

ESTUDIO DE FRAMEWORKS PARA EL DESARROLLO DE APLICACIONES
WEB BAJO ENTORNOS DE DESARROLLO DE CÓDIGO LIBRE, APLICADO A
UN PORTAL DE COMERCIO ELECTRÓNICO

JUAN CAMILO PRADA OJEDA

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA
FACULTAD DE INGENIERÍA DE SISTEMAS
SISTEMAS DE INFORMACIÓN E INGENIERÍA DE SOFTWARE
BUCARAMANGA
2008

ESTUDIO DE FRAMEWORKS PARA EL DESARROLLO DE APLICACIONES
WEB BAJO ENTORNOS DE DESARROLLO DE CÓDIGO LIBRE, APLICADO A
UN PORTAL DE COMERCIO ELECTRÓNICO

JUAN CAMILO PRADA OJEDA

Trabajo de grado presentado para optar al título de: Ingeniero de Sistemas

MCC. JAIME RANGEL CABALLERO

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA
FACULTAD DE INGENIERÍA DE SISTEMAS
SISTEMAS DE INFORMACIÓN E INGENIERÍA DE SOFTWARE
BUCARAMANGA

2008

Nota de aceptación:

Firma del presidente del jurado

Firma del jurado

Firma del jurado

Bucaramanga, 17 de junio de 2008

TABLA DE CONTENIDO

	Pág.
INTRODUCCIÓN	8
1. PRÁCTICAS DE DESARROLLO	11
1.1 TENDENCIAS DE DESARROLLO EN LA WEB	11
1.2 LOS FRAMEWORKS DE PROGRAMACIÓN	15
1.2.1 Utilidad de un framework.	16
1.3 PATRONES DE DISEÑO EN LOS FRAMEWORKS	17
1.4 PATRÓN MODELO – VISTA – CONTROLADOR.....	20
2. DISEÑO DE LA METODOLOGÍA DE EVALUACIÓN DE FRAMEWORKS	24

2.1	MODELO SQ-MET.....	24
2.1.1	Facetas de una aplicación Web.	25
2.1.2	Método de calificación en SQ-MET.	26
2.2	ADAPTACIÓN DEL MODELO SQ-MET AL MODELO DE EVALUACION DE FRAMEWORKS.....	27
2.3	DEFINICIÓN DEL MODELO DE EVALUACIÓN DE FRAMEWORKS	28
2.3.1	Definición de los Módulos	29
2.3.2	Definición del peso de un módulo	31
2.3.3	Definición de la puntuación de un módulo.	35
2.3.4	El punto de vista del programador.....	37
3.	SISTEMAS DE COMERCIO ELECTRÓNICO	41

3.1 ARQUITECTURA DE UN SISTEMA DE COMERCIO ELECTRÓNICO	42
4. APLICACIÓN DEL MODELO DE EVALUACIÓN DE FRAMEWORKS.....	47
4.1 FACETAS DEL PROTOTIPO	47
4.1.1 Especificación de facetas para un sistema de comercio electrónico.....	48
4.2 MÓDULOS DEL FRAMEWORK	51
4.2.1 Patrón de diseño Modelo Vista Controlador (MVC).	51
4.2.2 Manejo de usuarios (UM).....	52
4.2.3 Manejo de Bases de datos (DBM).	53
4.2.4 Manipulación de información (IM).	53
4.2.5 Manipulación de Documentos (DM).	55

4.2.6 Internacionalización (i18n).	55
4.2.7 Integración con Web Services (WSI).	56
4.3 PERSPECTIVA DEL PROGRAMADOR	56
4.4 EVALUACIÓN	57
4.4.1 Pesos de los módulos.	59
5. DESARROLLO DEL PROTOTIPO	65
5.1 EL ARCHIVO BOOTSTRAP	67
5.2 IMPLEMENTACIÓN DEL MVC	70
5.2.1 El controlador y la clase Zend_Controller_Action.	70
5.2.2 El modelo y la clase Zend_Db_Table.	71

5.2.3 La vista y la clase Zend_View.	73
5.3 MANIPULACIÓN DE USUARIOS Y LA CLASE ZEND_AUTH.....	75
5.4 FORMULARIOS Y LA CLASE ZEND_FORM	78
5.5 BUSCADOR Y LA CLASE ZEND_SEARCH_LUCENE	81
5.6 WEB SERVICES Y LA CLASE ZEND_SERVICE_AUDIOSCROBBLER	84
5.7 LISTAS DE CORREOS Y LA CLASE ZEND_MAIL.....	87
5.8 MANEJO DE MONEDAS Y LA CLASE ZEND_CURRENCY	89
6. CONCLUSIONES	91
7. RECOMENDACIONES Y TRABAJOS FUTUROS.....	92
BIBLIOGRAFÍA.....	93

LISTA DE TABLAS

Pág.

Tabla 1. Valores para la escala de importancia de un módulo	32
Tabla 2. Comparación de los módulos	33
Tabla 3. Comparación de los módulos con más de un participante	34
Tabla 4. Calculo del peso de los módulos	34
Tabla 5. Tabla de puntuación de un Sub-módulo	37
Tabla 6. Tabla de puntuación de las Generalidades	39
Tabla 7. Calculo de pesos de los módulos	59
Tabla 8. Calculo de pesos para el sub-módulo MVC	60
Tabla 9. Calculo de pesos para el sub-módulo UM	60
Tabla 10. Calculo de pesos para el sub-módulo DBM	60
Tabla 11. Calculo de pesos para el sub-módulo DM	61
Tabla 12. Calculo de pesos para el sub-módulo i18n	61

Tabla 13. Calculo de pesos para el sub-módulo IM	61
Tabla 14. Calculo de pesos para el sub-módulo WSI	62
Tabla 15. Calculo de pesos para las generalidades	62
Tabla 16. Calificación de frameworks.....	64
Tabla 17. Enrutamiento de las acciones en la clase AdminController	71

LISTA DE FIGURAS

	Pág.
Figura 1. Meme del concepto de Web 2.0	12
Figura 2. Esquema de la arquitectura MVC	21
Figura 3. Ejemplo de interacción del patrón MVC	22
Figura 4. Diagrama de secuencia del Modelo de evaluación de Frameworks	40
Figura 5. Estructura de directorios propuesta para el Zend Framework	66
Figura 6. Ejemplo de organización de las vistas	74
Figura 7. Formulario de ingreso al sistema	75
Figura 8. Fallo en la autenticación	77
Figura 9. Ejemplo de formulario generado con la clase Zend_Form.....	80
Figura 10. Resultado de una búsqueda en el prototipo de sistema e-commerce ..	84
Figura 11. Integración de Audioscrobbler en el prototipo	87

RESUMEN

Durante el periodo de elaboración de una aplicación web, el desarrollador se ve comprometido desde el principio a entregar un producto funcional en un tiempo reducido, manteniendo la calidad del software.

Para permitir a los programadores satisfacer sus necesidades a la hora de desarrollar, los frameworks de programación web libres se proponen como una solución. La nueva dificultad radica en la selección de dicho framework, pues cada uno ataca el problema de una manera diferente, haciendo que, lo que en un principio pudo ser una tarea trivial, ahora y debido a la gran cantidad de frameworks existentes, se convierta en un arduo proceso de selección a partir de la práctica con el framework.

Es por eso que este proyecto propone una metodología para la evaluación de frameworks y realiza una comparación entre cuatro de los mas representantes frameworks de programación de código libre basados en el lenguaje PHP, permitiendo realizar una selección basada en el estudio y finalmente soportando dicha selección mediante una aplicación realizada con el framework..

Palabras claves: Frameworks, PHP, Patrón de diseño, Facetas, Módulos.

Línea de investigación: Sistemas de información e ingeniería de software

INTRODUCCIÓN

La necesidad de manipulación de información en empresas y organizaciones, demanda una respuesta por parte de los desarrolladores de software a la hora de construir aplicaciones. Limitaciones de tiempo y dinero, reducen las posibilidades de un estudio elaborado y lento a la hora de desarrollar software que satisfaga las necesidades de un entorno tan dinámico como el entorno web actual¹.

En respuesta a esas necesidades, las tendencias de desarrollo se han dirigido hacia nuevos paradigmas metodológicos. Las antiguas y lentas tendencias de desarrollo han visto la necesidad de evolucionar a nuevos y ágiles procesos que permitan el despliegue de plataformas informáticas, en menor tiempo y a un costo más bajo.

En el campo del desarrollo web, las tendencias sociales actuales basadas en una comunidad activa de usuarios en la red, determinan la prioridad en el desarrollo de aplicaciones; en la medida en que más usuarios interactúan con el sistema y demandan más información, los nuevos procesos de desarrollo deben permitir la elaboración de aplicaciones que satisfagan las demandas actuales por parte de usuarios en un tiempo reducido².

1 FILEV, Andrew. Adoptar y aprovechar los procesos ágiles en el desarrollo de software internacional [en línea] MSDN, 26 de Enero de 2007 [Citado el 22 de Octubre de 2007] Disponible en Internet: <<http://www.microsoft.com/spanish/msdn/articulos/archivo/260107/voices/bb245671.aspx#EVB>>

2 LADD, Seth y Donald. Expert Spring MVC and the Web Flows, Apress. Estados Unidos. 2006. 423 p.

Sin embargo, para que estos procesos ágiles sean efectivos, deben desarrollarse con base en la colaboración y la flexibilidad de los equipos de trabajo. Es por esto que, teóricamente, no es posible satisfacer las demandas actuales con equipos de desarrollo cerrados y tecnologías privativas, pues sus restricciones inhiben la colaboración, manteniendo tendencias con ciclos de desarrollo lentos.

Por otro lado las tecnologías liberadas con licencias permisivas y abiertas, permiten el continuo mejoramiento del producto software por parte de comunidades de desarrolladores activos, por lo que mejoras en las aplicaciones son liberadas en tiempos considerablemente reducidos.

En vista de las necesidades previamente descritas, la utilización de frameworks de programación de software libre, se propone como una solución tecnológica a un problema de desarrollo; en el caso de la web, los frameworks de programación de código libre proponen esta solución, compitiendo con otros frameworks basados en tecnologías privativas más comúnmente usados que sus contra partes libres³.

Sin embargo, fueron varias las respuestas proporcionadas por diferentes casas desarrolladoras de software, comunidades de entusiastas y/o individuos particulares. Por esta razón, existe actualmente un número muy grande de frameworks desarrollados con el fin de suplir este tipo de necesidades. Y es bajo esta situación que un desarrollador web se enfrentará con la disyuntiva generada a partir del gran número de opciones posibles para escoger.

3 THOMAS, Dave y Hansson. The Pragmatic Programmers: Agile Web Development with Rails, segunda edición. Raleigh, North Carolina 2007. p14

Teniendo en cuenta las anteriores razones, este proyecto plantea las bases para el estudio y utilización de diferentes frameworks de programación, en especial, siendo estos liberados bajo licencias representativas del software libre, que permitan el desarrollo de sistemas de información basados en web.

Para esto, se pretende establecer una guía para la selección de un framework que se adecue al desarrollo de una aplicación específica, mediante el planteamiento de un modelo de evaluación que permita realizar la selección de la herramienta de desarrollo, basada en un estudio comparativo.

Para demostrar el uso del modelo de evaluación, se compararán cuatro frameworks de programación libres basados en PHP. Finalmente y gracias al modelo de evaluación una de las opciones será seleccionada para demostrar su facilidad de uso a través del desarrollo de una aplicación enfocada al comercio electrónico.

1. PRÁCTICAS DE DESARROLLO

1.1 TENDENCIAS DE DESARROLLO EN LA WEB

A partir de 2001 se marca un momento crucial para la web, justo después de una crisis económica, característica que normalmente precede a alguna revolución tecnológica, demostrando cómo, cuando estas crisis suceden, son signo de que la tecnología está lista para establecer su lugar en el ámbito económico⁴.

O'Reilly junto con MediaLive International, establecieron el concepto de WEB 2.0 después de una sesión de *brainstorming* en la que descubrieron que, lejos de ser una tecnología acabada y con poco futuro⁵, como se creía en esa época, la web había evolucionado. Nuevas tecnologías aparecían en la red y lo hacían a grandes velocidades; sin embargo, el término WEB 2.0 no ha sido claramente especificado, llegando al punto de que se cree que es sólo un término de moda.

Desafortunadamente el término Web 2.0 es un *meme*^{*} que está siendo masificado no por su significado real, sino por el enfoque que el marketing de muchas compañías está dando a las tecnologías que comprenden este término. Esto lo convirtió simplemente en una palabra de moda sin aterrizar su significado, lo que conduce a una confusión obvia por lo que puede llegar a ser la Web 2.0.

4 O'REILLY, Tim. Qué es Web 2.0: Patrones de Diseño y modelos del negocio para la siguiente generación del software. O'Reilly Media, INC. 2006. p. 2

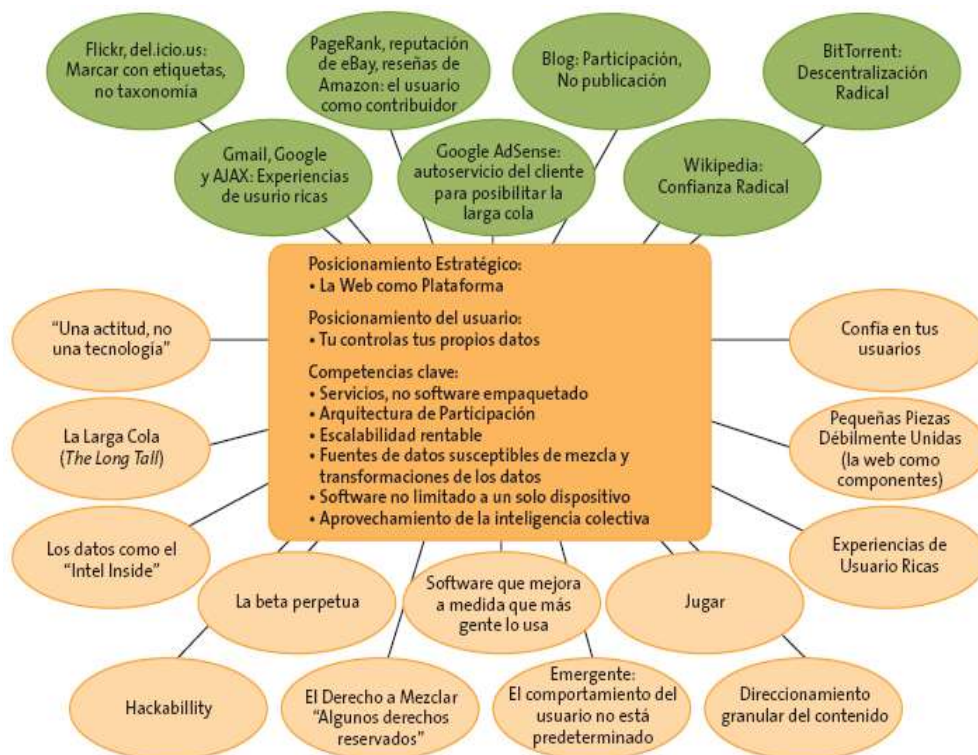
5 Wikipedia, The free encyclopedia. Crisis de las Puntocom. [en línea] Comunitario [Citado el 27 de Agosto de 2007] Disponible en internet: <http://es.wikipedia.org/wiki/Crisis_de_las_puntocom>.

* El término “meme” es un neologismo inventado por Richard Dawkins que se refiere a la información mínima acumulada en nuestra memoria y captada generalmente por imitación (mimesis), enseñanza o asimilación.

Debido a esto muchas de las empresas que promueven el término como producto de marketing, ni siquiera cuentan con tecnologías propias de la Web 2.0, mientras que algunas aplicaciones que se asocian comúnmente a la Web 2.0 (Napster, Bittorrent) no son en realidad aplicaciones web.

La **Figura 1**⁶, presenta un *meme* de la Web 2.0 desarrollado por Tim O'Reilly en su sesión de *brainstorming* durante el FOO Camp, una conferencia realizada en O'Reilly Media, y aunque no está completo del todo, presenta las variadas ideas provenientes del núcleo de la Web 2.0.

Figura 1. Meme del concepto de Web 2.0



Fuente: O'REILLY, Tim. Qué es Web 2.0.

