UNIVERSIDAD DEL AZUAY

FACULTAD DE CIENCIAS DE LA ADMINISTRACIÓN

ESCUELA DE INGENIERÍA DE SISTEMAS

"Comparativa de *Frameworks* para el desarrollo de aplicaciones con php"

Trabajo de Monografía previo a la obtención del Título de:

Ingeniero de Sistemas

Autor:
Cindy Belén Espinoza Aguirre

Director:
Ing. Pablo Esquivel

Cuenca, Ecuador
2012

Las opiniones y comentarios vertidos en esta monografía son de responsabilidad absoluta del autor.

Cindy Espinoza

ÍNDICE DE CONTENIDOS

AGRADECIMIENTO

A toda mi familia por su apoyo constante e incondicional en cada momento de mi vida.

A todos los profesores de la escuela de sistemas por compartir sus conocimientos y otorgarnos tiempo valioso de su vida. Gracias, aprovechare al máximo cuanto ustedes me brindaron.

Mis compañeros por estar junto a mí en esas arduas horas de trabajo, y compartir tantos momentos inolvidables.

RESUMEN

Los desarrolladores eligen frameworks al azar, generando dificultades al proyecto, es por esta razón que se pretende comparar los frameworks más populares.

Mi propuesta se basa en escoger la mejor herramienta, mediante la implementación de un caso práctico, evitando que los programadores pierdan tiempo identificando tediosos detalles de bajo nivel, en vez de requerimientos de software.

Para la selección se utilizara modelos o métodos de calidad y en función del *framework* seleccionado generar aplicaciones uniformes que faciliten reutilizar código y promover buenas prácticas de desarrollo, como el uso de patrones y a su vez incremente las ganancias.

ABSTRACT

Developers choose frameworks randomly, generating difficulties for the project, which is why the intention of this study is to compare the most popular frameworks.

My proposal is based on choosing the best tool through a practical case, which will help the programmers avoid wasting time identifying tedious low level details instead of looking for software requirements.

Quality models and methods will be selected and uniform applications will be generated bearing in mind the selected framework. These applications will facilitate the re-utilization of the code and they will promote good development practices such as the use of patterns that will increase profit.

UNIVERSIDAD DEL AZUAY
DPTO. IDIOMAS

Translated by,

Diana Lee Rodas

CAPITULO I

MODELOS DE EVALUACIÓN DE CALIDAD DE SOFTWARE

Introducción

La obtención de un software con calidad implica la utilización de modelos, procedimientos o estándares, claves de la gestión de calidad, para el análisis, diseño, desarrollo y pruebas del software, logrando una mayor confiabilidad y productividad, tanto para la labor de desarrollo, como para el control de la calidad del software.

Por esta razón en el siguiente capítulo se abordarán los conceptos de: modelo de calidad, estructura, tipos, estándares, propiedades, además de generar cuadros resumen de los modelos más utilizados, con los temas previamente definidos.

1.1 Concepto de modelo de calidad de software

El concepto de modelo de calidad de software no es un concepto nuevo entre sus definiciones están: según la norma ISO/IEC 14598-1, un "Modelo de Calidad es un conjunto de características y las relaciones que proveen la base para la especificación de los requisitos de calidad y la evaluación de la calidad".
Según ISO 9000 Se la puede definir como un estándar que posibilita que una organización sea certificada para el diseño, desarrollo, instalación y el mantenimiento de productos y servicios tales como: desarrollo de software, operación y soporte.

