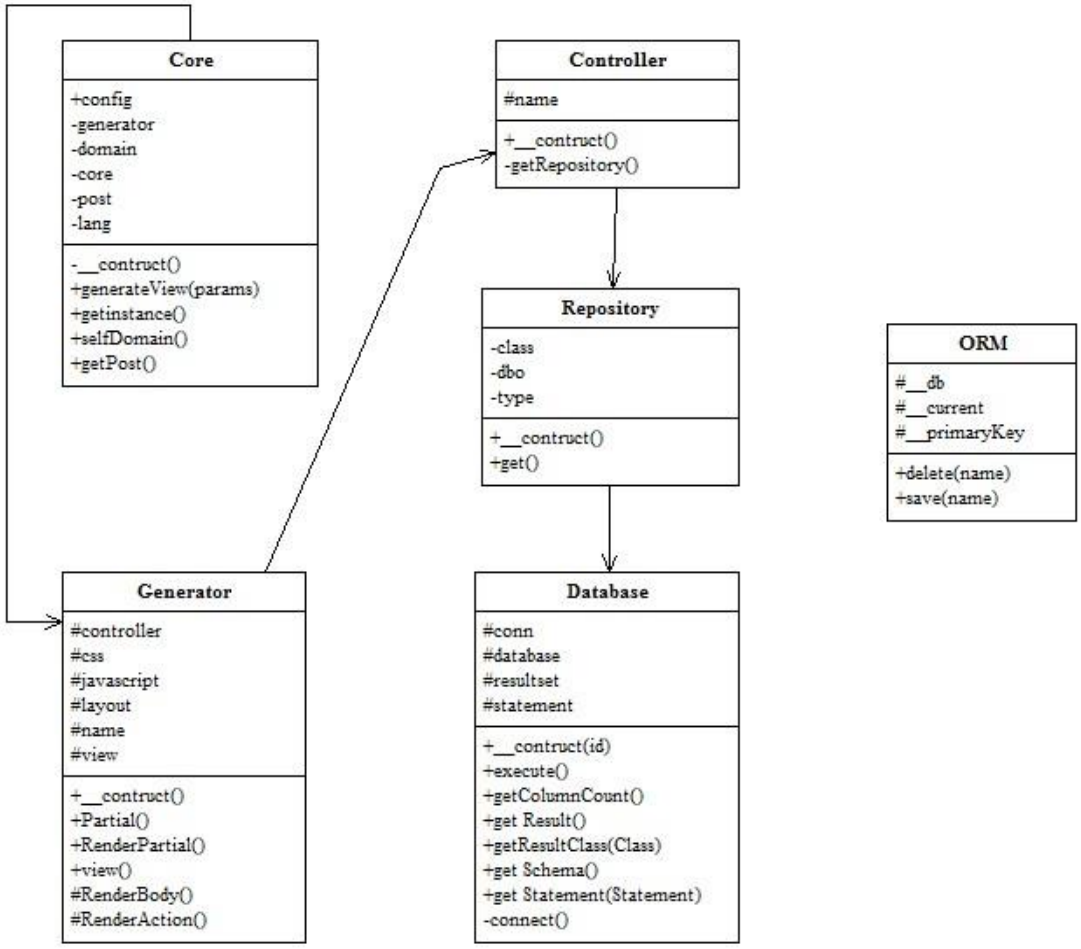
DISEÑO E IMPLEMENTACIÓN DEL FRAMEWORK

A partir de la estructura lógica y física de Elementary, presentadas en las subsecciones ANÁLISIS DE DOMINIO y DISEÑO ARQUITECTÓNICO, se proponen las clases necesarias y sus relaciones para el funcionamiento. Se dividen las funcionalidades en 6 clases: Core, Generator, Controller, Repository, ORM, y Database. La Figura 3 exhibe el Diagrama de Clases de Elementary, además, la descripción de los atributos y métodos se halla en la Fuente: elaboración propia.

Tabla 2.

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

Figura 3. Diagrama De Clases De Elementary



Fuente: elaboración propia.

Tabla 2. Descripción de Clases de Elementary

Clase	Atributos	Métodos
Core Recibe petición http del cliente, además maneja la configuración general del Framework en el archivo Web.ini. Se basa en el patrón de diseño Singleton y otros (1995).	config: arreglo de valores de configuración dados en archivo Web.ini, otras clases del Framework pueden leer la configuración. generator: contiene petición del cliente e instancia atributo al generador correspondiente. domain: dirección del dominio	__contruct: en correspondencia con patrón Singleton visibilidad del método privada. generateView: método llamado automáticamente al recibir la petición del cliente. Instancia al generador respectivo y ejecuta el método apropiado. getInstance: Retorna instancia

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

Clase	Atributos	Métodos
	del servidor, ejem. http://dominio.com. core: instancia para acceder a la clase, por ser una clase Singleton, se instancia una vez por ejecución. post: arreglo de valores enviados por POST, todas las clases del Framework pueden acceder estos valores de forma sencilla. lang: lenguaje actual de la aplicación	de la clase Core. getPost: accede los datos en el arreglo post. selfDomain: Retorna string (nombre del dominio actual).
Generator Todas las clases de la Capa generadora se extienden de una clase abstracta, contiene los métodos y atributos necesarios para interactuar con el controlador y las vistas.	controller: Instancia del controlador con el mismo nombre del generador css: Arreglo de archivos de estilos para la vista a generar. javaScript: Arreglo de archivos para la vista a generar. layout: plantilla a utilizar en la vista a generar. name: Nombre del generador actual (utilizado para generar automáticamente el controlador correspondiente) view: Archivo de la vista que se interpreta con datos recibidos desde el controlador	__contruct: Constructor de clase, automáticamente llamado por clase Core. Partial: Interpreta vista actual y retorna su contenido, esta vista no es agregada a la plantilla almacenada en atributo Layout. RenderPartial: Recibe nombre de vista, para representar una vista dentro de otra. View: Interpreta vista actual y retorna su contenido, agrega la vista generada a la plantilla RenderAction: Ejecuta el método Renderpartial de otro generador, así aumenta la reutilización de código.
Controller Al igual que Generator, todas las clases de la capa controladora se extienden de una clase abstracta, ésta contendrá métodos y atributos para interactuar con Repository.	name: nombre del controlador actual	__construct: Constructor de la Clase, llamado automáticamente por el generador. getRepository: Retorna instancia de un repositorio, dado el nombre de la tabla en base de datos.