6 O'REILLY, Tim. Qué es Web 2.0: Patrones de Diseño y modelos del negocio para la siguiente generación del software. O'Reilly Media, INC. 2006. p. 3

Aquellas compañías que se mantuvieron después de la crisis del punto-com⁷ y que ahora lideran la era de la Web 2.0 tienen una serie de características en común. Para Tim O'Reilly son las siguientes⁸:

- 1. Los hipervínculos conforman los cimientos de la web, a medida que nuevos sitios se crean y se agrega nuevo contenido a los existentes, se genera la estructura de la web, cuando otros usuarios descubren la información y la enlazan con sus propios sitios.*
- 2. Yahoo!, uno de los casos de éxito más conocidos, nació como un directorio de enlaces y aunque el esquema de trabajo de Yahoo! ha cambiado hacia la creación de nuevos tipos de contenido, su valor sigue radicando en su papel como portal de trabajo colectivo.*
- 3. La innovación de Google en el sistema de búsqueda fue PageRank, un método para usar la propia estructura de enlaces en la web para proporcionar mejores resultados en la búsqueda, en lugar de limitarse a escudriñar las características de los documentos.*
- 4. eBay crece a medida que sus usuarios interactúan con el sistema. eBay crece orgánicamente en respuesta a las actividades de sus usuarios y el papel de esta compañía es el de habilitar un contexto para que el usuario pueda desarrollar su actividad.*
- 5. La masificación de la blogósfera* y el uso de la tecnología actual para crear bitácoras personales ha traído un auge en la interacción de usuarios con la red. El uso de RSS permite suscribirse a una página y recibir notificaciones sobre la actualización de la misma. El uso de RSS es el avance más significativo en la arquitectura básica*

7 Wikipedia, The free encyclopedia. Crisis de las Puntocom. [en línea] Comunitario [Citado el 27 de Agosto de 2007] Disponible en internet: <http://es.wikipedia.org/wiki/Crisis_de_las_puntocom>.

8 O'REILLY, Tim. Qué es Web 2.0: Patrones de Diseño y modelos del negocio para la siguiente generación del software. O'Reilly Media, INC. 2006. p 11.

* Blogósfera es un neologismo referente a la masa de usuarios que interrelacionan en blogs, poseen un blog persona o desempeñan algún papel en alguno.

de la web desde que se descubrió que con el CGI se podían conseguir páginas dinámicas a partir de la información existente en una base de datos.

Teniendo en cuenta lo anterior, se puede determinar un patrón a seguir en las tendencias actuales, donde el verdadero proveedor de los servicios que se pueden encontrar en la web, es el usuario. Entre más relacionado esté el usuario con la web por la que navega, mejor servicio ésta prestará⁹.

Debido a esto, no solo el uso de la web ha cambiado, también se ha modificado la forma de desarrollar aplicaciones que presten éstos servicios en la web. Dado que el constante cambio en la información es algo común en la web, los nuevos sistemas de información, deben permitir su agilización, sin restringir las posibilidades de uso ni la cantidad de información procesable dentro del sistema.¹⁰

Cada vez más, las nuevas plataformas de desarrollo buscan incluir entre sus APIs y componentes, formas para facilitar el uso de las nuevas tecnologías web (RSS, AJAX... etc.)¹¹. Pero no solo eso, dado que una de las ideologías con estas tendencias web es la de permitir el acceso a la información, indiferente a la plataforma con la que se visualice, los nuevos entornos de desarrollo buscan también herramientas para mostrar dicha información en diferentes formatos (PDF, PostScript, XML... etc.) accediendo a ésta desde cualquier tipo de base de datos. Con este objetivo en mente es que los frameworks de programación web fueron creados.

9 OLSEN, Dave y Janzen. Blogging for Retailers. Elastic Path Software. Vancouver, Canada. Enero de 2007 p 3.

10 WILLIAMS, Justin. Rails Solutions: Ruby on Rails Made Easy, Friendsoft, California. 2007. p 16

11 GEHTLAND, Justin, Galbraith y Almaer, The Pragmatic Programmers: Pragmatic Ajax a Web 2.0 Primer, Raleigh, North Carolina 2007. 163 p3.

1.2 LOS FRAMEWORKS DE PROGRAMACIÓN

Con el término de framework, se hace referencia a una estructura de software que posee componentes personalizables e intercambiables para el desarrollo de una aplicación.

“Un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado”¹². Según lo anterior, el framework no solo es un componente software establecido, está determinado por un arreglo estructurado e inter-relacionado de otros componentes diferenciados entre sí (clases en Java, módulos en PHP, descriptores y archivos de configuración en XML), creando un árbol jerárquico donde las relaciones entre cada componente proporcionan la funcionalidad al framework.

Normalmente se puede considerar al framework como un componente incompleto pero configurable, pues a medida que se desarrolla, se añaden, se eliminan o se crean nuevos módulos¹³, conformando un diseño reutilizable que agiliza y facilita las labores de desarrollo de sistemas. Usualmente un framework incluye soporte de programas, librerías y un lenguaje de scripting que ayudan a desarrollar y unir los diferentes componentes de un proyecto de software.

Es por esto que un framework agrega funcionalidad extendida a un lenguaje de programación, automatizando patrones de desarrollo y permitiendo una orientación hacia un propósito fundamental del proyecto.

12 GUTIERREZ, Andres Felipe. Kumbia PHP Framework: Porque programar debería ser más fácil. Comunidad Kumbia [en línea] Colombia [Citado el 29 de Agosto de 2007]. Disponible en internet: <<http://www.kumbia.org/LibroDeKumbia.pdf>>. p 17.

13 D’SOUZA. Desmond, Francis y Wills. Objects, Components and Frameworks with UML, Addison Wesley, California, October 1998. p 339.

Existe una gran variedad de frameworks en el mercado, muchos de los cuales son liberados bajo la licencia GNU/GPL, insignia del software libre. La ventaja de utilizar estos frameworks de código abierto, además de las facilidades que de por sí presta a la hora de desarrollar, radica en la posibilidad de adquirir la herramienta de trabajo a un precio prácticamente nulo, sin preocuparse por problemas de licencias y consiguiendo un soporte dado por la comunidad que permite que el desarrollo del framework se acelere de manera exponencial, con lo que nuevas utilidades, componentes y actualizaciones saldrán de forma más periódica¹⁴.

1.2.1 Utilidad de un framework. La idea de utilizar un framework a la hora de desarrollar aplicaciones es la de disminuir el tiempo de desarrollo, reutilizar código existente, y promover buenas prácticas de desarrollo como el uso de patrones o estándares. Bajo esa idea, el uso de frameworks define una filosofía de trabajo, donde el objetivo de un proyecto puede ser sintetizado en uno solo y es en su alcance, donde el desarrollador se centrará, dejando los detalles de operación a las estructuras del framework.

Adentrándose en el desarrollo web, una aplicación termina siendo, a la final, una interfaz de usuario con una serie de formularios y presentaciones extraídas de bases de datos, texto plano, imágenes y applets o estructuras en javascript. Es así como un usuario concibe la web a la hora de navegar, pues su interés no va más allá de lo que el navegador le está presentando.

Debido a esto el diseño de páginas web se sintetiza en simplemente unas cuantas instrucciones que dan formato e interactividad al usuario. Esta serie de instrucciones embebidas en el HTML se repiten una y otra vez a lo largo de un

14 Wikipedia. Web Application Framework. [en línea] Comunitario [Citado el 27 de Agosto de 2007] Disponible en internet:<http://en.wikipedia.org/wiki/Web_application_framework>

portal web, por lo que el trabajo de desarrollo partiendo desde cero, puede convertirse en una tarea repetitiva y poco productiva pues se programa el mismo código una y otra vez.

Con la ayuda de un framework, es posible implementar esa serie de código repetitivo de una manera eficiente y sin tener que partir desde cero. Es posible establecer diferentes parámetros para cada estructura que se desea implementar, pero el desarrollo parte ya desde una base pre-programada en el framework. Para esto existen diferentes librerías, funciones y APIs que están disponibles a la hora de desarrollar aplicaciones con un framework.

La utilidad de un framework radica en la estructura de esos componentes que se encuentran ya definidos, por eso, al disponer de ciertas instrucciones predefinidas, permite que el desarrollador ahorre tiempo en el diseño e implementación de dichas estructuras, por lo que puede dedicarse plenamente a solucionar los problemas claves del proyecto que desarrolla, sin perder tiempo en detalles del sistema. Esto implica también que el código que va a ser escrito carecerá de módulos repetidos y código duplicado por lo que el mantenimiento del sistema resultará también mucho más sencillo y práctico.

1.3 PATRONES DE DISEÑO EN LOS FRAMEWORKS

Dado que un framework posee una estructura definida de instrucciones que proveen de un medio práctico a los desarrolladores para crear soluciones informáticas a un grado de productividad bastante elevado, hay que pensar que el desarrollo de aplicaciones basadas en algún framework debe seguir un patrón que permita estandarizar la forma en que debe realizarse un proyecto de software.

Christopher Alexander dice, “Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, y luego describe el núcleo de la solución a ese problema, de tal manera que se pueda usar esa solución un millón de veces sin tener que hacerlo de la misma manera dos veces”¹⁵. Aún cuando Alexander se refiriera a patrones de construcción de edificaciones y pueblos, la esencia de lo que es un patrón está plasmada en esa frase. En general, un patrón debe contar con cuatro elementos esenciales¹⁶.

1. **El nombre del patrón** es una herramienta que se puede utilizar para describir el problema de diseño al que se pretende atacar, sus soluciones, y las consecuencias en una o dos palabras.
2. **El problema** describe cuando aplicar un patrón. Pero no solo se debe describir el problema sino también su contexto, ya que un mismo problema en diferente situación puede requerir una solución diferente. El problema debe describir a su vez problemas del mismo diseño, tales como la representación algorítmica de cierto tipo de funciones u objetos. Debe describir las estructuras de las clases, objetos o funciones, representantes de un diseño inflexible, y en algunas ocasiones, también debe especificar una lista de condiciones que se deben cumplir antes de que tenga sentido la utilización del patrón.
3. **La solución** describe los elementos que conforman el diseño. las relaciones entre sí, sus funciones y responsabilidades. Sin embargo, la solución no describe en concreto que diseño debe implementarse, dado que un patrón puede ser usado en diferentes condiciones. Por esto, la solución debe abstraerse un

15 CHRISTOPHER, Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, Shlomo Angel. A Pattern Language. Oxford University Press, New York, 1977. p 19.

16 GAMMA, Erich, Helm, Johnson y Vlissides. Design Patterns. Elements of Reusable Object- Oriented Software. Addison Wesley, USA, 1995.

poco más y representar una descripción de cómo diferentes funciones, clases y objetos pueden solucionar el problema.

4. **Las consecuencias** son el resultado de aplicar un patrón a un problema de diseño. Aunque las consecuencias solo pueden evaluarse al final, después de haber aplicado un patrón, estas son esenciales para evaluar diferentes alternativas de diseño y entender los costos y beneficios del uso del patrón.

Dependiendo de las perspectivas se puede discernir sobre lo que es y lo que no es un patrón. Para este proyecto se recurrirá al punto de vista descrito por Erich Gamma en su libro Design Patterns: Elements of Reusable Object-Oriented Software. Para Erich Gamma los patrones de diseño son “... descripciones de objetos que se comunican y de clases que son modificadas para resolver un problema general de diseño en un contexto en particular”¹⁷.

Un patrón de diseño permite nombrar, abstraer e identificar diferentes aspectos claves de una estructura de diseño que hacen que esta sea útil para la creación de diseños orientados a objetos que sean reutilizables. El patrón de diseño identifica las clases que deben participar en el diseño, sus instancias y sus roles en la estructura. De esta manera el patrón define a cada clase un objetivo en particular, y describe cuando se aplica dicha clase en la solución de un problema de diseño orientado a objetos.

17 GAMMA, Erich, Helm, Johnson y Vlissides. Design Patterns. Elements of Reusable Object Oriented Software. Addison Wesley, USA, 1995. p 10

1.4 PATRÓN MODELO – VISTA – CONTROLADOR

Para entender cómo trabajan los frameworks web actuales, es necesario entender este patrón de diseño conocido como Modelo-Vista-Controlador (MVC). Este patrón es una guía para el diseño de arquitecturas de aplicaciones que ofrecen una gran interactividad con el usuario. La idea con este patrón es organizar, de forma separada, las aplicaciones en tres componentes diferenciados entre sí.

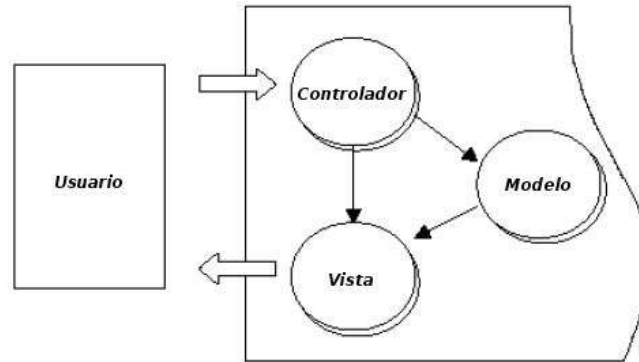
La triada MVC es comúnmente usada para construir interfaces de usuario. Esta arquitectura consiste de tres tipos de objetos. El **Modelo** es el objeto de aplicación. La **Vista** es la presentación en pantalla. Y el **Controlador**, define la manera en que la interfaz de usuario reacciona a las peticiones del usuario.

Antes de la definición de la arquitectura MVC, el diseño de interfaces de usuario tendía a combinar todos esos objetos en uno solo. El MVC los separó para incrementar la flexibilidad y reusabilidad del patrón. Un esquema de cómo se comporta una aplicación según el patrón MVC se puede ver en la **Figura 2**.¹⁸

El Modelo se encarga de mantener el estado de la aplicación. En algunas ocasiones ese estado es transitorio y dura por un par de interacciones con el usuario. En otros casos el estado será permanente y será almacenado fuera de la aplicación, usualmente en una base de datos

¹⁸ RAIBLE, Matt. Comparing Web Frameworks: Struts, Spring, MVC, Webwork, Tapestry & JSF. Virtuas. United States. 2005. 43 p

Figura 2. Esquema de la arquitectura MVC



Fuente: GEHTLAND, Justin, Galbraith y Almaer, The Pragmatic Programmers

Sin embargo, el modelo es más que simples datos; este contiene todas las reglas del negocio que dan estructura e importancia a los datos que el sistema utiliza. Poniendo un ejemplo, en una aplicación de comercio electrónico un producto posee descuento si este tiene un valor mayor a los US\$40 (valor descrito en el modelo de datos). En caso de que el producto no posea ese mínimo valor, el modelo se encargara de no aplicarle el descuento. En caso contrario el modelo hará cumplir el descuento al producto.

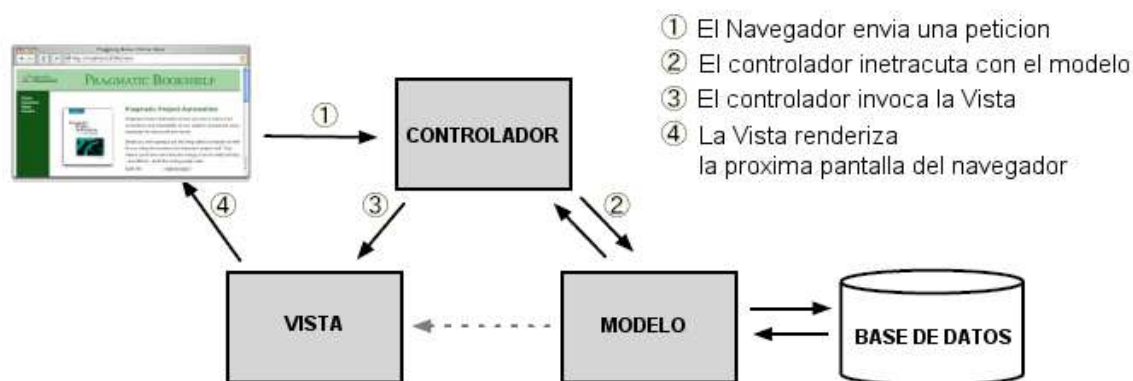
La vista es responsable por la generación de la interfaz del usuario, generalmente basado en los datos del modelo. Como ejemplo se puede tener a una tienda virtual que tendrá en alguna de sus secciones una página con una lista de productos mostrados en un catalogo. Esta lista será accesible vía el Modelo, pero existirá también una vista que accederá a la lista a través del modelo y le dará el formato para el usuario final.

Aunque la Vista presenta al usuario una manera para acceder información al sistema, esta no se encarga de ese flujo de datos ya que el trabajo de la capa de

Vista finaliza una vez la información es mostrada. Un ejemplo gráfico de la interacción del patrón se presenta en la **Figura 3**¹⁹.

El trabajo de manipular la información procedente del usuario le corresponde a la capa del Controlador. Este se dedica a recibir datos, procesarlos dentro del modelo y presentar la capa de Vista adecuada una vez los datos han sido procesados.

Figura 3. Ejemplo de interacción del patrón MVC



Fuente: GEHTLAND, Justin, Galbraith y Almaer, The Pragmatic Programmers.

Adoptando la arquitectura de Modelo Vista Controlador, un framework al final busca no solo facilitar el trabajo del desarrollador, sino también, el desarrollo de aplicaciones más intuitivas para el usuario final, procurando que las interfaces, la lógica y los procesos sean tan prácticos como sea posible. De esta manera el

¹⁹ GEHTLAND, Justin, Galbraith y Almaer, The Pragmatic Programmers: Pragmatic Ajax a Web 2.0 Primer, Raleigh, North Carolina 2007. p 74.

desarrollador evita estar pensando en una serie de detalles que pueden distraerlo a la hora de desarrollar una aplicación que resulte cómoda para los usuarios y se centra en atacar necesidades reales sin tener que partir de supuestos de utilidad.