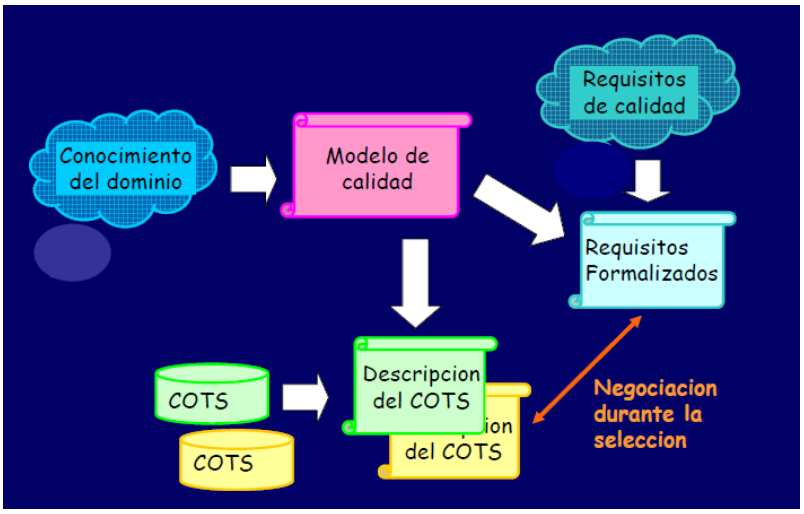
Según la ISO/IEC 25000 Requisitos y Evaluación de Calidad de Productos de Software (SQUARE), un modelo de calidad orienta en el desarrollo del software, con la especificación y evaluación de requisitos de calidad, al establecer criterios y métricas.

Se establecería como concepto principal que los modelos de calidad, son un conjunto de características y sub-características, y cómo estas se relacionan entre sí. Por supuesto, el modelo de calidad a utilizar va a depender del tipo de producto a evaluar, cuya característica fundamental es la mejora continua convirtiéndose en una ventaja competitiva, al brindar especificaciones de qué tipos de requisitos deben implementarse, generando así productos y servicios de alto nivel.

1.1 Características principales

- Definición estructurada de criterios de evaluación.
- Especificación de requisitos.
- Descripción de componentes en un marco común.

- Definición de métricas y prioridades.



Fuente: Sitio UPC Departamento de Ingeniería de Servicios y Sistemas de Información (http://www.essi.upc.edu/~carvallo/Sesion4CS.pdf)

Gráfico: 1 Modelo de calidad uso en la evaluación de componentes

La siguiente inquietud vertida al analizar dicho tema, gira entorno a la pregunta ¿porque usar modelos de calidad?, y la respuesta es muy sencilla el usar modelos de calidad permite identificar especificaciones de software, además permite la toma de decisiones económicas en relación al rendimiento del software así como, evaluar el desarrollo de software usando componentes OTS (Off-The-Shel) o COTS (Commercial Off-the-shelf).

Sin embargo se debe entender que un modelo de calidad no es una metodología que nos resuelva la vida de forma sencilla y clara, puesto que los modelos de calidad nos dicen que hacer, no como hacerlo.

1.2 Estructura de los modelos de calidad de software

Todos los modelos de calidad comparten:

- Un catálogo de factores de calidad (fijo, desechable).
- Diferentes niveles de abstracción (número de capas, jerarquía, grafo).

Algunos autores recomiendan su descripción en forma de un modelo conceptual que explique la forma del modelo, propiedades, elementos medibles, aspectos de formalización o definiciones.

1.3 Tipos de modelos de calidad

Se puede clasificar los modelos de calidad en : modelos fijos a la medida y mixtos.

1.3.1 Modelos fijos:

Existe un catálogo de partida del cual se elige un subset de características de calidad, entre sus ventajas están: reutilizable, comparable, fácil manejo; sin embargo este tipo de modelo es inflexible.

Aproximaciones típicas: Jerarquía multi-nivel (con pocos niveles fijos).