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

Clase	Atributos	Métodos
Repository Compuesta por el ORM y Database, encargados de acceder a la base de datos y mapear las tablas a objetos del Framework; detecta si el Modelo que se intenta acceder existe, sino genera el archivo de la clase automáticamente utilizando las columnas de la tabla correspondiente. Para consultar la base de datos utiliza notación de LINQ, que facilita la creación de filtros de datos y mantiene la estructura libre de sintaxis SQL dentro del Framework.	class: almacena el nombre de la clase que será utilizada para mapear la tabla de la base de datos dbo: Almacena el objeto con el cual se extrae la data de la Base de Datos type: nombre de la tabla que se mapeara al objeto del Framework.	__contruct: Constructor de clase, recibirá el nombre de la tabla de base de datos la cual será mapeada get: Retorna el objeto mapeado sobre el cual se podrán hacer operaciones de búsqueda.
ORM Todas las clases de la capa Model extenderán de esta clase. Se encarga automáticamente de obtener los registros externos asociados a su modelo**.**	db: instancia de la clase Database, para acceder a los datos de la Base de Datos. _current: nombre de la tabla sobre la cual opera este objeto. _primaryKey: nombre de la columna clave primaria de tabla, para diferenciar entre realizar actualización de un objeto o registrar uno nuevo en base de datos.	delete: Elimina objeto actual y su registro de base de datos. save: Crea o actualiza el registro actual en la base de datos, utiliza la clave primaria de tabla para determinar si dicho objeto existe en base de datos y actualizarlo o crea un nuevo registro.
Database Se encarga de crear la conexión a base de datos y las consultas a la misma. Es una clase restringida para solo ser usada por la clase ORM automáticamente.	conn: conexión a la base de datos, persistente durante la ejecución del código. database: datos de conexión descritos en el archivo database.json.php resultSet: Resultado de la ejecución de un query, no debe accederse a este objeto directamente, usar alguno de	__contruct: Recibe el id o nombre de la conexión, estos datos se configuran en el archivo Database.json.php execute: Ejecuta el query almacenado en statement. getColumnCount: Retorna el número de columnas en el resultset.

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

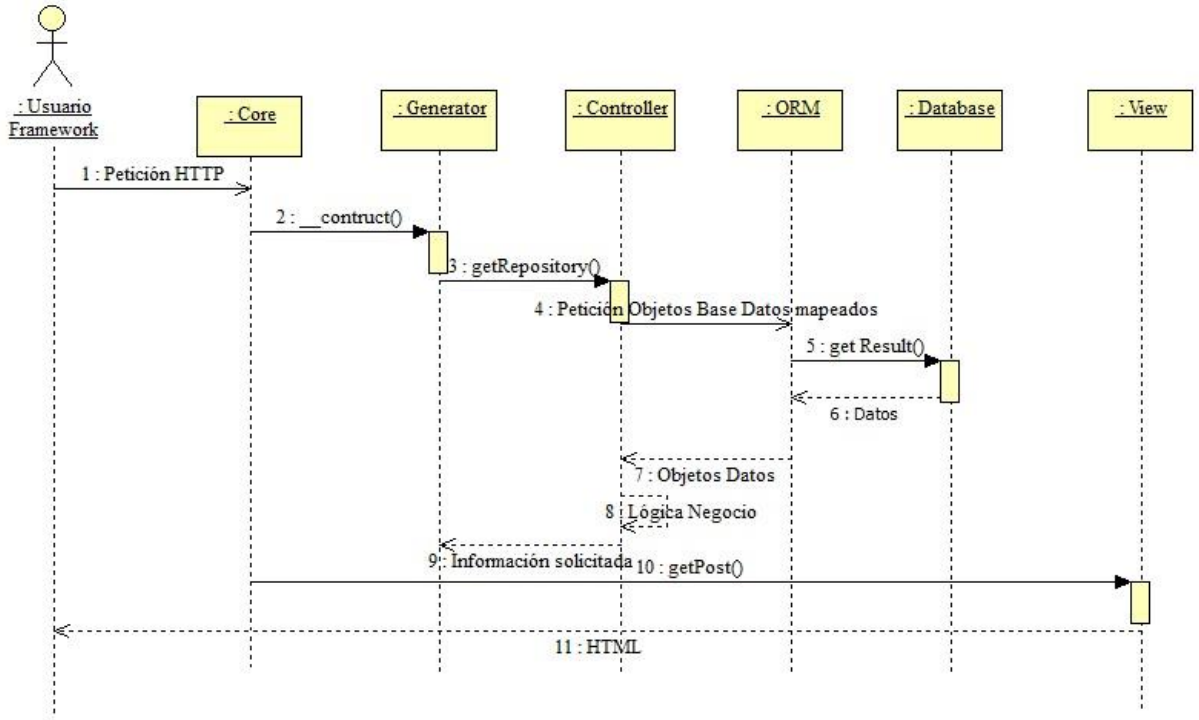
Clase	Atributos	Métodos
	los métodos getResult o getResultClass. statement: Query para ejecutar.	getResult: Retorna un objeto anónimo o arreglo de objetos anónimos con el resultado del query ejecutado. getResultClass: Recibe nombre de una clase y mapea las columnas del resultset con los atributos de la clase. getSchema: Regresa nombre del esquema o base de datos actual. setStatement: Almacena el query a ejecutar. bindParameters: Recibe nombre de una variable y su valor para reemplazarla en el statement. connect: Genera la conexión con base de datos.
View Contiene HTML's usados para presentar información al usuario, generados por Generator o por otra vista. A diferencia de otras capas es un archivo PHP plano, que puede contener etiquetas HTML y código PHP.		