2. DISEÑO DE LA METODOLOGÍA DE EVALUACIÓN DE FRAMEWORKS

A pesar de que el concepto de aplicación web es entendido en la jerga común como una generalización que reúne a las diferentes clases de aplicaciones existentes en Internet (buscadores, sistemas e-commerce, blogs, portales de descargas, etc.) y que conforman a un sitio web como tal, la realidad es que la diversidad de los tipos de aplicaciones es tanta como la de los usuarios que pueden llegar a utilizarlas.

Bajo esta premisa, es imposible seleccionar un único framework como “óptimo”, que permita el desarrollo de “aplicaciones web”, entendiendo este concepto como la generalización de las aplicaciones a las que representa, debido a que no todos los frameworks atacan un problema desde la misma perspectiva ni están enfocados a las soluciones de los mismos problemas.

Para poder hacer una selección acertada, primero es necesario establecer el tipo de aplicación web que se desea realizar. Con base en esto será posible luego establecer los requerimientos de dicha aplicación y finalmente teniéndolos en cuenta, será posible la selección de un framework adecuado que permita la agilidad y calidad en el desarrollo de dicha aplicación.

2.1 MODELO SQ-MET.

A pesar de lo sencillo que pueda parecer la clasificación de una aplicación web, la realidad es que es bastante más complicado de lo que se cree. En caso de que

una aplicación contenga funcionalidades que puedan ser atribuidas a más de un solo tipo de aplicación web (un buscador por ejemplo puede pertenecer a un sistema de comercio electrónico o a un portal de descargas), deberían asignársele los requisitos correspondientes a todos los tipos de aplicaciones a los que dicho requerimiento pertenezca²⁰.

2.1.1 Facetas de una aplicación Web. Como se dijo antes, una aplicación web puede contener multitud de requerimientos que no sean realmente propios de dicho tipo de aplicación. *“La excesiva heterogeneidad en la funcionalidad de las aplicaciones web actuales hace que prácticamente todas las aplicaciones posean funciones o facetas atribuibles a todos los tipos de aplicaciones”*²¹. Para ejemplificar esta afirmación, se escogieron cuatro clases de páginas web que a simple vista un usuario común no dudaría en catalogar.

Para este ejemplo se cuenta con el *buscador* www.google.com, el *portal de noticias* www.eltiempo.com, el *portal de descargas* www.download.com, y el sistema *e-commerce*, www.ebay.com. Desde un principio cualquier usuario estaría de acuerdo con esta clasificación, pero ésta comienza a verse más complicada cuando, entrando en detalle, se aprecia que las facetas de una categoría pueden ser encontradas en otra.

Por ejemplo, a pesar de que www.ebay.com fuese escogido como una aplicación de comercio electrónico, lo cierto es que dentro de sus facetas existe también un buscador interno y externo, por lo que comparte similitudes con www.google.com (escogido en un principio como un buscador). El portal de

20 CORTAZAR, Rebeca. SQ-MET: Desarrollo de una Metodología de Validación de Aplicaciones Internet. Departamento de Ingeniería del Software. Universidad de Deusto. p. 1

21 CORTAZAR, Rebeca. Análisis y aportaciones a la metodología SQ-MET. Departamento de Ingeniería del Software. Universidad de Deusto. p. 3

descargas www.download.com tiene la opción de restringir secciones a usuarios registrados, creando listas de accesos para sus usuarios, algo muy típico de aplicaciones e-business o e-commerce (como www.ebay.com).

Sin embargo, resulta ilógico a nivel de ingeniería de software, evaluar los requerimientos de una aplicación en base a otra que tenga funcionalidades similares pero que su objetivo como aplicación sea totalmente diferente, pues más allá de la similitud entre las funciones, las reglas de negocio entre una y otra aplicación varían dependiendo del tipo de usuario al que se pretenda llegar, y de las políticas empresariales y/o comerciales establecidas para el proyecto.

2.1.2 Método de calificación en SQ-MET. Rebeca Cortazar, en su proyecto para crear una metodología para la evaluación y el aseguramiento de calidad de aplicaciones web (SQ-MET) se enfrentó con el mismo dilema. Para evitar caer en una ambigüedad en el momento de evaluar una aplicación, Cortazar propone evaluar las aplicaciones no por su categoría, como en un principio estaba pensado (y como se ejemplificó anteriormente), sino evaluarla por sus facetas, entendiendo como faceta “cualquier aspecto de una aplicación web que dé lugar a la adición de nuevos requerimientos”²².

Como ejemplo de una faceta Cortazar propone la existencia de *almacenamiento de datos confidenciales*. La existencia de esta faceta solo sería la causante de la generación de nuevos requerimientos como la necesidad de interacción del sistema con una base de datos externa, la restricción de la información a diferentes usuarios que no tengan los permisos necesarios, etc.

Sin embargo, Cortazar y su equipo de trabajo se dan cuenta de la necesidad de

²² Ibid., p. 3.

establecer un valor para cada faceta a la hora de realizar la evaluación de la aplicación web. Hay que entender que no todas las facetas presentes en una aplicación web tienen el mismo valor, es decir, mientras que algunas facetas se hacen indispensables en cierta aplicación, otras simplemente pueden estar dando un valor agregado a la misma, mas no influyendo realmente en la funcionalidad de la aplicación.

Por esta razón el modelo de evaluación debió modificarse para incluir una variable que altere la puntuación dada a una faceta y que permita darle el valor real que ésta debe tener dentro de la aplicación, de esta manera se evita que facetas menos importantes influyan sobre el porcentaje total de la evaluación de una aplicación y que arrastren los puntajes obtenidos por las facetas más influyentes.

2.2 ADAPTACIÓN DEL MODELO SQ-MET AL MODELO DE EVALUACION DE FRAMEWORKS.

Dado que el modelo de evaluación SQ-MET está orientado al aseguramiento de calidad de una aplicación web y no a la selección de un framework de programación, se deben realizar ciertos ajustes para adaptar la metodología a los requerimientos de la evaluación que en este proyecto se realiza, principalmente en la definición de las facetas, dado que en una framework las facetas como tales no existen.

A diferencia de una aplicación Web, en un framework de programación, en vez de facetas, existen las opciones necesarias para desarrollar las facetas presentes en una aplicación web. Por esta razón al modelo se le agregó el término **módulo**. Los **módulos**, a diferencia de las facetas, *son todas aquellas funcionalidades*

presentes en el framework listas para utilizarse en el desarrollo de una faceta de alguna aplicación.

Como ejemplo de un **módulo**, se pueden presentar todas las opciones de conectividad a una base de datos y/o la forma en que los datos son extraídos y presentados al usuario, reunidas dentro de una o varias clases o funciones. Otro ejemplo puede ser la utilización de clases o funciones que permitan la creación y manipulación de listas de acceso, listas de correo o documentos en formatos portables como PDF.

Es posible que un **módulo** utilizado correctamente pueda dar cabida a la generación de una faceta dentro de una aplicación web. Pero también existe la posibilidad de que varios **módulos** se complementen entre sí para la generación de una faceta concreta, sin que esto signifique que el framework sea menos funcional.

2.3 DEFINICIÓN DEL MODELO DE EVALUACIÓN DE FRAMEWORKS

Dado que la idea con la realización de éste modelo de evaluación es, que sea una herramienta general para escoger el framework más óptimo para desarrollar un tipo de aplicación web específico, se vio la necesidad de adoptar el sistema de facetas para darle versatilidad al modelo y no limitarlo simplemente a cierto tipo de aplicaciones.

Es por eso que, antes de iniciar con el uso del modelo, se deben establecer las facetas que la aplicación web deberá tener, con el fin de enmarcar la evaluación hacia un cierto tipo de aplicación definido por dichas facetas.

Es decir, es necesario tener claro qué tipo de aplicación se desarrollará y cuáles son sus requerimientos (generados a partir de las facetas), pues sabiendo esto será posible mas adelante establecer los **módulos** que el framework deberá tener para ser usado y que realmente facilite el desarrollo de dicha aplicación.

Una vez teniendo claro esto, es posible ver (mas allá de lo que hasta ahora se plantea) que el modelo podrá ser usado para cualquier aplicación, pues los parámetros para evaluar el framework son determinados por los requerimientos del software a desarrollar y no son simplemente parámetros estáticos o estándar que pueden o no estar presentes como requerimientos de una aplicación.

Una vez definidos los requerimientos del software a través de sus facetas, se puede plantear, qué tipo de módulos deberá tener el framework de programación web que permitan agilizar el proceso de desarrollo de dicha aplicación.

2.3.1 Definición de los Módulos. Trayendo de vuelta el ejemplo de faceta dado por Cortazar, si se plantea como requerimiento del software el tener *almacenamiento de datos confidenciales* dentro de la aplicación, es evidente la necesidad de desarrollar un mecanismo de conexión entre la aplicación y la base de datos. De igual manera se ve la necesidad de que esa información almacenada, sea únicamente accesible por los usuarios que tengan los permisos requeridos para hacerlo, por lo que se ve la necesidad de crear listas de acceso que faciliten el control de este caso de uso.

Con lo anterior se pueden establecer los dos primeros módulos a manera de ejemplo para ser evaluados en el framework. El primero de los módulos está definido por las herramientas con las que cuenta el framework para realizar las conexiones con la base de datos y se le dará el nombre de **DBC** (del ingles Data

Base Connection). El segundo de los módulos esta dado por las herramientas del framework que faciliten la generación de listas de acceso y se llamara **ACL** (del inglés Access Control Lists). Teniendo ambos módulos es posible realizar una evaluación para determinar que framework es el más óptimo para este tipo de aplicación.

Dado que no todos los frameworks son iguales, y no todos se orientan hacia el desarrollo del mismo tipo de aplicaciones, la forma más sencilla de realizar la evaluación es mediante la puntuación del módulo por su existencia o inexistencia dentro del framework. Así, si el módulo existe dentro del framework tendrá una puntuación del 100%, y por el contrario si el módulo para realizar dichas facetas no existe obtendría una puntuación de 0%.

Es posible también que, para dar un mayor nivel de detalle, el modulo sea calificado además por su complejidad o funcionalidad. Es decir, si el framework contiene las herramientas (**módulos**) para desarrollar una faceta, pero ésta herramienta es incompleta funcionalmente hablando, o solo realiza parte de lo que el requerimiento exige, es obvio que no debería obtener una puntuación similar al de otro framework que sí contenga el **módulo** de forma completa y permita el desarrollo de la faceta que satisfaga el requerimiento.

La razón de esta forma de evaluación radica en la necesidad de escoger el framework a utilizar en el menor tiempo posible. Es por eso que se evita entrar en demasiado detalle, pues la idea es que este modelo sea práctico a la hora de iniciar un proyecto de desarrollo y evite tener que lidiar con demasiadas complicaciones generadas por el estudio.

2.3.2 Definición del peso de un módulo. Como ya se dejó claro anteriormente y tomando como base nuevamente la metodología SQ-MET, la definición de unos pesos que normalicen la evaluación de los módulos en el framework se hace evidente para obtener un resultado más acorde a las necesidades de un proyecto de desarrollo.

Para este hecho, existen varias formas de establecer dichos pesos, que se diferencian entre sí por su objetividad o subjetividad (conceptos no necesariamente excluyentes en la práctica). El primer método está basado en la intuición sin referirse a este concepto en un sentido peyorativo, como lo expone Olsina en su propuesta de evaluación de calidad de sitios web, y por el contrario refiriéndose al conocimiento claro, íntimo e instantáneo generado a partir de la experiencia o práctica²³.

Este primer método carece de un soporte sustentable ya sea matemático o algorítmico que permita dar razón del por qué de un valor u otro. Sin embargo no deja de ser un método funcional y válido. Además en la mayoría de los casos será el ejercido por los desarrolladores a la hora de utilizar el modelo de selección de frameworks expuesto en este proyecto debido a la rapidez que permite este método para la toma de decisiones.

Para proporcionar un sistema más confiable donde la experiencia como tal no es requerida para determinar la importancia de un módulo en un framework, este proyecto propone un sistema de métricas compuesto por una matriz de importancia de los módulos, basado en la matriz de MOODY's para la toma de decisiones²⁴. De esta manera, en la ausencia del conocimiento necesario, es posible establecer unos pesos a través de unas operaciones matemáticas

23 OLSINA, Luis Antonio. Metodología Cuantitativa para la Evaluación y Comparación de la Calidad de Sitios Web . Universidad Nacional de La Plata, Facultad de Ciencias Exactas. Argentina. 1999

24 MOODY, Paul. Toma de decisiones gerenciales. McGraw Hill. Bogotá, 1991.

bastante sencillas pero que proporcionan de un sustento a la selección de pesos para el modelo.

Supóngase que se tiene un arreglo de módulos $X = \{ m1, m2, m3, m4 \}$, donde cada uno de los elementos es uno de los módulos a evaluar en la metodología para el framework X . Si se hace una comparación de la importancia del modulo mi con respecto a los demás módulos, determinando si es más o menos importante que los demás mediante un valor entero, al final a través de la sumatoria de los valores especificados en cada comparación es posible determinar un porcentaje de importancia del módulo.

Es necesario establecer desde un principio cuales serán los valores aceptados por la matriz de toma de decisiones. Para la propuesta dada en este proyecto, se consideran tres valores aceptados, definidos por la importancia de un modulo con respecto a otro. Un módulo puede ser *menos importante* que otro, *igual de importante* que otro o *más importante* que otro. Teniendo en cuenta esa escala de importancia, cada nivel puede ser determinado por un valor que represente la importancia del módulo en el modelo. La **tabla 1** presenta los valores establecidos para el modelo de acuerdo con la escala de importancia de un módulo.

Tabla 1. Valores para la escala de importancia de un módulo

Menos importante	0
Igual de importante	1
Más Importante	2

Fuente: Autor del proyecto

Teniendo los posibles valores que obtendrá un módulo en el modelo, se puede realizar la comparación directa de cada módulo con respecto a los demás y de esta manera establecer su importancia. En la **tabla 2** se refleja la comparación con tres módulos de ejemplo.

Tabla 2. Comparación de los módulos

$m1 - m2$	$m1 - m3$	$m1 - m4$	$m2 - m3$	$m2 - m4$	$m3 - m4$
0	2	1	2	2	2

Fuente: Autor del proyecto

Los valores aceptado para llenar la matriz están definidos por la importancia del primer modulo con respecto al segundo. Así es posible ver como el módulo 1 ($m1$) es definido como *menos importante* que el módulo 2 ($m2$) y como el módulo 2 es definido como *más importante* que el módulo 3 ($m3$). En el anterior ejemplo solo un participante en la evaluación decidió la puntuación en la comparación pero el modelo no está sujeto ni limitado por la cantidad de participantes.

En caso de que más de un participante realice la ponderación de la importancia de los módulos el resultado final será la sumatoria de los puntajes de todos los participantes, tal como lo muestra la **Tabla 3**.

A diferencia de lo presentado en la **Tabla 2**, en este caso los puntajes están determinados por la sumatoria y no solo por lo estipulado por uno de los participantes. El número de participantes es determinante a la hora de establecer los pesos. Para continuar con el ejercicio, se supondrá que solo un participante realizará la ponderación

Tabla 3. Comparación de los módulos con más de un participante.

	m1 – m2	m1 – m3	m1 – m4	m2 – m3	m2 – m4	m3 – m4
Participante 1	0	2	1	2	2	2
Participante 2	1	2	2	1	2	0
Total	1	4	3	3	4	2

Fuente: Autor del proyecto

Una vez obtenidos los puntajes con referencia a la importancia de los módulos con respecto a los demás, es necesario organizar dichos resultados en una matriz que permita la tabulación de los datos de manera más sencilla. La **Tabla 4** presenta dicha matriz, tabulando los resultados obtenidos de la **Tabla 2**.

Tabla 4. Calculo del peso de los módulos.

	M1	M2	M3	M4		Puntaje	Peso
M1	1	0	2	1	=	4	0,250
M2	2	1	2	2	=	7	0,438
M3	0	0	1	2	=	3	0,188
M4	1	0	0	1	=	2	0,125
Total	4	1	5	6	=	16	1,000

Fuente: Autor del proyecto

Nótese que la diagonal principal de la matriz (las regiones sombreadas) tienen el valor correspondiente a la cantidad de participantes de la evaluación. En caso de la existencia de más participantes, estos valores deben modificarse para que reflejen el efecto generado por el aumento en las puntuaciones debido la existencia de más participantes.