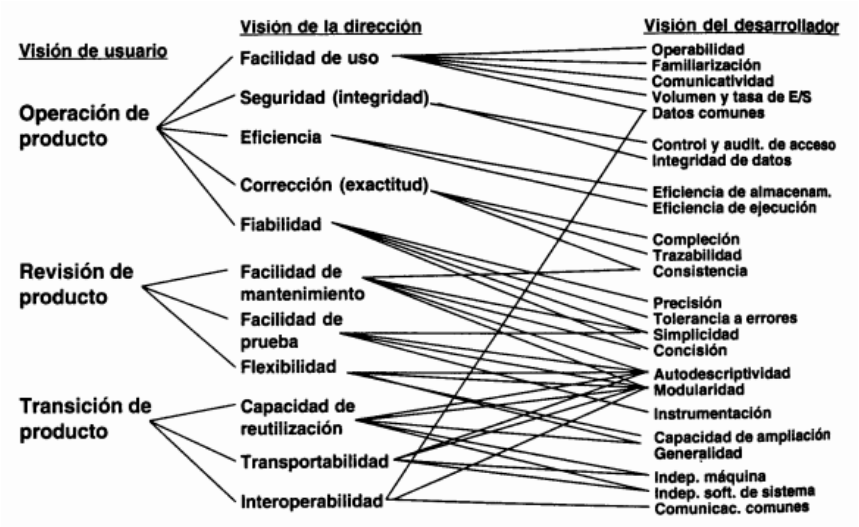
- Factores de alto nivel.
- Criterios más concretos
- Eventualmente métricas para medidas del software.

Modelo de McCall (1977)

11 factores de calidad (en 3 grupos), 23 criterios de calidad (no incluye métricas).

Descompone el concepto de calidad en tres usos o capacidades importantes para un producto software

- **revisión**
- **transición**
- **operación**



Fuente: Gestión de la Calidad (http://www.uhu.es/eyda.marin/apuntes/gesempre/Tema5_1IGE.pdf)

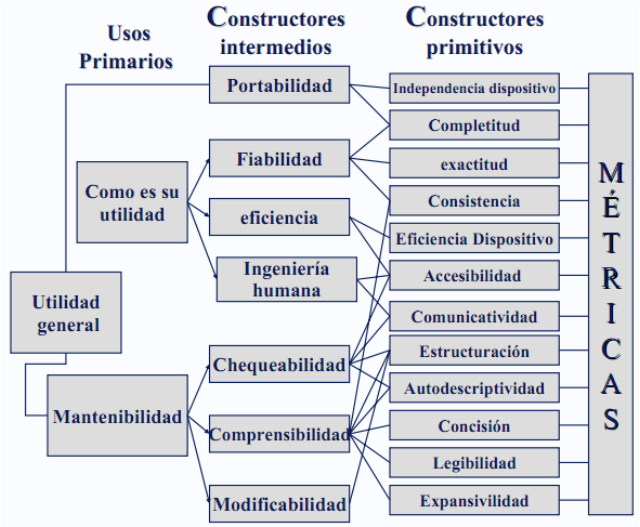
Gráfico: 3 Factores/criterios/métricas (McCall)

Cada factor determinante de la calidad se descompone, a su vez, en una serie de criterios o propiedades que determinan su calidad. Los criterios pueden ser evaluados mediante un conjunto de métricas. Para cada criterio deben fijarse unos valores máximo y mínimo aceptables para cada criterio.

Modelo de Boehm (1978)

6 factores de calidad (en 3 grupos), 12 criterios de calidad (no incluye métricas).Los componentes o constructores del modelo se centran en el producto final.

Se identifican características de calidad desde el punto de vista del usuario.



Fuente: Gestión de la Calidad (http://www.uhu.es/eyda.marin/apuntes/gesempre/Tema5_1IGE.pdf)

Gráfico 4: Características modelo de Boehm

Este modelo introduce características de alto nivel, características de nivel intermedio y características primitivas, cada una de las cuales contribuye al nivel general de calidad.

Criterio	McCall	Boehm	Criterio	McCall	Boehm
correctitud	+	+	confiabilidad	+	+
integridad	+	+	usabilidad	+	+
eficiencia	+	+	mantenim.	+	+
testeabilidad	+		interoperab.	+	
flexibilidad	+	+	reusabilidad	+	+
portabilidad	+	+	claridad		+
modificab.		+	document.		+
entendib.		+	validez		+

Fuente: Sitio Scribd (http://es.scribd.com/doc/75907374/clase6)

Gráfico 5: Comparación de modelo de McCall y Boehm

***Procurement-Oriented Requirements Engineering* (PORE)**

El método PORE [Ncube y Maiden, 1999] es una propuesta basada en plantillas (*templates*) para facilitar la captura de requisitos. PORE tiene seis procesos genéricos.