Fuente: elaboración propia.

También, se presenta un diagrama de secuencia de UML para describir el comportamiento de Elementary, desde la petición del cliente hasta el retorno del HTML, ver Figura 4.

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

Figura 4. Diagrama de Secuencia para Funcionamiento de Elementary



Fuente: elaboración propia.

PRUEBA

Para el desarrollo y prueba de Elementary se emplearon dos ambientes o entornos, Windows y Linux (Alto Rendimiento) como se resume en la Tabla 3. Todos los ambientes utilizaron software para realizar las pruebas de rendimiento y velocidad: Apache 2, PHP 5.4.5 y MySQL 5.5

Tabla 3. Recursos de Ambientes para Desarrollo y Prueba

Entorno 1	**Entorno 2**
Windows 8 64bits	Debian 64bits
Pentium Dual CoreE5700 3.0Ghz	Intel® Xeon® X3430 2.6Ghz
4 GB Ram	8 GB Ram
1 TB de Disco 7200rpm	500 GB Disco 7200rpm

Fuente: elaboración propia.

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

Para Elementary se realizaron varios tipos de evaluaciones: pruebas unitarias para evaluar el desempeño, análisis de características basado en un conjunto de criterios de evaluación y un caso de estudio de Yin (2002). A continuación, se describen cada una de ellas.

PRUEBAS UNITARIAS

Las pruebas unitarias se realizaron en los 6 componentes: Core, Generator, Controller, Repository, ORM y Database, descritos en la subsección DISEÑO E IMPLEMENTACIÓN DEL FRAMEWORK. La Tabla 4 indica el tipo de prueba, los resultados obtenidos y las restricciones identificadas. Algunas de estas restricciones pueden ser consideradas como nuevos requisitos en el desarrollo de futuras versiones.

Tabla 4. Pruebas Unitarias

	Tipo de Prueba	Cantidad de pruebas promedio	Resultados obtenidos	Restricciones
Core	Lectura del archivo de configuración	30	La lectura de los parámetros arroja un Objeto conformado por cada elemento del archivo de configuración, ordenado de forma jerárquica	Esta configuración es solo lectura
	Creación del objeto Generador	30	Objeto generador adecuado a la petición hecha	La petición es sensible a mayúsculas y minúsculas
	Lectura y aplicación de la internacionalización	30	HTML internacionalizado	Los archivos de lenguaje pueden tener máximo 2 niveles de jerarquía
Generator	Lectura de Datos del controlador	30	El(Los) objeto(s) devueltos por el controlador son leídos correctamente	Solo puede ejecutar llamadas al controlador asociado
	Retorno de Vista	30	Retorna una vista incrustada en la plantilla	No es posible modificar datos de la plantilla dinámicamente
	Retorno de Vista Parcial	30	Retorna una vista, sin incrustar en la plantilla del sistema	

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

	Tipo de Prueba	Cantidad de pruebas promedio	Resultados obtenidos	Restricciones
Controller	Creación de Repositorio y lectura de Datos	30	Se generó el objeto con el mapeo de los campos de Base de Datos	Los atributos del objeto retornado son iguales a nombres en Base datos (distingue mayúsculas/minúsculas)
Repository	Mapeo de Base de datos a Modelo	30	Objeto con los datos solicitados	
ORM	Creación de Modelo y extensión de este componente	30	El objeto creado hereda las funcionalidades de la clase ORM	
	Identificación de clave primaria de tabla asociada al modelo	30	Almacena en atributo interno el nombre del campo clave primaria	Las tablas no pueden tener llaves primarias de más de una columna
Database	Creación de Objeto modelo	30	Genera un archivo PHP con una clase nombrada igual que su tabla conteniendo atributos nombrados como las columnas de dicha tabla	No genera los atributos de las relaciones de la tabla, estos son generados sobre demanda dinámicamente
	Lectura de Datos desde base de datos	30	Retorna un objeto o una lista de objetos del tipo del modelo actual	Retorna objetos simples iguales a descritos en Base de Datos

Fuente: elaboración propia.

La columna de Resultados obtenidos se explica por sí sola; en cuanto a las Restricciones, se resalta que el archivo de lenguaje solo permite 2 niveles de jerarquía, en futuras versiones se podría generar una clase que recorra recursivamente este archivo y así tener más niveles para mejorar su organización, la clase Repository requiere que el sistema gestor de datos sea en particular MySQL, y las tablas han de estar relacionadas entre sí utilizando claves foráneas, y las claves primarias de las tablas no pueden ser compuestas.

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

ANÁLISIS DE CARACTERÍSTICAS

Adicionalmente, como parte de la actividad de prueba, se aplica uno de los métodos de DESMET (Kitchenham, 1996), el Análisis de Características para comparar un grupo de Frameworks reconocidos entre desarrolladores de sistemas Web y Elementary.

En la aplicación de este método, se identifican las características que hacen a un Framework funcional y un conjunto de criterios para valorar cada una de estas características. Para esta evaluación se siguieron los siguientes pasos.

1. Se seleccionan algunos de los Frameworks más utilizados actualmente y se comparan: Symfony que destaca por ser uno de los más aceptados, CakePHP empleado para desarrollo rápido de aplicaciones, Codeigniter considerado de complejidad intermedia y Elementary, propuesto en esta investigación.

2. Se establece un conjunto de características a ser evaluadas mediante criterios, con sus respectivas métricas y escalas.

3. Se aplican los criterios de la propuesta para comparar Frameworks Web de Fernández-Villamor y otros (2008) la cual se basa en 7 características resumidas en la Tabla 5.

Tabla 5. Características y Criterios Propuestos

Característica	Criterios de evaluación
Descripción del Dominio y Persistencia	**D1**. ¿La migración de data se encuentra contenida en el proceso de desarrollo? **D2**. ¿Es el esquema de la base de datos automáticamente inferido de la definición del dominio? **D3**. ¿Es la definición del dominio automáticamente generado del esquema de la Base de Datos? **D4**. ¿Soporta Validaciones? **D5**. ¿Soporta Transacciones?
Presentación	**P1**. Contiene un generador de código para el manejo de operaciones CRUD. **P2**. ¿La capa de presentación puede ser definida usando alguna otra tecnología? **P3**. ¿La capa de presentación está atada a una tecnología específica? **P4**. ¿Soporta multilenguaje?