Ahora bien, teniendo la matriz organizada de manera que presente la puntuación dada por la comparación, la suma de los puntajes obtenidos por cada modulo representan su puntuación total, esta sirve de base para sacar en términos de porcentaje, el peso de dicho módulo en el modelo. Para el caso del ejemplo se pudo determinar que el módulo 1 (m1) tiene un peso equivalente en el modelo al 25% (0,25), el módulo 2 (m2) al 43% (0,43) y el módulo 3 (m3) y módulo 4(m4) obtuvieron unos pesos del 18,8% (0,188) y 12,5% (0,125) respectivamente. La suma de estos porcentajes debe dar como resultado un 100%, de lo contrario, los cálculos se consideran errados.

Como se dijo anteriormente, este modelo no evita la existencia de la subjetividad a la hora de establecer los pesos, pues la importancia de un módulo sigue estando determinada por lo que el evaluador considere pertinente. Esto fue considerado a la hora de plantear el modelo y se determino que, para la viabilidad del mismo, se estableciera de ésta manera, pues en muchas ocasiones, la importancia de un módulo está determinada por las reglas del negocio establecidas para cada proyecto y/o sus requerimientos no estrictamente funcionales.

Por esta razón este método no permite determinar cuál de los módulos es más importante que otro, aunque si permite establecer en qué medida (dado en porcentaje) es más o menos importante, con lo que se puede eliminar la aproximación dada por el hombre y disminuir el rango de error generado al establecer pesos a través de la estimación o la simple intuición.

2.3.3 Definición de la puntuación de un módulo. Dado que, lo que se busca con este modelo de evaluación es realizar una selección mediante un proceso simple y que no consuma demasiado tiempo (la rapidez es esencial en el desarrollo de aplicaciones web), la puntuación en este modelo seguirá un modelo

de agregación. Con esto se evita un modelo demasiado sensible a las variaciones.

Este modelo no define un nivel de detalle máximo a tomar, y puede estar sujeto a las determinaciones de quien aplique el modelo. Para este proyecto en particular se propone un nivel de detalle de dos niveles. Es decir se propone trabajar con Módulos y Sub-módulos. Más allá de esto se considera que el peso determinado para cada uno de los módulos establecidos en un nivel superior al tercero (un Sub-módulo del Sub-módulo) no será lo suficientemente significativo y convertirá al modelo en un proceso demasiado sensible y poco reutilizable.

Entonces, definiendo una cantidad n de módulos y una cantidad m de Sub-módulos por cada uno de los módulos especificados, asignando los pesos a cada uno de acuerdo con lo establecido en la sección anterior, se puede determinar el puntaje obtenido por cada módulo (no el sub-módulo) mediante la sumatoria de los puntajes asignado a cada Sub-modulo. Para explicar lo anterior desde el punto de vista matemático, se debe definir un Identificador Elemental que represente la calificación de un Sub-modulo de la siguiente manera:

$$0\% \leq IE \leq 100\%$$

Una vez asignado el IE a cada uno de los sub-módulos se puede calcular el puntaje de un módulo determinado como el Identificador Global mediante la siguiente fórmula:

$$IG = (IE1 \times P1) + (IE2 \times P2) + \dots + (IEm \times Pm)$$

La variable P_i está determinada por el peso hallado mediante la matriz de importancia de cada sub-módulo. Nuevamente estos valores deben estar normalizados siguiendo lo establecido en la siguiente expresión:

$$0\% \leq P_i \leq 100\%$$

$$\text{Para } i = 1, 2, 3 \dots, m \text{ y } P_1 + P_2 + \dots + P_m = 100\%$$

Finalmente la puntuación final que determina la eficiencia del framework con respecto al tipo de aplicación a desarrollar esta dada por la sumatoria de todos los Identificadores Globales (IG) multiplicados por cada uno de sus pesos, tal y como lo muestra la siguiente fórmula:

$$PF = (IG_1 \times P_1) + (IG_2 \times P_2) + \dots + (IG_n \times P_n)$$

La comparación entre los distintos PF será el punto de referencia para la selección del framework. Finalmente, la **Tabla 5** presenta el esquema general de evaluación de un sub-módulo. Esta tabla representa los parámetros a tener en cuenta en la evaluación así como la métrica a seguir y el procedimiento de medida.

Tabla 5. Tabla de puntuación de un Sub-módulo

Sub-Módulo	<i>Nombre del Sub-módulo</i>
Métrica	<i>Existencia o Inexistencia del módulo en el framework.</i>
Valores aceptados	<i>0 = inexistencia, 1 = existencia parcial, 2 = existencia total.</i>
Medida	<i>Manual, Observación.</i>

Fuente: Autor del proyecto

2.3.4 El punto de vista del programador. Evidentemente, la decisión de usar o no un framework u otro la toma un desarrollador de software. Es por eso que se

considera importante establecer también una serie de parámetros que permitan calificar de alguna manera la respuesta dada por un framework a los intereses de un programador.

Muchas veces es preferible optar por una herramienta sin tantas prestaciones pero que pueda ser utilizada rápidamente, donde su documentación sea amplia y detallada y el soporte brindado genere confianza en el desarrollador. Este tipo de cosas deben ser tenidas en cuenta a la hora de evaluar un framework, pues un framework poco documentado y con poco soporte no puede proveer de una solución sólida a un producto empresarial o comercial en el que la estabilidad tanto de la aplicación como de la tecnología en la que se sustenta son un factor primordial.

Dado que en este proyecto se limitó el mercado de frameworks exclusivamente a unos cuantos de código libre (por las razones ya explicadas anteriormente), varios de los parámetros que deben ser evaluados desde el punto de vista de un programador terminan centrándose en la comunidad desarrolladora del framework. La metodología sugerida para la evaluación de los parámetros deseada por un programador, no difieren en la realidad de la metodología para evaluar los módulos de un framework.

Bajo el nombre de “generalidades” una serie de parámetros pueden ser especificados de manera que, en el modelo de evaluación, jueguen un papel como el de “sub-módulos”, por lo que pueden ser puntuados a través de la observación de la comunidad a la que se evalúa partiendo de los mismos principios con los que se evalúan los sub-módulos. De igual manera los pesos deben ser especificados e incluidos en los cálculos. La **Tabla 6** presenta la tabla de puntuación de estas generalidades.

Tabla 6. Tabla de puntuación de las Generalidades

Generalidad	<i>Nombre del parámetro a evaluar</i>
Métrica	<i>Experiencia del desarrollador frente al parámetro en el framework o en la comunidad.</i>
Valores aceptados	<i>0 = Mala experiencia; 1= Indiferente; 2 = Buena experiencia</i>
Procedimiento de medida	<i>Manual, Observación.</i>

Fuente: Autor del proyecto

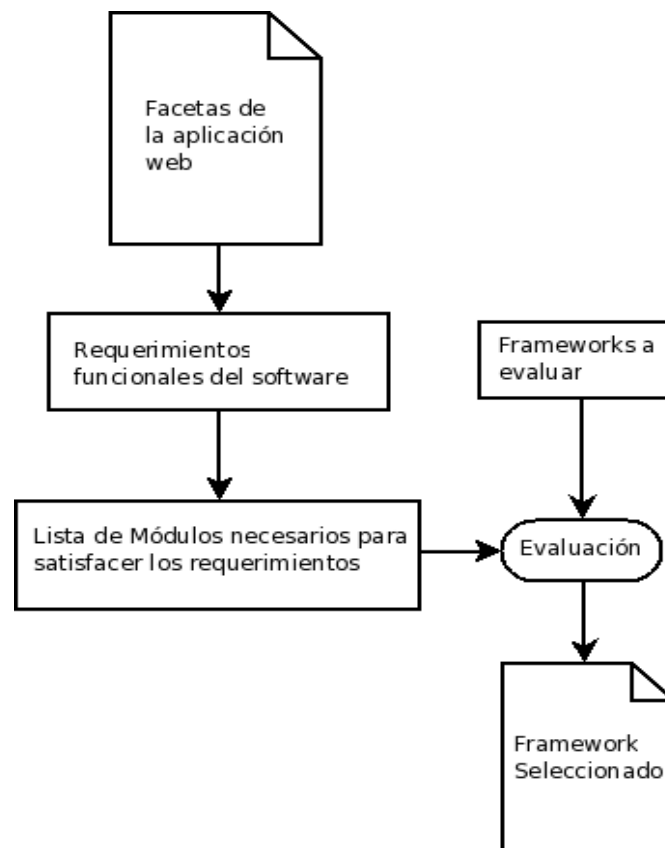
Si se comparan las tablas de puntuación de un sub-módulo con la de las generalidades, se pueden ver cambios sutiles en el contexto de la evaluación. En las generalidades el parámetro a evaluar no necesariamente se encuentra en el framework y no necesariamente es un requerimiento funcional del framework.

Como ejemplo se tiene el parámetro de **configuración**. Este parámetro no es realmente una funcionalidad del framework pero un desarrollador puede ver en él, una ventaja si la configuración de un framework es mucho más sencilla que en otros.

Por otro lado existen generalidades pertenecientes a la comunidad como lo puede ser la **documentación**. Esta depende enteramente de los desarrolladores del framework y tampoco es una característica funcional del mismo. Sin embargo es un parámetro que debe ser tomado en cuenta, sobre todo si el desarrollador no conoce el código del framework con el que posiblemente trabaje.

Para sintetizar toda la teoría aquí presente la **Figura 4** presenta a modo gráfico una abstracción del modelo de evaluación propuesto en este trabajo. La secuencia presente en el diagrama representa los pasos sugeridos para el desarrollo de esta metodología de evaluación.

Figura 4. Diagrama de secuencia del Modelo de evaluación de Frameworks.



Fuente: Autor del proyecto

3. SISTEMAS DE COMERCIO ELECTRÓNICO

La definición del modelo de evaluación, como se planteo anteriormente, debe estar regido por las facetas que determinarán la utilidad del framework. Realizando una especie de “ingeniería inversa”*, es posible partir de la aplicación e ir descubriendo que módulos conforman la aplicación que lo hacen funcionar de la manera en que lo hace en los casos (como el de éste estudio) en los que no se cuenta con unas reglas de negocio que provean a los desarrolladores de requerimientos para el desarrollo de una aplicación. Para esto, el presente capítulo pretende dar una base para el planteamiento de la aplicación a realizar en éste proyecto.

La razón de la realización de un prototipo de sistema de comercio electrónico se fundamenta en la cantidad de facetas claves que pueden encontrarse en este tipo de sistemas (en el siguiente capítulo serán definidas, por ahora se hablara de la arquitectura básica del sistema).

Para este caso, se definirá una arquitectura genérica para evitar lidiar con las reglas específicas de un negocio, que impidan la adaptación de dicha arquitectura a diferentes sistemas comerciales. Una vez definida la aplicación, y teniendo en mente las facetas que conforman al sitio, será posible plantear los **módulos** que posiblemente serán requeridos por los desarrolladores a la hora de realizar la aplicación.

* El objetivo de la **ingeniería inversa** es obtener información técnica a partir de un producto accesible al público, con el fin de determinar de qué está hecho, qué lo hace funcionar y cómo fue fabricado.

3.1 ARQUITECTURA DE UN SISTEMA DE COMERCIO ELECTRÓNICO

En cualquier sistema computacional, existe una arquitectura básica determinada por tres características comunes a todos los sistemas. Los procesos, los roles y los componentes.

Al hablar de procesos dentro de la arquitectura de un sistema comercial, Zuñiga plantea una serie de interrogantes que permiten definir la forma de los procesos del sistema en una aplicación de comercio electrónico:²⁵

¿Cómo es que el usuario realiza la transición?

En muchos casos, el usuario visualiza un botón que dice "oprima aquí para comprar" u otro botón que agrega los artículos a un carrito de compras para realizar la compra más tarde. La transición para la transacción se lleva a cabo en cualquier momento, ya sea en el "compre ahora" o en el de pagar la cuenta y salir (del Inglés, checkout), para el caso del carrito de compras.

¿Cómo se verifica la información?

Dependiendo de la tecnología que se tenga como referencia, será necesario verificar en el sistema de transacción que la información de compra como sería el precio, la identificación del artículo, etc., no fue modificada mientras era enviada a través de la red. Como el Web usa un protocolo sin estado, el sistema comercial cuenta con que el Web debiera manejar su propio estado. Si ese estado es manipulado por el cliente, el servidor deberá de ser capaz de

25 ZUÑIGA, Víctor. Comercio Electrónico: Estado actual, Perspectivas y servicios. Universidad de las Américas, Puebla 1999. p. 41

asegurar que el estado no ha sido modificado durante el tránsito.

¿Cómo es que la información concuerda?

Algunos sistemas de comercio electrónico incluyen un proceso denominado actualización de inventario en tiempo real para asegurarles a los clientes que el producto que necesitan se encuentra en existencia. Sin embargo, si el sistema muestra que tal artículo se encuentra en existencia, por cuánto tiempo será válida esta afirmación. Por otra parte, si el cliente coloca tal artículo en el carrito de compras para comprarlo más tarde, ¿el sistema garantizará que el artículo aún se encuentra disponible para ser comprado? ¿Qué hay si el cliente nunca regresa al sitio para realizar la compra del artículo que había dejado en el carrito? Las respuestas a estas preguntas ayudarán en la toma de decisiones para el diseño del sistema de comercio electrónico.

Tanto los compradores como los trabajadores de un sitio de comercio electrónico deben tener claramente definidos sus roles en el sistema. Esencialmente en el caso de la administración de un sitio de comercio electrónico, la diversidad de roles puede ayudar a definir las reglas por las cuales el sistema se regirá.

El diferenciar los roles permite identificar claramente los procesos. No es lo mismo hablar de un cliente que revisa un catalogo de productos a un cliente que realiza el “checkout” en el sitio de comercio electrónico. De la misma manera los procesos relacionados a ambos roles son diferentes.

Víctor Zuñiga²⁶ plantea una idea acerca de los roles existentes en una aplicación comercial. Zuñiga propone como ejemplo de los roles a cada uno de los compradores y vendedores que interactúan con el sistema de comercio electrónico. Sin embargo, la definición de los roles no termina en el simple enunciado de éstas.

Es imposible determinar solo por la nomenclatura de un rol las acciones que este puede desarrollar, por lo que es casi un requerimiento el detallar este tipo de situaciones en cada uno de los roles.

Como ejemplo, Zuñiga propone hablar del rol general entendido como Cliente. Conociendo las acciones que éste puede desarrollar en una aplicación de comercio electrónico (casos de uso), el cliente puede adoptar un rol de comprador o *buyer* si negocia los arreglos para un pago, pero también puede tomar el rol de lo que Zuñiga llama un *specifier* cuando el cliente simplemente se dedica a seleccionar productos del catalogo para ser comprados.

De igual manera, los roles de los clientes pueden ser determinados no por sus acciones sino por su relación con el sistema de comercio o el vendedor. Un comprador casual ("walk-in-customer") no mantiene una relación con el sistema más allá de la generada por una compra que probablemente no se repetirá.

Un cliente de membresía, es aquel que tiene una relación más estrecha con el sistema de comercio y que frecuentemente visita el sistema para realizar sus compras. Lo usual es que estos tipos de usuarios, por su fidelidad reciban descuentos, promociones o regalos por parte del vendedor.

26 ZUÑIGA, Víctor. Comercio Electrónico: Estado actual, Perspectivas y servicios. Universidad de las Américas, Puebla 1999.

Este tipo de roles deben ser especificados en la generalización de una aplicación de comercio electrónico, pero de nada sirven si los procesos resultantes de la interacción entre los diferentes roles y el sistema no se especifican.

Finalmente teniendo en cuenta los usuarios y los procesos, es posible especificar los componentes que conformarán la aplicación. Una vez analizados los puntos anteriores acerca de roles y procesos, finalmente se puede plantear la idea del tipo de componentes que el sistema debe llevar (facetas).

Los componentes básicos para un sistema de comercio electrónico están determinados por los clientes, los vendedores, el sistema de transacción y el medio de pago.

El cliente usualmente es un usuario conectado a Internet a través de un ISP (Internet Service Provider) que realiza la navegación por el sitio de comercio y realiza las compras.

El cliente debe ser identificado en el sistema y para realizar una compra, deberá ingresar a éste con el fin de poder mantener un registro de las ventas y los perfiles de los clientes que maneje el sistema. Además de las opciones básicas de compra de productos (agregar o eliminar productos del “carrito de compras”), el usuario podrá modificar su información personal que no será limitada a la provista durante el registro en el sistema.

Los vendedores deben tener la opción administrativa del sistema; poder agregar, eliminar, revisar y editar (CRUD*) productos, despachar pedidos, generar reportes y administrar usuarios, serán las tareas básicas presentes para este componente.

* Del ingles Create, Read, Update and Delete

El sistema de transacción se encuentra estrechamente ligado con el componente de clientes y de vendedores. El sistema de transacción aparece en el momento en el que el usuario decide adquirir un producto. Dicho producto debe ser descontado de la base de datos y ser almacenado como un pedido y esta situación debe aparecer en el registro de pedidos que debe poder ser exportado para su posterior manipulación.

Este proyecto planteara la arquitectura llegando únicamente hasta el sistema de transacción dejando de lado el medio de pago pues no pertenece al alcance de este proyecto establecer un medio de pago con alguna entidad bancaria.