- Gestión del sistema: Los objetivos de este proceso son la planificación y el control del proceso de obtención.
- Adquisición de requisitos: Este proceso obtiene y valida los requisitos de calidad del usuario y la arquitectura del sistema.
- Selección de suministradores: Establece criterios de selección de los suministradores para evaluarlos, ordenarlos de acuerdo a esos criterios de calidad y los escoge.
- Selección de paquetes/componentes software: Identifica los paquetes (componentes) candidatos, establecer los criterios de selección usando los requisitos del cliente, evalúa los componentes.
- Contrato de producción: Su objetivo es negociar los contratos legales con los suministradores de los componentes software.
- Aceptación del paquete/componente: El objetivo de este proceso es comprobar que el paquete, componente o sistema comercializado cumple con el núcleo original de requisitos esenciales del cliente.

La principal limitación de esta propuesta, que es también común a la mayoría de las que existen al nivel de proceso, es que se centra solamente en el proceso de evaluación de los componentes, pero sin llegar a detallar los atributos concretos que se han de medir, o las métricas a ser utilizadas.

1.3.2 Modelos a la medida

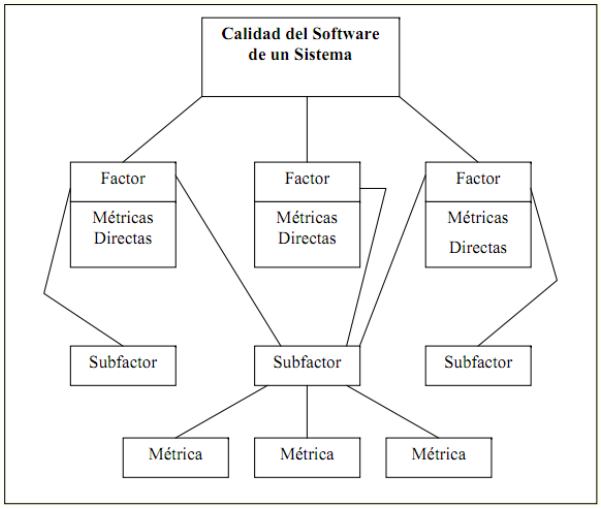
Permite determinar factores de calidad basada en necesidades del contexto, su comportamiento es contrario al caso anterior.

Razonamiento es requerido para determinar el modelo de calidad:

- Identificar objetivos a alcanzar.
- Hacer operativos estos objetivos.

Estándar IEEE 1061 (1998)

- Define factores subfactores y métricas.
- No fija ninguna instancia particular o métrica.
- 1992 y 1998, identifica claramente donde se aplican las métricas, recomienda el uso de técnicas GQM para la identificación de métricas.



Fuente: Sitio Alarcos (http://alarcos.inf-cr.uclm.es/doc/psgc/doc/psgc-4a.pdf)

Gráfico 6: Marco de Trabajo para Métricas de Calidad del Software

Establecimiento de los Requisitos	Establecimiento de los Requisitos.	El propósito de las métricas del software es hacer evaluaciones a través del ciclo de vida del software para comprobar si los requisitos de calidad del software se están cumpliendo, aunque sin que ello elimine la necesidad de un juicio humano en las evaluaciones de software
	Identificar una lista de posibles requisitos de calidad	
	Determinar la lista de requisitos de calidad	
	Cuantificar cada factor de calidad	
Identificación de las Métricas de Calidad del Software	Aplicar el marco de trabajo de las métricas de calidad del software	
	Realizar un análisis costo-beneficio	
	Identificar los costos de la implementación de las métricas	
	Ajustar el conjunto de métricas	
	Adquirir un compromiso con el conjunto de métricas	
	Identificar los beneficios al aplicar las métricas	
Implementación de las Métricas de Calidad del Software	Definición de los procedimientos de la colección de datos	
	Realizar un prototipo del proceso de medición	
	Agrupar los datos y calcular los valores de las métricas	
Análisis de los Resultados de las Métricas del Software	Interpretar los resultados	
	Identificar la calidad del software	
	Hacer predicciones de la calidad del software	
	Garantizar la conformidad con los requisitos	
Validación de las Métricas de Calidad del Software	Propuesta de validación de las métricas	
	Uso de criterios de validación	
	Procedimiento de validación	
	Requisitos adicionales	