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

Seguridad	**S1**. Escapa automáticamente los parámetros enviados desde el cliente o provee una herramienta para hacerlo para evitar inyección de código? **S2**. Soporta autenticación de usuario?
Usabilidad	**U1**. Soporta generación automática de Código? **U2**. Soporta tipos de datos dinámicos? **U3**. Soporta varios ambientes de implementación? (Desarrollo, Prueba, producción)
Pruebas	**T1**. Soporta unidades de prueba? **T2**. Soporta pruebas funcionales? **T3**. Soporta pruebas de integración? **T4**. Soporta pruebas de rendimiento? **T5**. Soporta pre llenado de base de datos?
Orientación a Servicios	**OS1**. Soporta arquitectura REST? **OS2**. Soporta servicios SOAP?
Orientación a Componentes:	**O1**. Permite diferentes módulos de persistencia sin refactorización? **O2**. Soporta diferentes tecnologías en la capa de presentación sin refactorización? **O3**. Soporta la conexión con sistemas realizados con otros Frameworks?

Fuente: Fernández-Villamor, Díaz-Casillas & Iglesias (2008).

Para la evaluación se emplea la siguiente escala: **0** si el Framework no posee la característica, **1** si podría poseerla utilizando un software de tercero o plugin y **2** si la posee. La

Tabla 6 muestra la valoración de criterios y la ponderación de características.

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

Tabla 6. Análisis De Características

Criterio	Symfony	Codeigniter	CakePHP	Elementary
D1	2	1	0	2
D2	2	1	0	0
D3	2	1	2	2
D4	2	2	2	2
D5	2	2	2	2
Total Descripción del Dominio y Persistencia	10/10 (100%)	7/10 (70%)	6/10 (60%)	8/10 (80%)
P1	2	1	2	2
P2	2	2	0	2
P3	2	0	2	2
P4	2	2	2	2
Total Presentación	8/8 (100%)	5/8 (62.5%)	6/8 (75%)	8/8 (100%)
S1	2	1	2	2
S2	2	2	2	2
Total Seguridad	4/4 (100%)	3/4 (75%)	4/4 (100%)	4/4 (100%)
U1	2	2	2	2
U2	2	1	0	2
U3	2	1	0	1
Total Usabilidad	6/6 (100%)	4/6 (66.6%)	2/6 (33.3%)	5/6 (83.3%)
T1	2	1	2	0

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

Criterio	Symfony	Codeigniter	CakePHP	Elementary
T2	2	0	2	0
T3	2	2	1	0
T4	2	1	1	2
T5	2	0	2	0
Total Pruebas	10/10 (100%)	4/10 (40%)	8/10 (80%)	2/10 (20%)
OS1	1	2	1	2
OS2	1	1	1	1
Total Orientación a Servicios	2/4 (50%)	3/4 (75%)	2/4 (50%)	3/4 (75%)
O1	2	1	2	2
O2	2	1	0	2
O3	1	0	1	0
Total Orientación a Componentes:	5/6 (83.3%)	2/6 (33.3%)	3/6 (50%)	4/6 (66.6%)

Fuente: elaboración propia.

Se evidencia que Elementary iguala o supera a la mayoría de los Frameworks actuales seleccionados en las características estudiadas (Dominio, Presentación, Seguridad, Usabilidad, Servicios y Componentes). Sin embargo, en la característica de pruebas es superado, porque no permite la aplicación de pruebas unitarias sobre las clases generadas.

Finalmente, se aplica el Estudio de Caso para determinar la aceptación de Elementary.

ESTUDIO DE CASO

Se empleó el método de Yin (2002), que consta de 5 (cinco) componentes:

- Pregunta de la investigación: "¿Elementary es un Framework simple, bien documentado, evolutivo, integrado y consistente?"

- Supuestos: ¿Al utilizar Elementary aumenta la eficiencia del desarrollo en los proyectos de desarrollo web en PHP?

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

- Unidad de análisis: El individuo se convierte en el objeto de análisis del caso de estudio, ya que se desea conocer su experiencia al utilizar Elementary y cómo afecta su proceso de desarrollo de software.

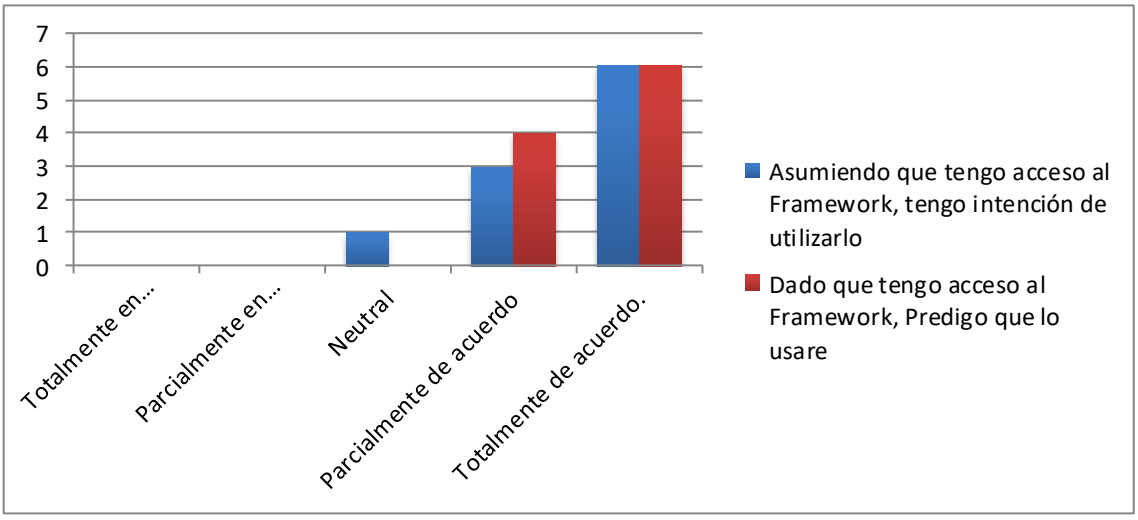
- Relación entre datos y los supuestos, postulados o proposiciones: para comprobar los supuestos se proporcionó Elementary a varios programadores para que desarrollaran una pequeña aplicación CRUD (Create, Read, Update and Delete) (Heller, 2007) utilizando una base de datos dada y luego se entregó un cuestionario basado en TAM (Technology Acceptance Model) (Venkatesh y Davis, 2000) para medir la percepción de utilidad, facilidad de uso y utilidad, del software, es decir, predecir la aceptación y uso en el trabajo de esta herramienta tecnológica (Venkatesh y otros, 2003). El Criterio para interpretar los hallazgos es la utilidad del Framework para los programadores durante el proceso de desarrollo.