A partir de los componentes básicos establecidos se desprende una serie de facetas que dan funcionalidad al software. Estas facetas se detallan en el siguiente capítulo.

4. APLICACIÓN DEL MODELO DE EVALUACIÓN DE FRAMEWORKS

4.1 FACETAS DEL PROTOTIPO

Para realizar la selección de la aplicación web a utilizar en este estudio, se tuvieron en cuenta ciertos requerimientos que permitieran a la aplicación acogerse a las características deseadas para el momento de la evaluación de los frameworks que fueron explicadas en el capítulo anterior. Como se dijo antes, la necesidad de generalizar el estudio, llevó a la utilización de facetas para establecer un tipo de aplicación sobre la cual realizar las pruebas.

Para realizar la evaluación es necesario agrupar un número considerable de facetas que conformen una aplicación. Nuevamente se recuerda que son los módulos y no las facetas las que serán parámetros a evaluar en el framework. Sin embargo la definición de facetas es necesaria para el planteamiento de los módulos a evaluar.

Para este caso, la definición de facetas se realizó de forma general, y sin entrar en detalle, pues solo servirán para establecer el tipo de módulos necesarios para construir una aplicación web (estos serán detallados más adelante).

4.1.1 Especificación de facetas para un sistema de comercio electrónico

- **Información Confidencial.** Esta faceta hace referencia a todo lo relacionado con el control de usuarios. Dentro de la lista de requisitos para que exista esta faceta, de una forma medianamente notable en una aplicación web, está el poder distinguir entre un usuario que haya iniciado sesión en el sistema, de un simple visitante. Además esta faceta debe poder proveer a los diferentes tipos de usuarios una serie de recursos a los que pueda acceder dependiendo de su rol en el sistema.

- **Internacionalización.** La internacionalización hace referencia a la posibilidad de presentar la información de forma localizada. Esto permite a un usuario de una región específica encontrar datos e información de forma entendible para él, es decir, en su idioma, bajo reglas de su región (como fechas, unidades, etc. ...) e incluso cuando se hable de sistemas e-business, la moneda..

- **RIA.** AJAX es una serie de tecnologías o una arquitectura, proveniente de la integración entre JavaScript, XML y la a-sincronización entre una aplicación y un servidor de aplicaciones. Cuando fue creado se establecieron unos puntos a tener en cuenta para definir a una tecnología como AJAX.

1. Presentaciones estándar usando XHTML y CSS
2. Vistas dinámicas e interactivas utilizando el Document Object Model o DOM (por sus siglas en ingles)
3. Intercambio y manipulación de datos utilizando XML y XSLT
4. Recolección de datos de forma asíncrona utilizando XMLHttpRequest

5. La utilización de JavaScript para integrar todo.

Este tipo de tecnología permite desarrollar aplicaciones conocidas como RIA o Rich Internet Application, pues desprenden una gran interactividad con el usuario sin limitar la funcionalidad y eficiencia de la aplicación, teniendo en cuenta que se desarrolla en la web.

Se catalogó el uso de AJAX como una faceta porque es una forma de presentación de datos e información que puede o no estar presente en una aplicación y su utilización desprende una serie de requerimientos dependientes de ciertos tipos de tecnologías.

- **Sindicación.** RSS es un formato de datos utilizado en la re-difusión de información proveniente de un sitio web. La utilización de este formato permite la lectura de la información sin la necesidad de un navegador web, para esto se utiliza un lector (que puede o no estar integrado en algunos navegadores). La utilización de estos formatos en una aplicación web, permite a los usuarios suscritos al sitio, informarse de las últimas actualizaciones realizadas.
- **Exportación a formato PDF.** Esta faceta permite al usuario de la aplicación exportar la información presente a un formato portable, así mismo permite la generación de informes o la importación de dichos documentos para ser re-formateados y ser nuevamente presentados al usuario dentro de la aplicación.
- **Listas de Correo.** Las listas de correo permiten la comunicación con una gran cantidad de usuarios de una forma directa. Los mensajes enviados a la lista de correo son re direccionados a cada uno de los usuarios pertenecientes a la lista. El

flujo de mensajes es bidireccional, es decir, los usuarios no solo están limitados a recibir los correos, sino tienen también la posibilidad de enviar sus comentarios a la lista.

- **Uso de Formularios.** El uso de formularios es una faceta muy común en una aplicación web (tanto así que incluso páginas web, no consideradas como aplicaciones web, los tienen) y permiten la recolección de datos realizando peticiones a los usuarios. El uso de formularios es la forma más común de interacción en una aplicación.
- **Buscador Interno.** Esta faceta permite la búsqueda y/o indexación de información en una aplicación web. Su funcionalidad es clara y los requerimientos son básicos. Debe poder realizar las búsquedas dependiendo de los parámetros establecidos por el usuario, así como de la cadena de caracteres introducida.
- **Web Services.** Un servicio web o Web Service, es un protocolo que permite el intercambio de datos entre aplicaciones. El integrar una aplicación ya existente dentro de otra (mash-ups), permite a las aplicaciones proveer servicios integrados, mejorando la experiencia del usuario.

La anterior, es una lista de facetas generalizadas, es decir, no se entró en detalle acerca de todas las sub-facetas que puedan existir en la aplicación, dado que el objetivo de este proyecto no es discernir sobre estas. Por el contrario, teniendo esta generalización de facetas, se pueden plantear una serie de **módulos** que debería tener un framework de programación web para que su objetivo (como framework) verdaderamente sea cumplido, y realmente disminuya el tiempo de

desarrollo y promueva la calidad del software para este caso.

4.2 MÓDULOS DEL FRAMEWORK

Dentro de los módulos del framework que permitirán al desarrollador realizar las facetas de la aplicación, se encuentran todas las clases y funciones propias del framework que facilitan la tarea de programar. Son estos módulos los que se evalúan en el modelo. De momento es necesario definir los módulos que serán evaluados.

A diferencia de lo realizado con las facetas, en la definición de módulos, se establecieron dos niveles para la evaluación. El primer nivel será la generalización de la reunión de varios sub-módulos, que en muchos casos proveen al framework de la opción de realizar una faceta que lleva el mismo nombre; en otros casos, la generalización hará referencia a otras características presentes en el framework pero que, por si solas, no constituyen una herramienta para el desarrollo de facetas, pero que brindan al framework un valor agregado que mejora su utilidad.

4.2.1 Patrón de diseño Modelo Vista Controlador (MVC). El patrón de diseño Modelo Vista Controlador es comúnmente usado en el desarrollo de software para separar las reglas del negocio de la interfaz de usuario y los mecanismos de control²⁷.

²⁷ RAIBLE, Matt. Comparing Web Frameworks: Struts, Spring, MVC, Webwork, Tapestry & JSF. Virtuas. United States. 2005. p. 43

La presencia de este patrón en el framework, a pesar de no poseer una funcionalidad definida, pues por sí sola no genera una aplicación o componente, radica en la facilidad que brinda al programador de realizar el mantenimiento de la aplicación, al igual que de desarrollar el proceso de debugging o depuración del programa.

Para el modelo de evaluación se definieron como sub-módulos de éste cada uno de los componentes que conforman el patrón. Es decir, el modelo, la vista y el controlador serán evaluados por separado.

4.2.2 Manejo de usuarios (UM). No todas las aplicaciones requieren de manipulación de usuarios, pero la existencia de este tipo de módulos facilita al programador trabajar con las sesiones y registros de usuarios en las aplicaciones que así lo requieran²⁸. La autenticación es el proceso mediante el cual se le da a un usuario un cierto tipo de acceso a algún recurso.

Por otro lado la Autorización es la decisión acerca de, si el usuario autenticado tiene permisos sobre algún recurso, ya sean de lectura, escritura o edición o de simple operación. Usualmente el proceso de autorización se lleva a cabo mediante la realización de listas de acceso, donde se definen los recursos y los roles que pueden acceder a estos recursos. Por otro lado la autenticación es quizá el proceso más usado en los sistemas actuales, especialmente los relacionados con la web, donde el usuario, a través de un nombre de usuario y contraseña se registra en el sistema.

Dentro del modulo de Manejo de Usuarios, se evaluarán por separado los sub-módulos de autenticación y autorización (o listas de acceso).

28 ALLEN, Rob. Zend Framework in Action: Early Access Edition. MANNING publications, 2007. p 12

4.2.3 Manejo de Bases de datos (DBM). En cualquier aplicación web el manejo de bases de datos es esencial. Al hablar de una aplicación de comercio electrónico suena imposible concebir dicha aplicación sin usar bases datos y registros de los productos ofrecidos en las mismas. Dentro del framework, debe existir una forma sencilla de lidiar con todo lo relacionado a la manipulación de datos procedentes de una base de datos, desde las consultas hasta las ediciones y posterior almacenamiento de la información así como los medios necesarios para realizar la conexión.

Actualmente, muchos de los frameworks se están dirigiendo hacia la total desaparición del lenguaje SQL dentro del proceso de desarrollo, es decir, todas las consultas y conexiones son realizadas mediante el código propio del framework evitando que el desarrollador tenga que escribir una sola línea de SQL.

Dado que las políticas empresariales tienden a cambiar y nuevos acuerdos y tecnologías son desarrollados al interior de cualquier empresa, el almacenamiento de la información puede estar sujeto a cambios en el entorno, y es posible que el motor de base de datos cambie. Es esencial en las aplicaciones actuales que la información pueda ser accedida sin importar la forma de almacenamiento o la base de datos utilizada. Es por eso que los frameworks deben poseer herramientas que permitan mudar la aplicación de una base de datos a otra, sin necesidad de tener que realizar cambios en el código de la aplicación.

En la evaluación se tendrán en cuenta los siguientes sub-módulos: manipulación de bases de datos y conexiones con diferentes bases de datos.

4.2.4 Manipulación de información (IM). Tal vez este sea el modulo mas importante en cuanto a la funcionalidad que pueda brindar un framework. La

interacción con el usuario usualmente se da a través de formularios, donde el usuario responde a peticiones del sistema ingresando los datos requeridos que luego serán procesados por el controlador (en caso de existir). El uso de formularios está muy extendido en la web y un framework debe proveer herramientas para la creación y validación de los mismos.

Por otro lado, la búsqueda de información en un sitio relativamente grande, se hace esencial para cualquier usuario y el requerimiento de una aplicación de poseer un buscador interno se hace evidente. La integración en un framework de un modulo para la búsqueda de información facilitará también la indexación de datos que permitan una búsqueda más fluida y acorde a la relevancia de los datos.

Ahora bien, en el momento de necesitar interacción entre aplicaciones a través del protocolo HTTP, la forma más común de hacerlo es a través del formato XML. El framework debe poseer formas de procesar este tipo de documentos. Además, el surgimiento de la utilización de nuevas tecnologías para realizar las peticiones de información de forma asíncrona (AJAX) se está convirtiendo en un requerimiento de toda aplicación web.

Dentro de este módulo, se evaluarán por separado, la manipulación de formularios, el uso de AJAX, la integración con formatos XML y la existencia de un modulo para la búsqueda de datos dentro del framework.

4.2.5 Manipulación de Documentos (DM). Dentro de las tendencias actuales en el desarrollo web esta la posibilidad del acceso de la información sin importar el formato, el navegador o la plataforma que se utilice para realizar las peticiones. Es por eso que el framework debe permitir el desarrollo de aplicaciones con información exportable a diferentes formatos.

De la misma manera, cada vez más los usuarios requieren información proveniente de la web, la posibilidad de informarse acerca de los cambios recientes y actualizaciones en la información de una aplicación, se hace latente en las nuevas tendencias web 2.0.

Los frameworks actuales deben permitir la generación tanto de fuentes RSS como de listas de correo y manipulación de las mismas para suplir la necesidad antes mencionada. La manipulación de documentos PDF, listas de correo, y fuentes RSS/ATOM son los sub-módulos que se tendrán en cuenta en la evaluación.

4.2.6 Internacionalización (i18n). Dado que Internet abarca gran parte del mundo, toda aplicación web que sea accesible desde Internet esta propensa a recibir usuarios pertenecientes a otras culturas. Esto significa a que diferentes usuarios, con diferentes idiomas y diferentes estándares deban adaptarse al idioma y formatos planteados en la aplicación.

Permitir que la aplicación sea accedida según los estándares locales del usuario, conlleva a que la información llegue a su destino sin tergiversaciones y que el usuario adquiera realmente la información deseada sin realizar un esfuerzo adicional. La existencia de herramientas para manipular el idioma, así como datos locales, es esencial para poder realizar esta propuesta. En la evaluación se tendrán en cuenta los sub-módulos para la manipulación del idioma y la

manipulación de la “moneda local” (divisas).

4.2.7 Integración con Web Services (WSI). Dado que la funcionalidad del framework evaluado, se probará mediante la realización de un prototipo de sistema de comercio electrónico, se seleccionaron cinco servicios web para ser integrados (no necesariamente todos) con el prototipo. Los servicios web seleccionados como sub-módulos para ser evaluados son: Amazon, Last.FM (audioscrobbler), Del.Icio.Us, eBay y Yahoo!.

4.3 PERSPECTIVA DEL PROGRAMADOR

Finalmente, a pesar de no ser un **módulo**, las generalidades del framework establecen toda la funcionalidad que éste provee a la hora de ser utilizado dejando de lado los detalles técnicos en el desarrollo de una aplicación, centrándose en los detalles prácticos de configuración y utilización.

A la hora de seleccionar un framework, más allá de los demás módulos que éste pueda brindar, la facilidad de configuración del framework juega un papel importante. Un framework complicado de configurar, que requiera de varias horas de lectura para el entendimiento de sus componentes y de la forma de desarrollo con éste es un framework que está destinado a fracasar, especialmente en el campo de la web donde la rapidez es esencial.

Por otro lado, la documentación debe existir y ser clara y práctica. De nada sirve la documentación del código sin una guía que permita a los nuevos desarrolladores aprender a usar el framework.

Dado que este proyecto se centro en el estudio en frameworks de código libre, el nivel de la comunidad también merece ser resaltado, entendiéndolo como la cantidad de usuarios que proveen al framework con retroalimentación, generación de nuevas herramientas y corrección de errores del código.

A esto se le suma la periodicidad con la que nuevas actualizaciones del código son liberadas para su uso y el soporte brindado no solo por parte de la comunidad, sino, si existe, por alguna empresa que respalde el desarrollo del framework.

En este caso, los parámetros a evaluar serán: Configuración del framework, documentación, nivel de la comunidad, curva de aprendizaje, soporte y periodicidad de actualizaciones.

Como se planteo en la explicación del modelo en un principio, aunque estas generalidades no pertenezcan a la lista de módulos de un framework, serán evaluadas de la misma manera dentro del modelo. Sin embargo es necesario que queden claras las diferencias existentes entre los módulos y las generalidades con el fin de que no se juzguen bajo la misma perspectiva y sea más objetivo a la hora de la evaluación de ambos parámetros.

4.4 EVALUACIÓN

Una vez concretados los módulos a ser evaluados, se estableció un criterio de evaluación que permitiera la selección del framework. Como se dijo anteriormente, en el proceso se establecieron dos niveles de evaluación correspondientes a los módulos y sub-módulos del framework. Cada módulo contiene una serie de sub-módulos; estos serán los que se evalúen y la puntuación de éstos define el valor

obtenido de aquel que los contenga.

También, los valores reales de los módulos y sub-módulos deben ser especificados y deben estar acorde con lo que se considere relevante para la aplicación. Para esto, dentro de cada módulo, el puntaje obtenido por cada uno de sus sub-módulos, será multiplicado por un porcentaje que refleje su importancia en la evaluación.

De la misma manera, una vez obtenido el puntaje del módulo, este será multiplicado a su vez por un porcentaje que refleje su importancia para el framework. La sumatoria de los porcentajes de cada uno de los módulos, al igual que los sub-módulos, no puede superar en ninguna medida el 100%.

Ahora bien, la puntuación en este caso se estableció siguiendo un rango de satisfacción o aceptabilidad en cuanto a la funcionalidad de un módulo o sub-módulo dentro del framework. Por efectos prácticos y facilidad del entendimiento del modelo se evitó trabajar este rango como un porcentaje de cero a cien (0 a 100) y se estableció como máximo valor el numero dos (2). De esta manera la inexistencia del módulo o sub-módulo en el framework obtendrá como valor un cero (0) y la existencia total que satisfaga los criterios de evaluación obtendrá como máximo un dos (2); a su vez, el nivel de satisfacción mínimo corresponde al 50% que en el modelo de evaluación equivale a uno (1).

Para determinar el puntaje que cada sub-módulo merecía tener, en la práctica se recurrió a la documentación de las APIs y los manuales de referencia de los diferentes frameworks. Esto con el fin de determinar la existencia o inexistencia de una herramienta dentro del framework que permitiera dar solución a los requerimientos de una faceta. Si la herramienta se encontraba, el módulo obtenía el mayor puntaje (2), en caso contrario se procedía a una nueva revisión.