Fuente: Autor

1.3.3 Modelos Mixtos

Modelo de alto nivel que puede ser refinado, es un modelo balanceado en relación a los dos anteriores.

Modelo de Gilb

- 4 dimensiones de calidad: trabajabilidad, disponibilidad, adaptabilidad, usabilidad
- 4 atributos de recursos: tiempo, dinero, personas, herramientas.

Determinar una lista de características que definen la calidad de la aplicación. Pueden ser de dos tipos

- Originales
- De los modelos tradicionales

Las características se pueden medir mediante varias sub-características o métricas detalladas. Para cada una de ellas se debe especificar los siguientes conceptos:

- Nombre y definición de la característica
- Escala o unidades de medición
- Recogida de datos o prueba
- El valor previsto
- El valor óptimo
- El valor en el sistema actual
- Comentarios

Este modelo se ha asociado con la filosofía QFD (*Quality Function Deployment*) para la gestión de la calidad industrial.

1.4 Estándares de los modelos de calidad de software

ISO/IEC 9126

ISO/IEC 9126 está clasificado como modelo mixto con un catálogo de partida más elaborado el cual posee 6 características, 27 subcaracterísticas descomponibles en atributos (jerarquía multi-nivel).

Cuyo objetivo es establecer un marco para la evaluación de software, dentro del cual se distingue calidad interna al definir factores durante su desarrollo, externa por que pretende medir la calidad del software teniendo en cuenta su comportamiento dentro de un sistema del cual forme parte y calidad de datos desde el punto de vista del usuario.

El último modelo ofrecido establecido por la ISO/IEC 9126 fue substituido en 2001 por dos estándares relacionados, el ISO/IEC 9126 de calidad del software y el ISO/IEC 14598 de evaluación de productos software, dentro de los factores a considerar están:

Modelo de calidad de uso

- Eficiencia
- Productividad
- Seguridad
- Satisfacción

Modelo de calidad interna y externa

Características	Sub-características
Funcionalidad	Oportunidad,precisión,interoperabilidad,seguridad,funcionalidad,conformidad
Confiabilidad	Madurez, tolerancia a fallos, capacidad de recuperación, fiabilidad de cumplimiento
Facilidad de Uso	Comprensibilidad, capacidad de aprendizaje,operabilidad,cumpliento de usabilidad
Eficiencia	Comportamiento del tiempo, comportamiento de los recursos, eficiencia de cumplimiento
Mantenimiento	Analizabilidad,posibilidad de cambio, estabilidad, capacidad de prueba, cumplimiento con el mantenimiento
Portabilidad	Adaptabilidad, facilidad de instalación, coexistencia, intercambiabilidad, portabilidad, conformidad

Fuente: Autor

A pesar de sus amplias ventajas muchos consideran que el estándar es ambiguo en algunos puntos al definir subcaracterísticas vs atributos, además posee una estructura jerarquía multi-nivel y solapamiento.