La muestra utilizada consistió en 10 programadores con diferentes niveles de experticia a quienes se les solicitó que probaran desarrollar una pequeña aplicación. De estas 10 pruebas, 5 se realizaron en simultáneo durante un encuentro de tecnología en la ciudad de Santiago de Chile, las otras 5 se realizaron por separado con programadores venezolanos residenciados en España y Venezuela.

Los resultados del Estudio de Caso se resumen a continuación:

La Figura 5 muestra que un 60% de los encuestados tiene una alta intención de utilizar Elementary, mientras que el 40% se divide entre interesado o neutral.

Figura 5. Intención de Uso de Elementary

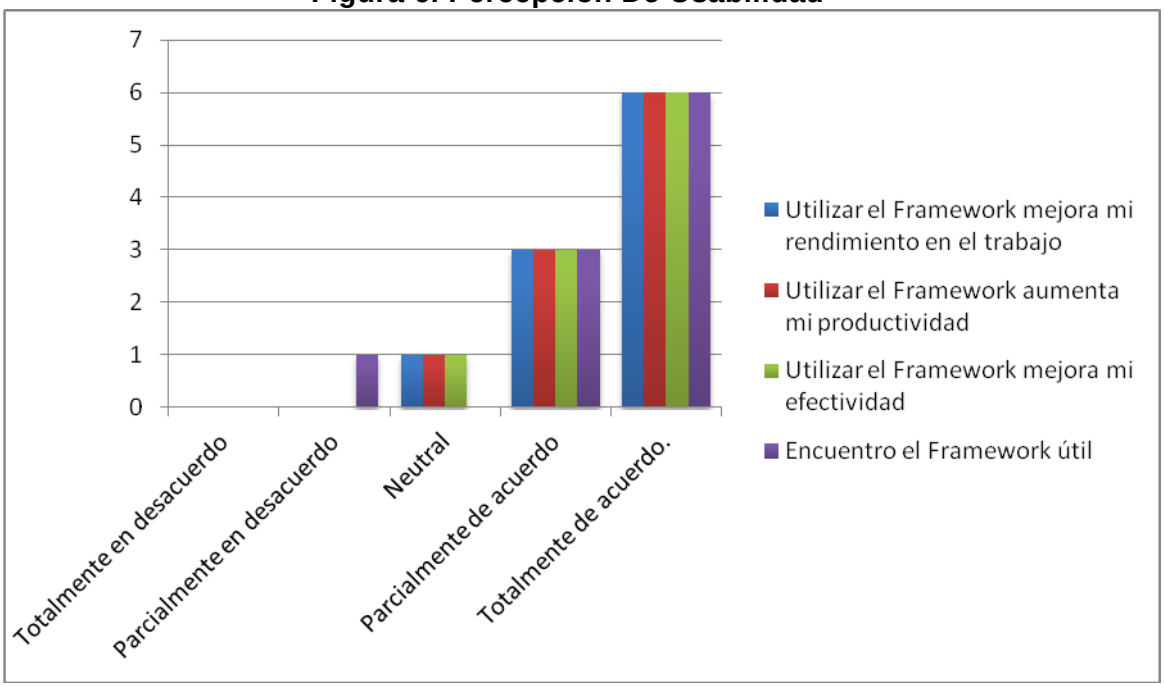


Fuente: elaboración propia.

En la Figura 6, se observa que un 60% de los encuestados respondieron que Elementary lo consideran útil durante el proceso de desarrollo de sus aplicaciones.