En la nueva revisión se verificaba en la documentación si es posible dar solución a un requerimiento exclusivamente con el código provisto por el framework, ya fuera combinando el uso de varias clases, o recurriendo a métodos existentes en alguna clase específica cuyo propósito no fuera esencialmente el de dar solución al tipo de requerimiento que se estudia.

En caso de encontrar en el framework una situación similar a la previamente descrita, el módulo para el framework obtenía el puntaje mínimo de satisfacción (1) y si esta situación no se presentaba en absoluto, se daba por entendido que dicho módulo era imposible de utilizarse exclusivamente con las herramientas o código provisto por el framework, por lo que éste obtendría el puntaje de cero (0).

4.4.1 Pesos de los módulos. Siguiendo lo establecido en un principio en el modelo de evaluación se establecieron los pesos que se consideraron pertinentes para evaluar a cada uno de los frameworks. Se recuerda que la decisión de la importancia de cada módulo está sujeta a los criterios del evaluador y que es mediante la matriz de Moody's para toma de decisiones que se normalizaron los pesos. Las siguientes tablas presentan la matriz de Moody's utilizada para normalizar los pesos de los módulos y sub-módulos establecidos en un principio.

Tabla 7. Calculo de pesos de los módulos

Variables a comparar								
M1	Patron de diseño							
M2	Manejo de usuarios							
M3	Manejo de bases de datos							
M4	Manipulacion de documentos							
M5	internacionalizacion							
M6	manipulacion de información							
M7	Integración con web services							
M8	Generalidades							

	M1	M2	M3	M4	M5	M6	M7	M8		Puntaje	Peso	
M1	1	0	0	0	0	0	2	0	=	3	0,047	
M2	2	1	0	2	2	0	1	2	=	10	0,156	
M3	2	2	1	2	2	1	2	2	=	14	0,219	
M4	2	0	0	1	2	0	2	2	=	9	0,141	
M5	2	0	0	0	1	0	2	1	=	6	0,094	
M6	2	2	1	2	2	1	2	2	=	14	0,219	
M7	0	1	0	0	0	0	1	0	=	2	0,031	
M8	2	0	0	0	1	0	2	1	=	6	0,094	
Total										=	64	1,000

Fuente: Autor del proyecto

Tabla 8. Calculo de pesos para el sub-módulo MVC

Variables a comparar				
M1	Modelo			
M2	Vista			
M3	Controlador			

	M1	M2	M3		Puntaje	Pesos
M1	1	1	1	=	3	0,33
M2	1	1	1	=	3	0,33
M3	1	1	1	=	3	0,33
					9	1

Fuente: Autor del proyecto

Tabla 9. Calculo de pesos para el sub-módulo UM

Variables a comparar	
M1	Listas de acceso
M2	Sesiones

	M1	M2		Puntaje	Pesos
M1	1	0	=	1	0,25
M2	2	1	=	3	0,75
				4	1

Fuente: Autor del proyecto

Tabla 10. Calculo de pesos para el sub-módulo DBM

Variables a comparar	
M1	Multiples DB
M2	Manipulacion de DB

	M1	M2		Puntaje	Pesos
M1	1	1	=	2	0,5
M2	1	1	=	2	0,5
				4	1

Fuente: Autor del proyecto

Tabla 11. Calculo de pesos para el sub-módulo DM

Variables a comparar	
M1	PDF
M2	Listas de Correo
M3	Fuentes RSS

	M1	M2	M3		Puntaje	Pesos
M1	1	1	1	=	3	0,33
M2	1	1	1	=	3	0,33
M3	1	1	1	=	3	0,33
				9	1	

Fuente: Autor del proyecto

Tabla 12. Calculo de pesos para el sub-módulo i18n

Variables a comparar	
M1	Idioma
M2	Moneda

	M1	M2		Puntaje	Pesos
M1	1	0	=	1	0,25
M2	2	1	=	3	0,75
				<hr/>	<hr/>
				4	1

Fuente: Autor del proyecto

Tabla 13. Calculo de pesos para el sub-módulo IM

Variables a comparar	
M1	Soporte XML
M2	Soporte AJAX
M3	Manipulacion de Formularios
M4	Buscador

	M1	M2	M3	M4		Puntaje	Peso
M1	1	1	0	0	=	2	0,125
M2	1	1	0	0	=	2	0,125
M3	2	2	1	1	=	6	0,375
M4	2	2	1	1	=	6	0,375
						<hr/>	<hr/>
						= 16	1,000

Fuente: Autor del proyecto

Tabla 14. Calculo de pesos para el sub-módulo WSI

Variables a comparar					
M1	Amazon				
M2	Audioscrobbler				
M3	Del.icio.us				
M4	eBay				
M5	Yahoo!				

	M1	M2	M3	M4	M5		Puntaje	Peso
M1	1	1	1	1	1	=	5	0,200
M2	1	1	1	1	1	=	5	0,200
M3	1	1	1	1	1	=	5	0,200
M4	1	1	1	1	1	=	5	0,200
M5	1	1	1	1	1	=	5	0,200
						=	25	1,000

Fuente: Autor del proyecto

Tabla 15. Calculo de pesos para las generalidades

Variables a comparar						
M1	Configuración					
M2	Documentación					
M3	Nivel de la comunidad					
M4	Curva de aprendizaje					
M5	Soporte					
M6	Periodicidad de actualizaciones					

	M1	M2	M3	M4	M5	M6		Puntaje	Peso
M1	1	0	0	0	1	0	=	2	0,056
M2	2	1	2	2	2	2	=	11	0,306
M3	2	0	1	1	2	0	=	6	0,167
M4	2	0	1	1	2	0	=	6	0,167
M5	1	0	0	0	1	0	=	2	0,056
M6	2	0	2	2	2	1	=	9	0,250
							=	36	1,000

Fuente: Autor del proyecto

Teniendo ya los pesos es posible proceder a realizar la evaluación de los frameworks. Para esto, como sugiere el modelo, se realiza la evaluación a través de la observación. En este caso la documentación de cada framework, así como la

comunidad proveyeron a esta evaluación del conocimiento necesario para tener un criterio a la hora de evaluar. Para esto se recurrió al uso del IRC* (Internet Relay Chat) en el servidor Freenode, bajo los canales #zftalk, #symfony, #cakephp, para tener un contacto con la comunidad de cada framework y realizar las consultas necesarias.

La **Tabla 16** presenta los resultados de la evaluación realizada a los cuatro frameworks escogidos para este proyecto. Como se muestra en la tabla, cada sub-módulo fue puntuado por separado bajo las condiciones anteriormente expuestas y el módulo recibe su puntaje como un promedio de todos los puntajes de sus sub-módulos.

* Es un protocolo de comunicación en tiempo real basado en texto, que permite debates en grupo o entre dos personas y que está clasificado dentro de la mensajería instantánea. (wikipedia).

Tabla 16. Calificación de frameworks

MODULOS	Modulo	sub modulo	ZEND	SYMFONY	KUMBIA	CAKEPHP
1. Patrón de Diseño	0,047		2	2	2	2
1.1 Modelo		0,333	2	2	2	2
1.2 Vista		0,333	2	2	2	2
1.3 Controlador		0,333	2	2	2	2
2. Manejo de Usuarios	0,156		2	1,5	0	2
2.1 Listas de Acceso		0,250	2	0	0	2
2.2 Manejo de Sesiones		0,750	2	2	0	2
3. Manejo de Bases de Datos	0,219		2	2	2	1,5
3.2 Múltiples Bases de Datos		0,500	2	2	2	2
3.3 Manipulación de Bases de Datos		0,500	2	2	2	1
4. Manipulación de Documentos	0,141		1,67	0,33	0	1,33
4.1 Manejo de documentos PDF		0,333	2	0	0	1
4.2 Manejo de Listas de Correo		0,333	1	0	0	1
4.3 Manejo de fuentes RSS o ATOM		0,333	2	1	0	2
5. Internacionalización	0,094		1,75	0,5	0	0
5.1 Manipulación del Idioma		0,250	1	2	0	0
5.2 Manipulación de Monedas Locales		0,750	2	0	0	0
6. Manipulación de Información	0,219		1,38	1,13	0,75	1
6.1 Soporte de XML		0,125	2	1	0	0
6.2 Soporte de Ajax		0,125	0	2	0	2
6.3 Manipulación de Formularios		0,375	1	2	2	2
6.4 Buscador		0,375	2	0	0	0
7. Integración con Web Services	0,031		2	0	0	0
7.1 Amazon		0,200	2	0	0	0
7.2 Last.FM		0,200	2	0	0	0
7.3 Del.icio.us		0,200	2	0	0	0
7.4 eBay		0,200	2	0	0	0
7.5 Yahoo!		0,200	2	0	0	0
8. Generalidades	0,094		1,78	1,36	1,25	1,89
8.1 Configuración		0,056	1	2	1	1
8.2 Documentación		0,306	2	2	2	2
8.3 Nivel de la Comunidad		0,167	2	1	0	2
8.4 Curva de Aprendizaje		0,167	1	1	2	2
8.5 Soporte		0,056	2	1	0	1
8.6 Periodicidad de actualizaciones		0,250	2	1	1	2
TOTAL	1		1,61	1,11	0,7	1,14

Fuente: Autor del proyecto

5. DESARROLLO DEL PROTOTIPO

Una vez teniendo el resultado arrojado por la evaluación es hora de iniciar con el desarrollo del prototipo de la aplicación. En el momento de evaluación y desarrollo del prototipo, el Zend Framework se encuentra en su versión estable 1.5.

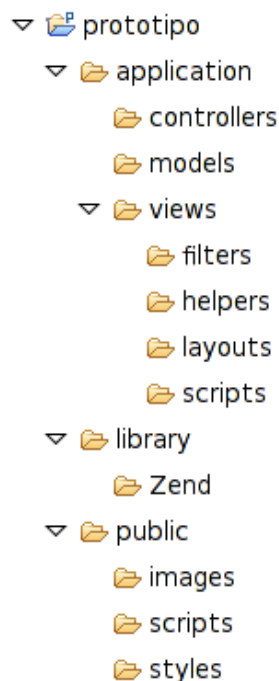
Nuevamente hay que recordad las facetas planteadas en un principio pues serán las que se deben desarrollar con el fin de obtener una aplicación de comercio electrónico mínimamente estable y cumpliendo con la arquitectura genérica planteada en este proyecto.

Para recordar, las facetas son las siguientes: Manejo de usuarios (administración de usuarios), Internacionalización, uso de AJAX (asincronización), uso de Fuentes RSS o ATOM, exportación a PDF, listas de correo, uso y manejo de formularios, buscador interno y web services.

Teniendo esto en mente, se trabajara en el desarrollo individual de cada una de las facetas utilizando, en lo posible, únicamente código provisto por el framework.

Antes de iniciar con el desarrollo del prototipo, la versión estable del framework se debe descargar y configurar correctamente para su uso. De acuerdo con la documentación se propone una estructura de directorios para implementar el patrón de diseño MVC. La **Figura 5** presenta la estructura básica de directorios recomendada para el uso de este framework.

Figura 5. Estructura de directorios propuesta para el Zend Framework.



Fuente: Autor del proyecto

El directorio *Public* contiene toda la estructura de archivos públicos accesibles por el cliente directamente y será el ROOT de la aplicación por lo que en un servidor web el URL a visitar para ver la aplicación sería “http://localhost/prototipo/public/”. El resto de archivos propios de la aplicación se encuentran fuera del ROOT del sistema por lo que no son accesibles directamente por el usuario, proveyendo de esta manera una estructura segura.

Usualmente en una configuración para producción se crearía un host virtual en el servidor web cuyo *DocumentRoot* apuntaría a la carpeta *Public* de la estructura de directorios del Zend Framework.

Todo lo relacionado con scripts en Javascript, imágenes y estilos en CSS se almacenan por separado dentro del directorio *public*. Por otro lado los archivos descargados del Zend Framework son colocados bajo el directorio *library*.

5.1 EL ARCHIVO BOOTSTRAP

La clase `Zend_Controller`, encargada de despachar y correr la aplicación cuando se utiliza el patrón MVC está configurada para utilizar URLs limpias y cortas. Para aprovechar esta funcionalidad, todas las peticiones realizadas a la aplicación deben ser enrutadas y deben pasar a través de un único archivo localizado en el directorio público de la aplicación (`index.php`). Es este archivo al que se le denomina Bootstrap.

Para usar este archivo, que inicializa el `FrontController` de la aplicación, el servidor debe estar configurado correctamente y debe establecer ciertas reglas de enrutamiento, haciendo que cualquier petición no especificada sea direccionada directamente al archivo Bootstrap. Esto se consigue añadiendo en el directorio `public` un archivo `.htaccess` (si el servidor acepta este tipo de archivos para modificar la configuración). A continuación se presenta la configuración establecida en el ROOT del prototipo.

prototipo/public/.htaccess

RewriteEngine on

RewriteRule !\.(js|ico|gif|jpg|png|css)\$ index.php

La instrucción RewriteRule puede ser interpretada como “para todo archivo que no se encuentre con las siguientes extensiones, diréccionelo al index.php”.

Una vez establecidas las reglas de enrutamiento, el archivo Bootstrap puede ser configurado correctamente para inicializar la aplicación. Esto se puede realizar con unas cuantas líneas de código, aunque es posible definir más parámetros que den aún más flexibilidad a la aplicación, las siguientes líneas deben ser suficientes para tener la aplicación funcional.

prototipo/public/inex.php

```
set_include_path('.'
    . PATH_SEPARATOR . ROOT_DIR.'/library/'
    . PATH_SEPARATOR . ROOT_DIR.'/application/models'
    . PATH_SEPARATOR . get_include_path()
);

require_once 'Zend/Loader.php';
Zend_Loader::registerAutoload();

$config = new Zend_Config_Ini(ROOT_DIR.'/application/config.ini',
    'general');
$registry = Zend_Registry::getInstance();
$registry->set('config', $config);

Zend_Layout::startMvc(array('layoutPath' =>
    ROOT_DIR.'/application/views/layouts'));

$dbase = Zend_Db::factory($config->db->adapter,
    $config->db->config->toArray());
```

```
Zend_Db_Table::setDefaultAdapter($database);  
$registry->set('database', $database);  
  
$frontController = Zend_Controller_Front::getInstance();  
$frontController->addControllerDirectory  
    (ROOT_DIR.'/application/controllers');  
  
$frontController->dispatch();
```

Viendo el código presente en el Bootstrap, se puede apreciar que mas allá de instanciar unas clases y especificar rutas para la configuración de la aplicación, de momento no existe nada funcional o visible para un usuario. Esto es porque este archivo simplemente inicializa el uso del Controlador y de momento ninguna Vista (parte del patrón MVC) ha sido renderizada.

La instrucción `set_include_path('...')`; es la parte primordial de la configuración y es donde se establece el uso del framework como tal. Dado que el framework debe usarse instanciando clases, es necesario especificar donde se encuentran estas clases para poder usarlas.

Luego de eso, se presenta la clase `Zend_Loader` que permite instanciar cualquier clase presente en el framework sin necesidad de realizar una carga previa de la misma en la aplicación. Después de esto se presentan unas líneas de configuración del patrón MVC y la configuración de conectividad a la base de datos.

Finalmente la clase `Zend_Controller_Front` es instanciada, configurada y ejecutada a través del método `dispatch()`. Con lo que la aplicación es inicializada.

5.2 IMPLEMENTACIÓN DEL MVC

5.2.1 El controlador y la clase Zend_Controller_Action. Para trabajar con el patrón MVC, es necesario crear controladores encargados de procesar las peticiones de los usuarios que ingresen al sistema. El primer aplicativo a realizar es el del Administrador de la aplicación. Para esto se debe crear la clase encargada de manipular todos los procesos relacionados con el usuario Administrador.

Los casos de uso para un administrador están definidos por los procesos CRUD (create, read, update and delete) de cualquier tipo de información. La clase encargada de la administración del sitio debe extender la funcionalidad del framework y debe especificar entre sus métodos cada uno de los procesos CRUD.

/prototipo/application/controllers/AdminController.php

```
class AdminController extends Zend_Controller_Action
{
    function indexAction()
    {
    }
    function addAction()
    {
    }
    function editAction()
    {
    }
    function deleteAction()
    {
    }
}
```

Una vez configuradas las cuatro acciones a realizar en el sistema de administración del sitio, es posible visualizar las acciones en el navegador visitando las URL como lo especifica la **Tabla 17**.

Teniendo en mente la forma de trabajar de la clase `Zend_Controller_Action`, es posible trabajar cada una de las acciones por separado como un método de la clase extendida de `Zend_Controller_Action`.