Es por esta razón que se ha implementado el modelo SQUARE (*Software Product Quality Requirements and Evaluation*) considerado una revisión de ISO/IEC 9126-1:2001, y conserva las mismas características de calidad de software cuya diferencia principal es considerar la evaluación desde una perspectiva de producto

Calidad del software (interna y externa)							
Funcionalidad	Seguridad	Interoperabilidad	Fiabilidad	Usabilidad	Eficiencia	Mantenibilidad	Portabilidad
Adecuación	Adherencia a normas	Adherencia a normas	Madurez	Comprensibilidad	Timpo de respuesta	Capacidad de análisis	Adaptabilidad
Exactitud			Tolerancia a fallos	Capacidad de aprendizaje	Utilización de recursos	Capacidad a cambios	Capacidad a instalación
Adherencia a normas			Recuperabilidad	Operabilidad	Adherencia a normas	Estabilidad	Capacidad a coexistencia
			Adherencia a normas	Atractivo		Capacidad a testing	Adherencia a normas
				Adherencia a normas		Adherencia a normas	

Fuente: Sitio Alarcos (http://alarcos.inf-cr.uclm.es/doc/cmsi/trabajos/Joaquin%20Ruiz%20Expo.pdf)

Gráfica 8: Modelo de calidad interna y externa y externa para la versión SQUARE

Calidad de uso			
Usabilidad	Contexto	Riesgo	Adaptabilidad
Efectividad	Coincidencia de usuario	Riesgos públicos	Aprendizaje
Productividad	Coincidencia de tarea	Riesgos comerciales	Universalidad
Satisfacción	Coincidencia de entorno	Seguridad	Accesibilidad
Adherencia a normas			

Fuente: Sitio Alarcos (http://alarcos.inf-cr.uclm.es/doc/cmsi/trabajos/Joaquin%20Ruiz%20Expo.pdf)

Gráfica 9: Calidad de uso

Square	ISO/IEC 9126-1	Características
Adecuación funcional	Funcionalidad	El nuevo nombre es mas preciso, y no provoca confusiones con otros significados de funcionalidad
	Interoperabilidad	Movido a compatibilidad
	Seguridad	Característica propia de ISO 9126-1
Disponibilidad	Madurez	Disponibilidad es mucho más importante que madurez
Robustez		Subcaracterística de Square
Eficiencia de rendimiento	Eficiencia	Renombrado para no provocar conflictos con otras definiciones
Operabilidad	Usabilidad	Renombrado para no provocar conflictos con otras definiciones
Reconocimiento de adecuación	Comprensibilidad	El nuevo nombre de Square es mucho más preciso
Facilidad de Uso	Operabilidad	Simplemente se ha renombrado
Útil		Nueva subcaracterística de Square
Accesiblidad técnica		Nueva subcaracterística de Square
Seguridad	Seguridad	En Squre es una característica, en la ISO 9126-1 es una subcaracterística
Compatibilidad		No estaba suficientemente declarado en la subcaracterísticas de portabilidad en la ISO 9126-1
Interoperabilidad		En la ISO 9126-1 es una subcaracterística de funcionalidad

Fuente: Sitio Alarcos (http://alarcos.inf-cr.uclm.es/doc/cmsi/trabajos/Joaquin%20Ruiz%20Expo.pdf)

Gráfico 10: Comparación estándar ISO/IEC 9126-1 y SQuare

Actualmente, este estándar está en proceso de revisión, esperándose su aprobación al igual que en la nueva serie de estándares ISO/IEC 25000, puesto que sólo la primera parte, ISO 9126-1, es un estándar aprobado y publicado, siendo el resto de partes de la norma, informes que se encuentran en la fase llamada *Technical Report*(TR).

El objetivo del catálogo de factores de calidad del modelo ISO/IEC 9126-1 es ofrecer un conjunto de factores de calidad que tienen sentido para cualquier dominio de