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

Figura 6. Percepción De Usabilidad

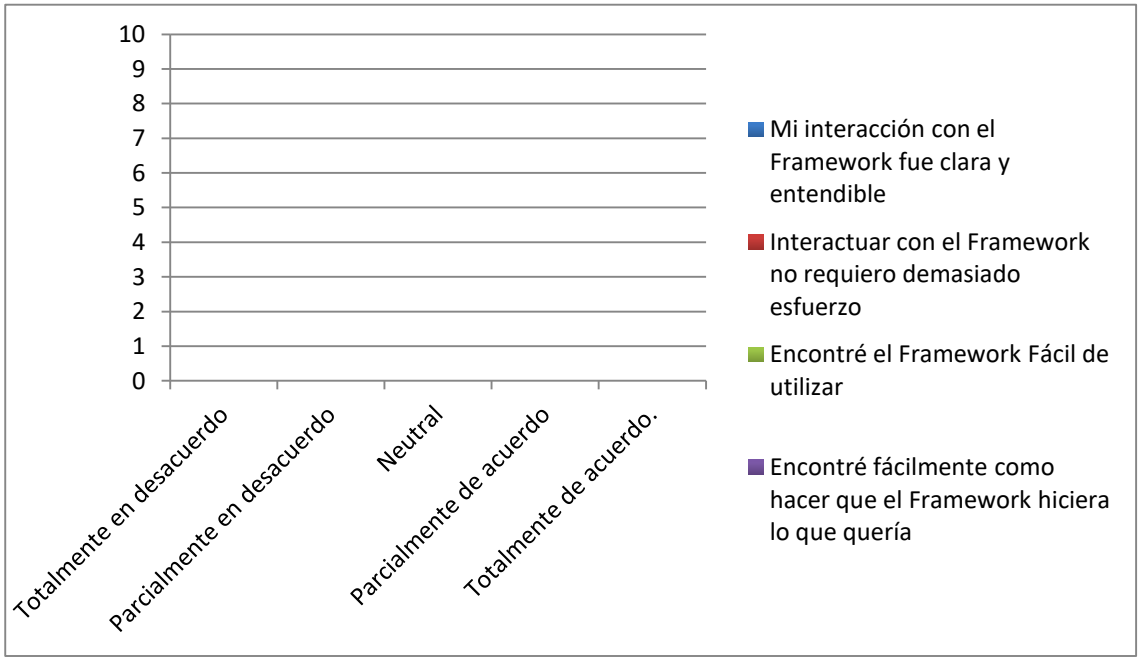


Fuente: elaboración propia.

La Figura 7 indica que entre el 80%-90% de los encuestados respondieron que Elementary fue fácil de usar.

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

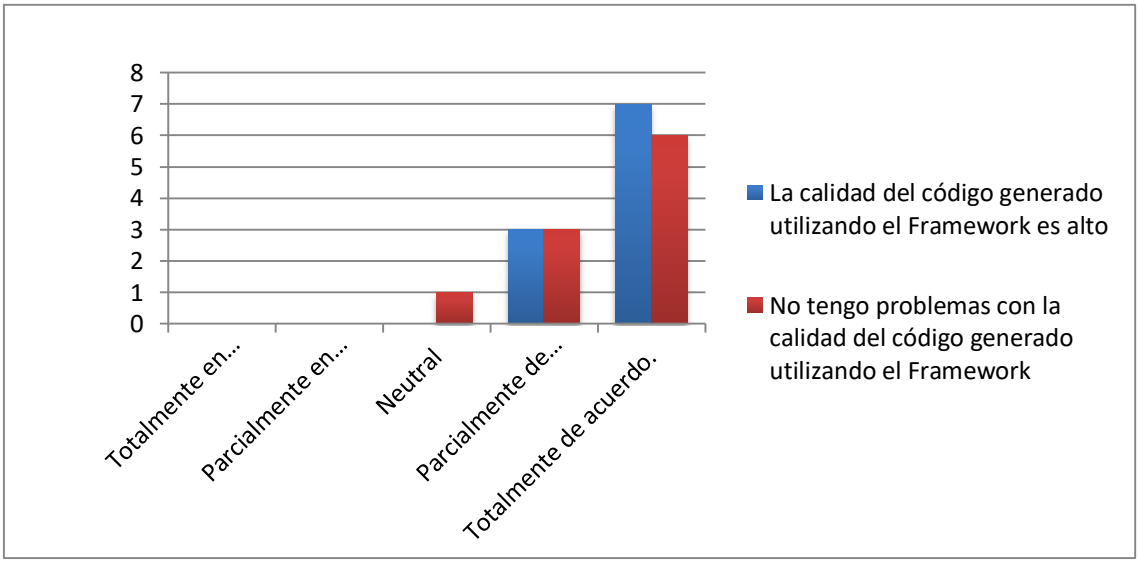
Figura 7. Percepción de Facilidad de Uso



Fuente: elaboración propia.

Se puede observar en la Figura 8 que 60-70% de los encuestados están satisfechos con la Legibilidad y comprensión del código que es generado al utilizar Elementary.

Figura 8. Legibilidad y Comprensión del Código Generado

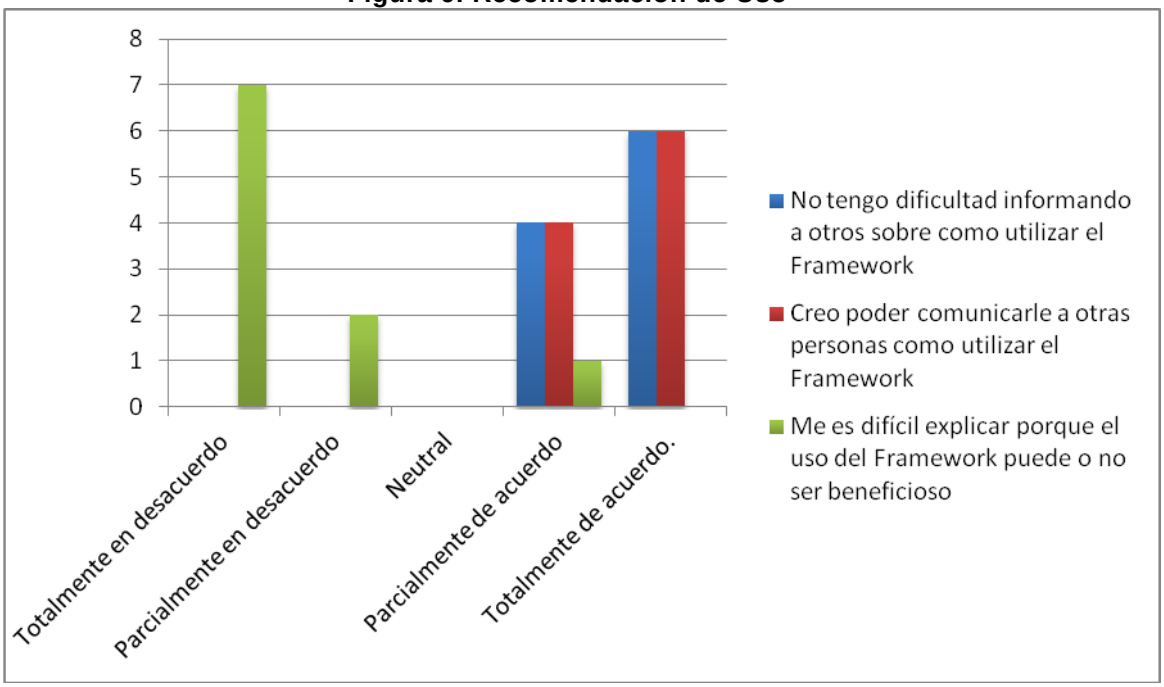


Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

Fuente: elaboración propia.