Tabla 17. Enrutamiento de las acciones en la clase `AdminController`

URL	Acción
<code>http://localhost/prototipo/admin/index</code>	<code>AdminController::indexAction()</code>
<code>http://localhost/prototipo/admin/add</code>	<code>AdminController::addAction()</code>
<code>http://localhost/prototipo/admin/edit</code>	<code>AdminController::editAction()</code>
<code>http://localhost/prototipo/admin/delete</code>	<code>AdminController::deleteAction()</code>

Fuente: Autor del proyecto

5.2.2 El modelo y la clase `Zend_Db_Table`. El extraer información de una base de datos es esencial, para este propósito Zend Framework provee la clase `Zend_Db_Table` que permite ser extendida para crear una clase que represente a una tabla en la base de datos. Esta clase automáticamente extrae la información de los registros y retorna un arreglo de objetos con los datos de la tabla.

/prototipo/application/models/Product.php

```
<?php
```

```
class Product extends Zend_Db_Table
```

```
{
```

```
        protected $_name = 'product';  
    }
```

Con tan solo extender la clase `Zend_Db_Table` en una clase que lleve el nombre de la tabla y finalmente como uno de sus parámetros establecer el nombre real de la tabla a la que se recurrirá con el modelo, es posible manipular la información de la base de datos.

Teniendo esta clase es fácil recurrir a ella para realizar todas las operaciones o *querys* en la base de datos. Por ejemplo para realizar una búsqueda de un producto por su número de identificación o ID se puede (existen otros métodos) realizar el *query* de la siguiente manera:

```
<?php  
$productos = new Product();  
$seleccion = $productos->select();  
$seleccion->where('id = ' . 10);  
$resultado = $productos->fetchAll($seleccion);
```

La variable `$resultado` termina siendo un arreglo de objetos de la clase `Product` con el resultado del *query* especificado en el método `fetchAll()`.

La forma para realizar una actualización a los datos de un registro de igual manera se realiza sin utilizar una sola línea de código SQL lo que a la hora de realizar una mudanza del motor de base de datos evita el tener que re-escribir las sentencias de conectividad, búsqueda y actualización de los registros.

```
$productos = new Product();  
$row = $productos->fetchRow('id=' . 10);
```

```
$row->nombre = "nombre del producto";  
$row->descripcion = "descripción del producto";  
$row->save();
```

En el código anterior, se actualiza la información en el registro del producto con ID igual a 10 y se establece como nombre del producto la cadena de caracteres *“nombre del producto”* y la descripción con *“descripción del producto”*.

5.2.3 La vista y la clase Zend_View. Por defecto la clase Zend_View viene configurada y lista para usarse al utilizar el patrón MVC. Para separar la interfaz de usuario en una aplicación Web, donde usualmente solo existe código en HTML, el Zend Framework utiliza la extensión .phtml para este fin. Colocando los archivos .phtml en el directorio scripts dentro del directorio views en la aplicación se puede tener acceso a ellos sin ninguna complicación.

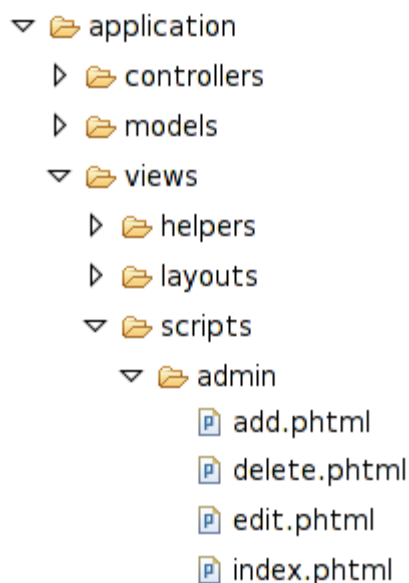
El controlador llama automáticamente a la vista dependiendo del nombre del controlador (esto es el nombre de la clase, por ejemplo la clase AdminController significa que el controlador tiene de nombre admin) y la acción especificada en la URL. Recordando la **Tabla 17** donde se presentaba la dirección de acceso de cada una de las acciones de la clase AdminController, se puede ver como todas tienen el mismo patrón de direccionamiento.

Http://localhost/prototipo/ es la dirección del ROOT de la aplicación. */admin/* representa al controlador que será llamado, en este caso el AdminController, y finalmente se presenta la acción que será realizada, como */add/*.

Una vez entendido esto, realizar las vistas de una aplicación se vuelve un trabajo bastante intuitivo, donde solo es necesario crear un directorio con el nombre del

controlador y dentro de este establecer todos los archivos con extensión .phtml que representan las vistas de cada una de las acciones. La **Figura 6** presenta un ejemplo de la especificación de las vistas para la clase AdminController.

Figura 6. Ejemplo de organización de las vistas



Fuente: Autor del proyecto

Ahora bien, cuando se requiere enviar información procesada en el controlador a alguna de las vistas es posible utilizar la clase `Zend_View` para este propósito.

```
$this->view->Titulo = "Prototipo de aplicación";
```

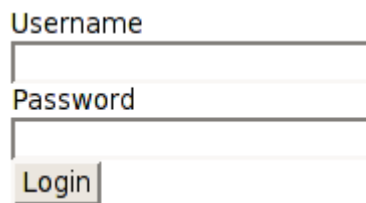
Utilizando el código anterior, es posible pasarle a la vista cualquier tipo de variable, la cual podrá ser accedida directamente desde la vista utilizando el siguiente código:


```
<?php echo $this->escape($this->Titulo);?>
```

5.3 MANIPULACIÓN DE USUARIOS Y LA CLASE ZEND_AUTH

La manipulación de los usuarios se realiza a través de las clases Zend_Auth y Zend_Session. Lo usual es que a través de un formulario el usuario ingrese al sistema y se identifique como un usuario registrado de la aplicación con derecho a usar ciertos recursos (como se muestra en la **figura 7**).

Figura 7. Formulario de ingreso al sistema

El formulario de ingreso al sistema contiene dos campos de texto y un botón. El primer campo está etiquetado como 'Username' y el segundo como 'Password'. Debajo de estos campos se encuentra un botón rectangular con el texto 'Login'.

Fuente: Autor del proyecto

Un controlador que manipule este caso de uso se crea de la misma manera que se explicó en la sección anterior. Sin embargo, las acciones en este caso (los métodos) deberían variar, pues esta vez no se realizarán procesos CRUD sino una simple autenticación en el sistema. Para esto simplemente se define una acción llamada loginAction() y será ésta la que sea llamada como tipo de acción una vez el formulario sea enviado (ósea, cuando el usuario oprima el botón “login”).

El siguiente código presenta las instrucciones necesarias para realizar la autenticación utilizando las clases del Zend Framework.

```

$username = $this->_request->getPost('username');
$password = $this->_request->getPost('password');

$db = Zend_Registry::get('database');
$authAdapter = new Zend_Auth_Adapter_DbTable($db);
$authAdapter->setTableName('users');
$authAdapter->setIdentityColumn('username');
$authAdapter->setCredentialColumn('password');

//ingresar los valores del usuario a la clase Zend_Auth
$authAdapter->setIdentity($username);
$authAdapter->setCredential($password);

//realizar la autenticación
$authentication = Zend_Auth::getInstance();
$result = $authentication->authenticate($authAdapter);

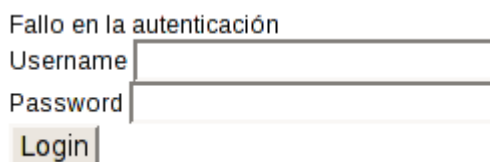
//verifica el resultado de la autenticación
if($result->isValid()) {
    $data = $authAdapter->getResultRowObject(null, 'password');
    $authentication->getStorage()->write($data);
    $this->_redirect('/');
}
else {
    $this->view->message = "Fallo en la autenticación";
}

```

El código anterior realiza una operación de autenticación estableciendo en una tabla de la base de datos los parámetros que serán comparados y verificados. Esto se realiza a través los métodos *setIdentityColumn()* y *setCredentialColumn()*.

Una vez especificados los parámetros a comparar se ingresan los valores obtenidos por el formulario utilizando los métodos *setIdentity()* y *setCredential()*. Finalmente una instancia de la clase *Zend_Auth* realiza la autenticación y devuelve un resultado que es evaluado. Si el resultado es válido el usuario es re direccionado a la página principal y su información como usuario es almacenada en la sesión del sistema. En caso contrario el mensaje “*fallo en la autenticación*” aparecerá en pantalla.

Figura 8. Fallo en la autenticación



Fallo en la autenticación

Username

Password

Login

Fuente: Autor del proyecto

Ahora bien, para realizar la finalización de la sesión de un usuario, basta con utilizar el método *clearIdentity()* de la clase *Zend_Auth*. Este método no recibe ningún parámetro pues simplemente revisa en la sesión de la aplicación la información del usuario y borra dicha información.

5.4 FORMULARIOS Y LA CLASE ZEND_FORM

A partir de la versión 1.5 del Zend Framework, se implementa la clase `Zend_Form`. Esta clase permite crear formularios a partir de una clase extendida, por lo que el formulario se vuelve reutilizable simplemente instanciando la clase que lo crea y enviando el resultado a la Vista del patrón de diseño. El siguiente código presenta un ejemplo de cómo se puede generar un formulario a partir de la clase `Zend_Form`.

```
<?php
class AlbumForm extends Zend_Form
{
    public function __construct($options = null)
    {
        parent::__construct($options);

        $this->setName('album');

        $id = new Zend_Form_Element_Hidden('id');

        $albumname = new Zend_Form_Element_Text('nombre');
        $albumname->setLabel('Nombre del Album')
        ->setRequired(true)
        ->addFilter('StripTags')
        ->addFilter('StringTrim')
        ->addValidator('NotEmpty');

        $artist = new Zend_Form_Element_Text('artista');
```

```

$artist->setLabel('Artista')
->addFilter('StripTags')
->addFilter('StringTrim')
->addValidator('NotEmpty');

$price = new Zend_Form_Element_Text('precio');
$price->setLabel('Precio')
->addFilter('StripTags')
->addFilter('StringTrim')
->addValidator('NotEmpty');

$submit = new Zend_Form_Element_Submit('submit');
$submit->setAttrib('id', 'submitbutton');
$this->addElements(array($id, $category, $albumname, $artist,
$price, $new_price, $description, $submit ));

}

```

Con el código anterior se genera un formulario únicamente con campos de texto para los parámetros: nombre, artista y precio. Adicional a esto, un campo oculto es generado para el parámetro ID y finalmente el botón para enviar el formulario se genera de igual forma utilizando una clase extendida de Zend_Form. El método addElements() recibe un arreglo con todos los campos que contendrá el formulario, por lo que ese arreglo puede ser dinámico y generar así formularios extendidos con base a otro ya existente.

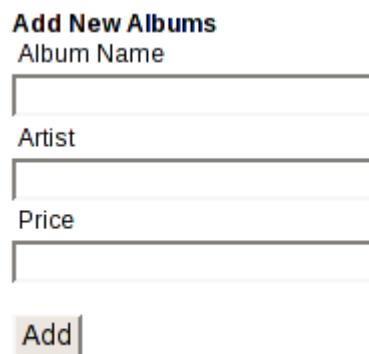
Ahora, para utilizar el formulario en la aplicación basta con instanciar el formulario y pasar dicha instancia a la vista.

```
$form = new AlbumForm();  
$this->view->form = $form;
```

Una vez el formulario es pasado a la vista, éste puede renderizarse simplemente realizando una impresión en pantalla de la variable form.

```
<?php echo $this->form; ?>
```

Figura 9. Ejemplo de formulario generado con la clase Zend_Form



The image shows a web form titled "Add New Albums". It contains three text input fields with labels "Album Name", "Artist", and "Price" above them. Below the input fields is a button labeled "Add".

Fuente: Autor del proyecto

Durante la creación de la clase AlbumForm, para cada uno de sus parámetros se establecieron ciertas reglas y filtros que permiten la manipulación y validación de los campos en el formulario. Si a algún parámetro se le adiciona la regla *setRequired()* y el formulario es enviado con ese campo vacío, retornara un error informando de la invalidez del formulario y del campo que hace falta especificar.

Por otro lado, es posible agregar filtros a los campos para evitar caracteres indeseados y errores en las validaciones en los momentos de búsqueda de información. Esto se hace utilizando el método *addFilter()*.

5.5 BUSCADOR Y LA CLASE ZEND_SEARCH_LUCENE

Zend_Search_Lucene es una clase implementada enteramente en PHP 5, que permite desarrollar un sistema de búsqueda general sin necesidad de realizar *queries* en bases de datos y por el contrario almacenando un índice del contenido de manera separada. Por esta razón esta clase puede brindar de un método de búsqueda sintáctico, similar al algoritmo de búsqueda desarrollado por Google, a cualquier aplicación desarrollada con el lenguaje PHP.

Antes de indexar la información es necesario crear un directorio que contenga los índices de los documentos que serán indexados. En el desarrollo del prototipo de sistema de comercio electrónico se estableció un directorio exclusivamente para almacenar los índices de cada uno de los productos ofrecidos de manera que fueran fáciles de encontrar para cualquier usuario. El siguiente código presenta la función que crea (en caso de no existir) un directorio de indexación en la ruta especificada y almacena un documento con la información que recibe como parámetros.

```
function addAlbumToSearchIndex($album_id, $cover, $category, $albumname,
    $artist, $price, $newprice, $description)
{
    $doc = new Zend_Search_Lucene_Document();
    $doc->addField(Zend_Search_Lucene_Field::Keyword('albumid', $album_id));
```

```

$doc->addField(Zend_Search_Lucene_Field::UnIndexed('cover', $cover));
$doc->addField(Zend_Search_Lucene_Field::Text('category', $category));
$doc->addField(Zend_Search_Lucene_Field::Text('albumname', $albumname));
$doc->addField(Zend_Search_Lucene_Field::Text('artist', $artist));
$doc->addField(Zend_Search_Lucene_Field::UnIndexed('price', $price));
$doc->addField(Zend_Search_Lucene_Field::UnIndexed(
                                'newprice', $newprice));
$doc->addField(Zend_Search_Lucene_Field::Text(
                                'description', $description));

$newIndex = !is_dir('/ruta/para/almacenar/el/directorio/');
$index = new Zend_Search_Lucene(INDEX, $newIndex);
$index->addDocument($doc);
$index->commit();

```

Al añadir un documento nuevo a través de esa función, se establece un tipo de campo para la indexación en el documento de cada uno de los parámetros recibidos por la función. Existen cinco tipos de indexación para los parámetros y estos son:

- **Keyword.** Estos son campos almacenados e indexados y significa que pueden ser parámetros de búsqueda y pueden ser mostrados como resultados.
- **UnIndexed.** Estos son campos que no pueden ser utilizados como parámetros de búsqueda, es decir si una palabra es almacenada como este tipo, y se hace una búsqueda por esa palabra específica el sistema no retornará ningún resultado, sin embargo si se hace una búsqueda de un documento (bajo otros parámetros) donde esa palabra se encuentra, esa palabra puede ser mostrada como resultado.

- **Binary.** Este tipo de dato funciona para almacenar datos en formato binario como imágenes. Sin embargo estos datos no son indexados y por lo tanto no sirven como parámetros de búsqueda.
- **Text.** Este tipo de dato es el más común y es utilizado para almacenar texto que será indexado, utilizado como parámetro de búsqueda y retornado como un elemento a mostrar en los resultados.
- **UnStored.** A diferencia del tipo **Text**, el tipo **UnStored** no almacena la información para ser mostrada como un resultado de la búsqueda, pero puede ser usada como parámetro de la misma. Funciona bien para grandes cantidades de texto, pues se evita almacenar la información y gastar los recursos de la maquina.

Una vez realizada la indexación de la información es posible hacer búsquedas en los documentos almacenados mediante la función *find()* de la clase `Zend_Search_Lucene`. Esta función acepta como parámetros una cadena de caracteres con el string provisto por el usuario que realiza la búsqueda al igual que ciertos parámetros propios de la clase que especifican que tipo de búsqueda realizar. Para el caso de este proyecto el siguiente código fue utilizado para realizar una búsqueda en el directorio de índices.

```
$newIndex = !is_dir("/ruta/para/almacenar/el/directorio");
$index = new Zend_Search_Lucene(INDEX, $newIndex);

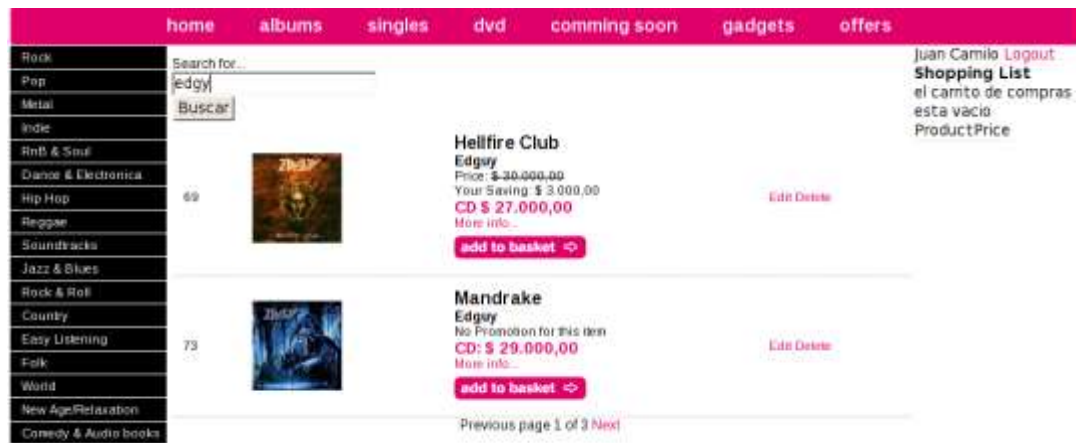
//la busqueda se realiza en esta linea
$hits = $index->find($query.'~');
```

//el resultado de la búsqueda es pasado a la Vista

`$this->view->hits = $hits;`

El parámetro '~' indica a la función *find()* que realice una búsqueda y retorne los índices que se asemejen en su forma de escritura al parámetro provisto por la variable *\$query*. La **Figura 10** muestra la aplicación retornando un resultado de una búsqueda.