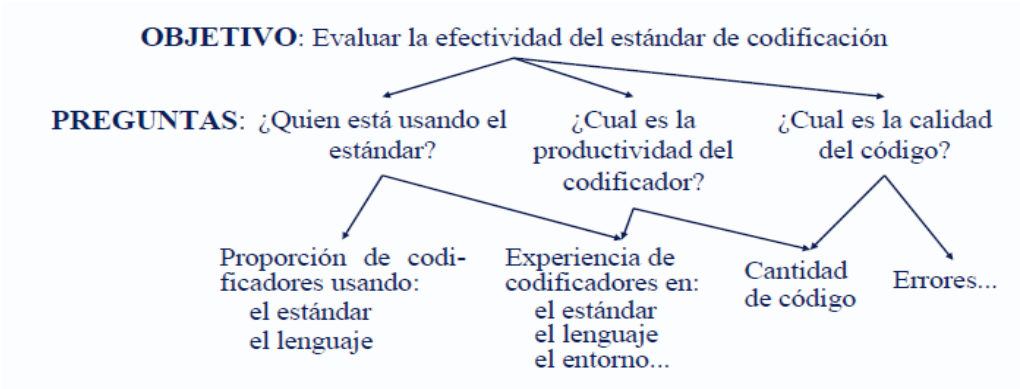
componente software además de brindan la posibilidad de evaluar requisitos no funcionales, que faciliten el estudio de la reutilización de componentes.

Métricas en una forma GQM:

El enfoque GQM basa la mejora en la definición clara de procesos y productos, los cuales proporcionan la estructura para obtener los objetivos cruciales del proyecto.
Consta de tres etapas:

- Lista de los objetivos principales del desarrollo y mantenimiento del proyecto.
- Para cada objetivo obtener las preguntas que deben contestarse para saber si se están cumpliendo los objetivos.
- Decidir qué medir para poder contestar las preguntas de forma adecuada.

Las medidas individuales obtenidas se relacionan para poder ser utilizadas en el contexto del proyecto completo.



Fuente: Gestión de la Calidad (http://www.uhu.es/eyda.marin/apuntes/gesempre/Tema5_1IGE.pdf)

Gráfico 11: Modelo CQM

Off-The-Shelf Option (OTSO)

El método OTSO desarrollado por Kontio [Kontio, 1995] establece un proceso de selección de "paquetes" de software reutilizables, denominados por los autores como OTS (*Off-The-Shelf*) pues incluyen tanto componentes comerciales (COTS) como componentes desarrollados internamente por las propias organizaciones.
El método OTSO facilita la búsqueda, evaluación y selección de software reutilizable. Además, proporciona técnicas específicas para la definición de criterios de evaluación, comparando los beneficios y costos de cada una de las alternativas posibles.

Este método define varias actividades o fases

- definición
- búsqueda
- filtrado

- evaluación
- análisis

Los factores de influencia en la selección de COTS:

- Requisitos de Usuarios
- Arquitectura de la aplicación
- Restricciones y objetivos del proyecto
- Disponibilidad de productos
- Infraestructura de la organización

Aunque el método OTSO resalta que el problema clave en la selección de componentes COTS es la falta de atención a los requisitos de calidad.

Cabe recalcar que OTSO antes que un modelo de calidad es considerado un método para la selección de componentes y entre sus características deben cumplir los criterios de calidad, al facilitar la definición de los requisitos estructurados, cumpliendo así las características de un modelo de calidad. El método OTSO se basa en el uso del proceso de Jerarquía (AHP) para la evaluación de datos, los cuales facilitan la toma de decisiones. La técnica AHP fue desarrollado por Thomas Saaty.

Nombre del elemento	Descripción
Título	Título para cada atributo de evaluación actúa como un identificador único.
Definición	Una definición del atributo de la evaluación.
Razón lógica	Descripción de los fundamentos para el atributo de la evaluación y cómo se relaciona con los criterios de evaluación.
Nivel	La escala o el tipo de descripción utilizada. *Descripción de formato libre *Lista: una lista de funciones, características, funciones, que se produce *Estructurado descripción: No es una plantilla o una lista de control que define lo que debe ser descrita para cada alternativa. *Nominal: Las clases se identifican pero no se ordenan *Ordenado: Las clases se identifican y se ordenan *Intervalo: La escala tiene una interpretación significativa de la distancia entre las entidades, pero sus relaciones no pueden ser calculados, es decir, "no hay un punto cero significativo". *Relación: Las entidades pueden tener relaciones, "el cero es un concepto significativo". *Absoluto: El número de entidades se cuenta
Grupos/Clases	Definición de la unidad de medida o las clases usadas, lo que sea aplicable
Detección de regla	Definición de un posible nivel que se requiere para una alternativa a ser seleccionado
Línea base	Línea de base es el nivel mínimo requerido de funcionalidad y características que la aplicación debe satisfacer cuando sea entregado.
Descripción Cualitativa	Directrices cómo obtener información adicional sobre el atributo de evaluación debe ser documentada.
Fuente	Cómo el valor del atributo de evaluación se puede determinar para cada alternativa
Prioridad	Descripción de la importancia del atributo particular de evaluación es.