La Figura 9 ilustra como más del 60% de los encuestados están capacitados no solo para explicar porque el uso de Elementary puede ser beneficioso en un proyecto, sino también de explicar a otros usuarios como utilizarlo.

Figura 9. Recomendación de Uso



Fuente: elaboración propia.

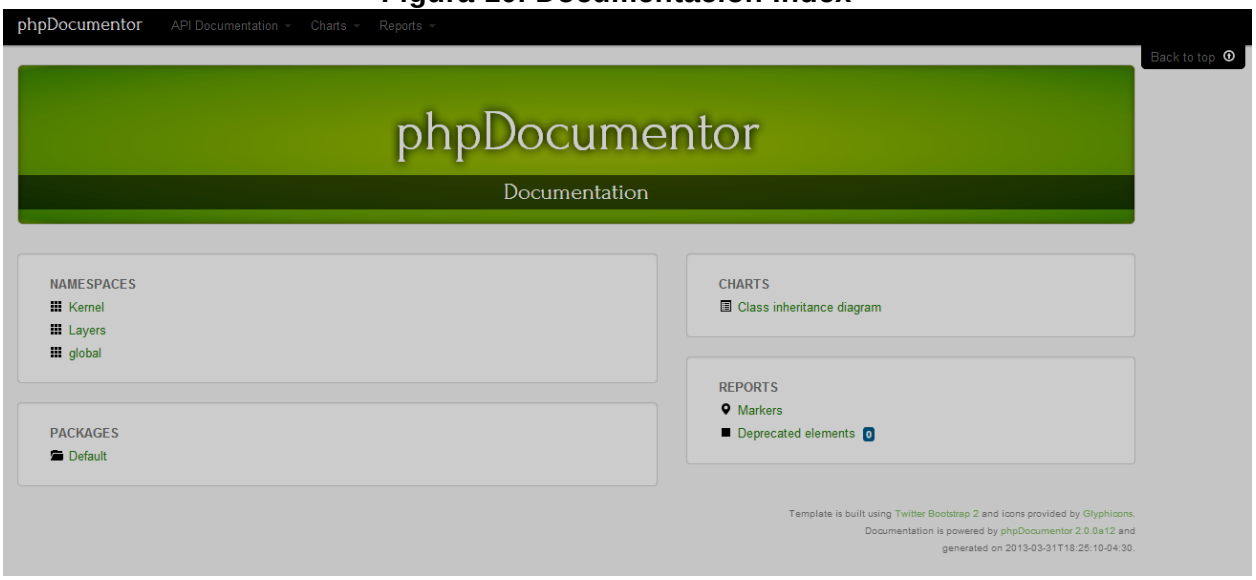
DOCUMENTACIÓN

En comparación con documentar una aplicación Orientada a Objetos o una Librería, la documentación de un Framework se considera una actividad difícil, siendo su generalidad una de las principales razones (Aguiar y David, 2000, 2011); Elementary se documentó siguiendo los lineamientos expuestos por los autores citados, esta documentación comprende el API de Elementary, también un manual de recetas (en inglés, Cookbook) donde se muestran ejemplos de cómo utilizar los distintos módulos y componentes. La Figura 10 muestra la organización de la documentación, generalidades del Framework, patrón de diseño, clases y ejemplos, en particular en la Fuente: elaboración propia.

Figura 11 se observa la descripción de las clases fundamentales.

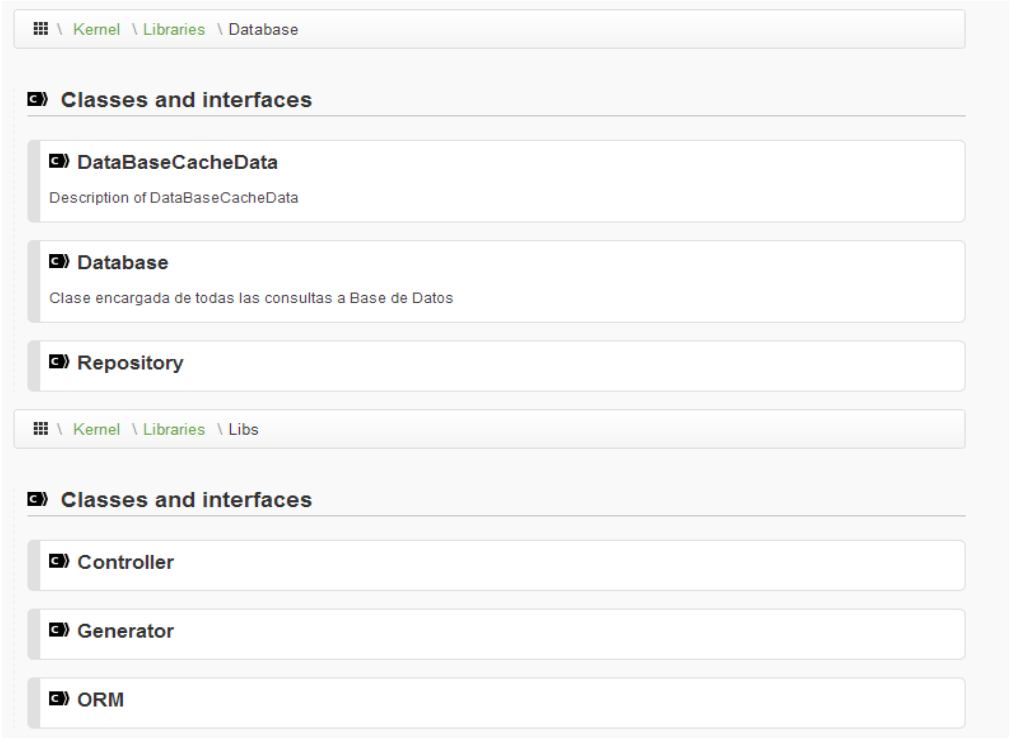
Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

Figura 10. Documentación Index



Fuente: elaboración propia.