Figura 10. Resultado de una búsqueda en el prototipo de sistema e-commerce



Fuente: Autor del proyecto

5.6 WEB SERVICES Y LA CLASE ZEND_SERVICE_AUDIOSCROBBLER

Dado que se desarrolló un sistema de comercio electrónico orientado a la venta de música por Internet, el servicio web de audioscrobblor provee a la aplicación de un componente que permite añadir información acerca de los álbumes que se presentan como productos sin necesidad de almacenar dicha información en la

base de datos.

El uso de este web service, permite a la aplicación adquirir una lista de los discos más reconocidos de un autor, así como una lista de los artistas más similares a un grupo específico. A pesar de lo técnico que puede resultar el uso de este web service, es una herramienta que brinda un valor agregado a una aplicación y permite adquirir información sin muchos esfuerzos de programación y sobre todo, sin necesidad de separarse del contexto del lenguaje PHP (los web services trabajan a través de XML por lo que es necesario implementar código en XML si no se utiliza la ayuda de un framework).

Para utilizar audioscrobbler en la aplicación, se instanció la clase `Zend_Service_Audioscrobbler` y se llamo a los métodos `albumGetInfo()`, `artistGetTopAlbums()` y `artistGetRelatedArtists()`. Cada uno retorna un arreglo con la información especificada. A continuación se presenta el código utilizado en el método `detailAction()` del `AlbumController()`, desarrollado en este proyecto.

```
function detailAction()
{
    //recibe el parámetro id del álbum a mostrar y busca el álbum
    //en la base de datos
    $id = (int)$this->_request->getParam('id', 0);
    $albums = new Album();
    $album = $albums->fetchRow('id = '.$id);
    $this->view->album = $album;

    //Uso de Web Services
    $audioScrobbler = new Zend_Service_Audioscrobbler();
    $audioScrobbler->set('artist', $album->artist);
```

```

$audioScrobbler->set('album', $album->albumname);

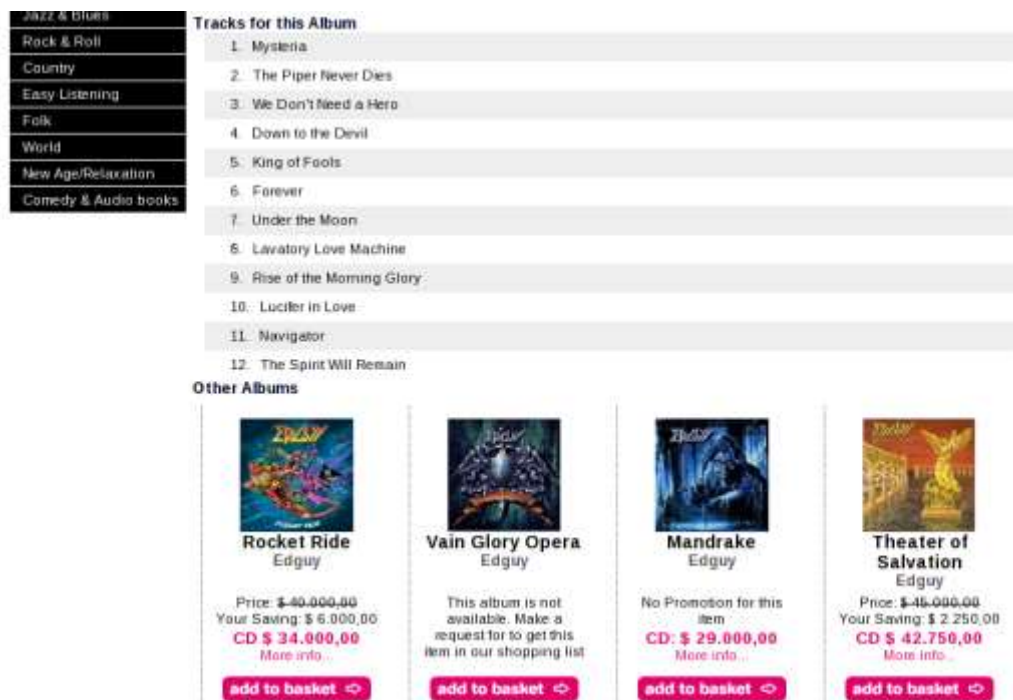
try {
    $albumInfo = $audioScrobbler->albumGetInfo();
    $topAlbums = $audioScrobbler->artistGetTopAlbums();
    $topSimilarArtists = $audioScrobbler
        ->artistGetRelatedArtists();
}
catch(Exception $e)
{
    echo nl2br($e->__toString());
}

$this->view->songs = $albumInfo->tracks->track;
$this->view->topAlbums = $topAlbums;
$this->view->topSimilarArtists = $topSimilarArtists;
}

```

El anterior método recibe un parámetro para realizar una búsqueda en la base de datos de un álbum específico. Una vez encontrado el álbum, se procede a recolectar la información provista por el servicio web. Obtenida toda la información, ésta es pasada a la vista para ser mostrada. La **figura 11** presenta la pantalla de la vista detallada de un álbum, donde se presenta la información provista por el servicio web.

Figura 11. Integración de Audioscrobbler en el prototipo



Fuente: Autor del proyecto

5.7 LISTAS DE CORREOS Y LA CLASE ZEND_MAIL

La clase Zend_Mail provee al framework de la capacidad de realizar envíos de texto plano o en formato HTML de correos electrónicos a través de algún servicio especificado durante el desarrollo de la aplicación. Para el caso de este proyecto donde se desarrolla un prototipo de sistema de comercio electrónico, se optó por utilizar el servidor de correos Sendmail. Esto provee a la aplicación de la opción de enviar correos masivos a los usuarios registrados en la aplicación, creación de listas de correo y/o medios para hacer publicidad sobre las promociones realizadas en el sistema.

Para utilizar la clase Zend_Mail, basta con instanciar la clase extendida que presente la opción de uso del servidor de correos deseado y pasar dicha instancia a la instancia de la clase Zend_Mail. En este proyecto se instanció la clase Zend_Mail_Transport_Sendmail. El código utilizado es el siguiente:

```
$form = new MailForm();
$tr = new Zend_Mail_Transport_Sendmail('webmaster@prototipo.com');
Zend_Mail::setDefaultTransport($tr);
$mail = new Zend_Mail();
$mail->setBodyText($form->getValue('body'));
$mail->setFrom('webmaster@prototipo.com',
              'Tesis Frameworks de programación Web (Webmaster)');

$users = new Users();
$select = $users->select();
$select->where('id > '. 0);
$user = $users->fetchAll($select);
$email = "";
foreach($user as $correo)
{
    $mail->addTo($correo->email);
}
$mail->setSubject($form->getValue('subject'));
$mail->send();
```

El código anterior instancia y configura la clase Zend_Mail, utilizando Zend_Mail_Transport_Sendmail para realizar la configuración. Utilizando el método *setBodyText()* se establece el cuerpo del documento que será enviado

como correo electrónico. Por otro lado el utilizar el método *setFrom()* permite configurar la información del remitente, como el correo electrónico e información adicional que permita identificar el correo.

Para este caso específico, se buscaba enviar el correo a toda la lista de usuarios registrados. Para esto se instanció la clase *Users* y se seleccionaron todos los usuarios existentes en los registros de la clase. Utilizando un ciclo *foreach* se recorrió el arreglo de objetos de usuarios y se obtuvieron los correos de los usuarios que son agregados como direcciones de envío en la clase *Zend_Mail* a través del método *addTo()*.

Finalmente se establece el *asunto* del correo y este es enviado utilizando el método *send()*. En este código, con excepción de los correos de los destinatarios, los demás parámetros fueron extraídos de un formulario, tal y como se puede ver con la inclusión de la clase *MailForm*, una clase extendida de *Zend_Form*.

5.8 MANEJO DE MONEDAS Y LA CLASE ZEND_CURRENCY

Como se dijo antes, el poder ofrecer a los usuarios información de manera localizada permite que llegue a ellos dicha información sin complicaciones y de forma entendible (hay que entender que no todos los usuarios tienen el mismo nivel de experiencia utilizando sistemas de información y mucho menos cuando se trata de sistemas e-business). Por esta razón Zend Framework provee una clase que permite manipular toda información que haga referencia al dinero y permite mostrarla de forma localizada.

Para utilizar la clase `Zend_Currency` basta con instanciar la clase y luego utilizar el método `toCurrency()` de dicha clase, pasando como parámetro el valor que se desea mostrar de forma localizada y , opcionalmente, un parámetro que defina el formato que debe retornar el método para la impresión de dinero.

```
$currency = new Zend_Currency():  
$currency->toCurrency($price, array('format' => 'de_AT'));
```

En el momento del desarrollo de este estudio, la clase `Zend_Currency` se limita a trabajar con el valor entregado como parámetro y a devolverlo como el mismo valor pero escrito de forma localizada, es decir, especificando el tipo de divisa que es, mas no realizando la conversión de la divisa, por lo que se ve una limitación en la clase. Debido a esto parece ser necesario la realización de este tipo de operaciones de forma manual.

6. CONCLUSIONES

Fue posible desarrollar un prototipo de sistema de comercio electrónico en un periodo reducido de tiempo, utilizando el framework seleccionado a través de la metodología propuesta en el estudio.

El modelo de evaluación propuesto determina el framework que más módulos contenga y que satisfagan los requerimientos funcionales del software. Siendo un modelo de agregación es poco sensible a los cambios en los parámetros.

El uso de un patrón de diseño disminuye en gran medida la necesidad de búsqueda de errores en el código. La aplicación, al estar dividida en tres componentes diferentes, permite identificar, cuando existen errores, que componente es el causante del problema por lo que se facilita el mantenimiento y corrección de errores de una aplicación.

El uso de la matriz de Moody's para la toma de decisiones permite disminuir la subjetividad en el cálculo de los pesos correspondientes a cada módulo.

7. RECOMENDACIONES Y TRABAJOS FUTUROS

Para que el modelo de comparación y evaluación planteado en este proyecto tenga mayor validez y se convierta en un método más efectivo se debe contar con un estándar en la documentación de los frameworks.

Se ve necesario, para corroborar la fiabilidad del modelo de evaluación, que los otros frameworks sean utilizados para desarrollar la misma aplicación, de forma que permitan medir el nivel de esfuerzo requerido por un programador con respecto a la opción sugerida por el modelo.

Se recomienda adaptar la metodología propuesta en este trabajo a la evaluación de frameworks basados en paradigmas diferentes a la orientación a objetos con el objetivo de verificar su validez con este tipo de frameworks y en el caso de ser necesario, extender el modelo de manera que acepte otros paradigmas de programación.

Al igual que los frameworks, el modelo presentado en este proyecto debe ser un trabajo en continua evolución y para que esta propuesta mantenga un nivel de aceptación es necesario detallar el proceso de evaluación dentro del modelo de manera que este permita un mayor nivel de detalle y no discrimine los módulos por lo encontrado en la documentación de los frameworks.

BIBLIOGRAFÍA

ALLEN, Rob. Zend Framework in Action: Early Access Edition. MANNING publications, 2007. p 12

ALEXANDER, Christopher, Ishikawa, Silverstein, Jacobson, Fiksdahl-King, y Angel. A Pattern Language. Oxford University Press, New York, 1977. 1216 p.

CORTAZAR, Rebeca. Análisis y aportaciones a la metodología SQ-MET. Departamento de Ingeniería del Software. Universidad de Deusto.

CORTAZAR, Rebeca. SQ-MET: Desarrollo de una Metodología de Validación de Aplicaciones Internet. Departamento de Ingeniería del Software. Universidad de Deusto.

DEGIOVANNINI, Marcio. Comparativa de Frameworks WEB javaHispano [en línea] Javahispano.com [Citado el 24 de Octubre de 2007] Disponible en internet: <http://www.javahispano.org/contenidos/es/comparativa_de_frameworks_web/>

D'SOUZA. Desmond, Francis y Wills. Objects, Components and Frameworks, Addison Wesley, California, October 1998. 745 p.

FILEV, Andrew, Adoptar y aprovechar los procesos ágiles en el desarrollo de software internacional [en línea] MSDN, 26 de Enero de 2007 [Citado el 22 de Octubre de 2007] Disponible en internet: <http://www.microsoft.com/spanish/msdn/articulos/archivo/260107/voices/bb245671.msp#EVB>

GAMMA, Erich, Helm, Johnson y Vlissides. Design Patterns. Elements of Reusable Object-Oriented Software. Addison Wesley, USA, 1995.

GEHTLAND, Justin, Galbraith y Almaer, The Pragmatic Programmers: Pragmatic Ajax a Web 2.0 Primer, Raleigh, North Carolina 2007. 163 p.

GUTIERREZ, Andres Felipe. Kumbia PHP Framework: Porque programar debería ser mas facil. Comunidad Kumbia [en línea] Colombia [Citado el 29 de Agosto de 2007]. Disponible en internet: < <http://www.kumbia.org/LibroDeKumbia.pdf> >

HALCHMI, Z., HOMMEL, K., y AVITAL. O, Electronic Commerce, citado por ZUÑIGA, Víctor. Comercio Electrónico: Estado actual, Perspectivas y servicios. Universidad de las Américas, Puebla 1999.

LADD, Seth y Donald. Expert Spring MVC and the Web Flows, Apress. Estados Unidos. 2006. 423 p.

MOODY, Paul. Toma de decisiones gerenciales. McGraw Hill. Bogotá, 1991.

O'Brien, Duane. PHP frameworks: Getting started with three popular frameworks [en línea] IBM, 9 de Octubre ed 2997 [Citado el 23 de Agosto de 2007] Disponible en internet:<http://www.ibm.com/developerworks/opensource/library/os-php-fwk1/?S_TACT=105AGX44&S>

OLSEN, Dave y Janzen. Blogging for Retailers. Elastic Path Software. Vancouver, Canada. Enero de 2007

OLSINA, Luis Antonio. Metodología Cuantitativa para la Evaluación y Comparación de la Calidad de Sitios Web. Universidad Nacional de La Plata, Facultad de Ciencias Exactas. Argentina. 1999.

O'REILLY, Tim. Qué es Web 2.0: Patrones de Diseño y modelos del negocio para la siguiente generación del software. O'Reilly Media, INC. 2006, 32 p.

PALLET, Dennis, Taking a look at ten different PHP frameworks. PHPit Totally PHP, [en línea] Estados Unidos: The Pallet Group, Marzo 20 de 2006 a las 12:21 CST [citado el 15 de septiembre de 2007]. Disponible en internet: <<http://www.phpit.net/article/ten-different-php-frameworks/>>

RAIBLE, Matt. Comparing Web Frameworks: Struts, Spring, MVC, Webwork, Tapestry & JSF. Virtuas. United States. 2005. 43 p.

RALPH VonSosen, The eCommerce 2.0 Handbook: how to prosper in the new era of online selling, Infotopia Inc. 2007. 65 p.

THOMAS, Dave y Hansson. The Pragmatic Programmers: Agile Web Development with Rails, segunda edición. Raleigh, North Carolina 2007. 715 p.

WILLIAMS, Justin. Rails Solutions: Ruby on Rails Made Easy, Friendsft, California. 2007. 285 p.

Wikipedia The free encyclopedia. Web Application Framework. [en linea] Comunitario [Citado el 27 de Agosto de 2007] Disponible en internet: <http://en.wikipedia.org/wiki/Web_application_framework>.

Wikipedia, The free encyclopedia. Crisis de las Puntocom. [en linea] Comunitario [Citado el 27 de Agosto de 2007] Disponible en internet: <http://es.wikipedia.org/wiki/Crisis_de_las_puntocom>.

ZENDFramework. Programmer's Reference Guide [en linea] Zend Framework, 17 de marzo de 2008 [Citado el 19 de marzo de 2008]. Disponible en Internet: <<http://framework.zend.com/manual/en/zend.search.lucene.html>>

ZUÑIGA, Víctor. Comercio Electrónico: Estado actual, Perspectivas y servicios. Universidad de las Américas, Puebla 1999.