Figura 11. Documentación Clases e Interfaces



Fuente: elaboración propia.

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

CONCLUSIONES

El Framework en PHP denominado “Elementary” es un aporte en el ámbito de las herramientas para la construcción de aplicaciones Web que se caracteriza por ser fácil de utilizar, bien documentado y diseñado para permitir expansiones con nuevos módulos o librerías, orientado a desarrolladores novatos, el cual establece un compromiso entre los principios de la Ingeniería del Software de Generalidad, Incrementalidad y la Reutilización.

Asimismo, la construcción de Elementary se considera una experiencia documentada de desarrollo sistemático de software, basada en la metodología de Bosch para desarrollo de Frameworks, el uso del patrón arquitectónico HGMVC propuesto y el enfoque para documentación de Aguiar y David, la cual puede ser seguida por otros investigadores del área en el desarrollo de nuevas propuestas de reutilización.

Adicionalmente, la validación realizada con un Estudio de Caso de Yin donde se aplicó un cuestionario, evidencia que Elementary tuvo un alto grado de aceptación e intención para ser utilizado en el desarrollo de aplicaciones web; de esta manera, se validó su facilidad de uso y aprendizaje para programadores novatos y expertos. Igualmente, la legibilidad y comprensión del código, así como la posibilidad de recomendar Elementary fueron valoradas favorablemente. Por otra parte, las pruebas unitarias permitieron establecer algunas restricciones, a saber: el archivo de lenguaje solo permite 2 niveles de jerarquía, en futuros trabajos se podría generar una clase que recorra recursivamente este archivo y así tener más niveles para mejorar su organización, la clase Repository requiere que la base de datos sea MySQL y que las tablas estén relacionadas entre sí utilizando claves foráneas, y las claves primarias de las tablas no pueden ser compuestas.

REFERENCIAS BIBLIOGRÁFICAS

Aguiar, A. y David, G. (2000). A Minimalist Approach to Framework Documentation. Proceedings. (Pp. 1-11).

Aguiar, A. y David, G. (2011). Patterns for Effectively Documenting Frameworks. Lecture Notes in Computer Science. Volume 6510, (Pp. 79-124).

Baskerville, R. (1999). Investigating Information Systems with Action Research. Communications of the Association for Information Systems. Volumen 2, (Pp. 1-32).

Bosch, J. Molin, P. Mattsson, M. Bengtsson, P. y Fayad, M. (1999). Object-oriented foundations of framework design. Documento en línea. Disponible en: http://www.wiley.com/legacy/compbooks/frameworks/preface.htm. Consulta: 06/11/2014.

Cogan, B. (2010). Nettuts. Documento en línea. Disponible en: http://net.tutsplus.com/tutorials/php/hvmc-an-introduction-and-application/. Consulta: 31/01/2013.

Cui, W., Huang, L., Liang, L., y Li, J. (2009). The Research of PHP Development Framework Based on MVC Pattern. Proceeeding. (Pp. 947-949).

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

Cwalina, K. y Abrams, B. (2006). Framework Design Guidelines. United States. Microsoft Windows Development Series.

De Troyer, O. Casteleyn, S. y Plessers, P. (2005). Using ORM to model Web Systems. Lecture Notes in Computer Science. Volume 3762, (Pp. 700-709).

Fernández-Villamor, J. Díaz-Casillas, L. y Iglesias, C. (2008). A comparison model for agile web frameworks. Proceedings. Número 14.

Ghezzi, C. Jazayeri, M. y Mandrioli, D. (2002). Fundamentals of Software Engineering. United States. Prentice Hall.

Heller, M. (2007). REST and CRUD: the Impedance Mismatch. Documento en línea. Disponible en: http://www.infoworld.com/d/developer-world/rest-and-crud-impedance-mismatch-927. Consulta: 25/07/2012.

Johnson, R. E. & Foote, B. (1988). Designing reusable classes. Journal of Object-Oriented Programming. Volumen 1, número 2, (Pp. 22–35).

Kitchenham, B. (1996). DESMET: A method for evaluating Software Engineering methods and tools. https://ai2-s2-pdfs.s3.amazonaws.com/1f41/32d4e048508e1f587d8ada8b9a08b5a180be.pdf. Consulta: 01/06/2010.

Miles, R. y Hamilton, K. (2006). Learning UML 2.0. United States. O'Reilly.

PHP. (2012). Php.net. PHP Nuevas Características. http://php.net/manual/es/migration54.new-features.php. Consulta: 25/07/2012.

Rizvi, S. y Hassan, S. (2009). Achieving Loose Coupling Between Different Components of Model-View-Controller For Web Based Application. Proceedings. (Pp. 1-4).

Rogers, G. (1997). Framework-Based Software Development in C++. United States. Prentice Hall.

Sommerville, I. (2005). Ingeniería del software. España. Addison Wesley.

Symfony. (2012). The Book for Symfony 2.0. United States. Symfony.

Venkatesh, V. y Davis, F. (2000). A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies. Management Science. Volumen 46, número 2, (Pp. 186-204).

Venkatesh, V. Morris, M. Davis, G. y Davis, F. (2003). User Acceptance of Information Technology: Toward a Unified View. MIS Quarterly. Volumen 27, número 3, (Pp. 425-478).

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016

Venkateswara Rao, K. Govardhan, A. y Chalapati Rao, K. (2011). An Overview of Object-Oriented Frameworks with Application in Spatiotemporal Data Mining. CVR Journal of Science & Technology. Número 1.

Yin, R. (2002). Case Study Research, Design and Methods. United States. Newbury Park: Sage Publications.

Zabala, X. y Ochoa, C. (2010). Estudio de Frameworks para PHP e integración a una herramienta IDE. Tesis de Ingeniería de Sistemas. Escuela Superior Politécnica de Chimborazo. Ecuador.

Depósito Legal: PPX200002ZU2142 / ISSN: 1856-4194. Volumen 15 Edición No 2 Julio – Diciembre 2016