Fuente: Autor

Gráfico 11: *Template* OTSO

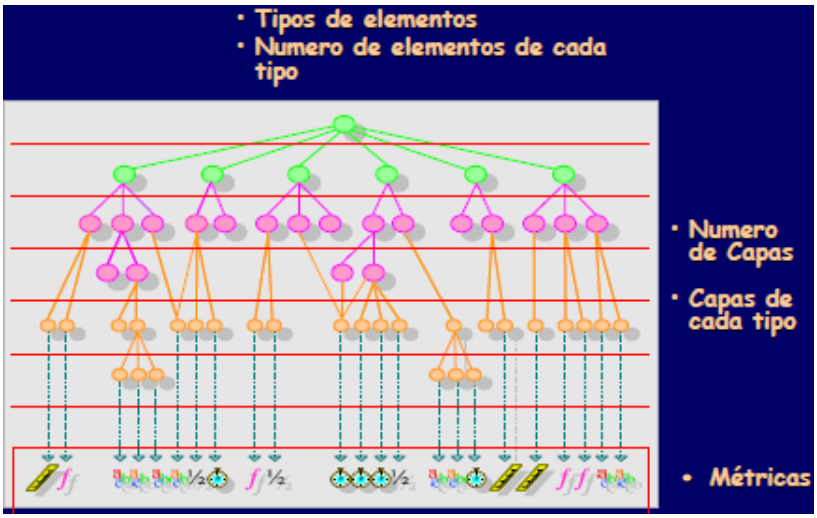
Para el análisis de criterios, en muchos casos debe exigir experiencia y estudios amplios por lo que AHP gracias a su tratamiento jerárquico se adapta a la evaluación de procesos los cálculos son propios del método.

El AHP permite la consolidación de toda la información cualitativa y la información financiera en una clasificación única de las alternativas.

1.5 Propiedades de los modelos de calidad de software

Estructura y elementos constitutivos

- Tipos de elementos
- Número de elementos de cada tipo
- Número de capas
- Capas de cada tipo
- Métricas.

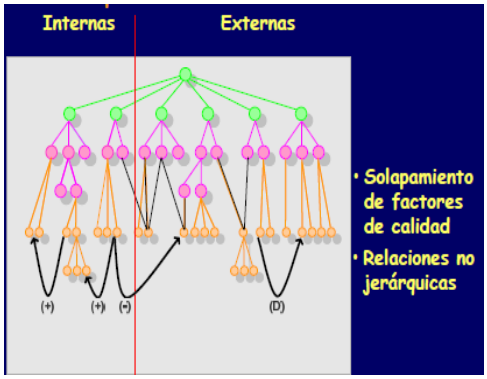


Fuente: Curso de Calidad de Software UDA Marzo-Julio 2006

Gráfico 12: Propiedades modelos de calidad

Factores Internos y Externos.

Algunos modelos hacen distinción de los factores internos y externos. Es decir los internos tienen un efecto sobre los externos, los externos ayudan a estimar el comportamiento de los internos y al hablar de factores externos, nos referimos a características directamente percibidos por los usuarios (al relacionarse con la funcionalidad y usabilidad) mientras que al estimar el comportamiento de los internos, se hace referencia a las características constructivas de los componentes, que son tan solo accesibles y controlables por sus fabricantes, así pues la falta de esta separación puede comprometer el uso de los modelos. Sin embargo no todos los modelos poseen dicha característica.



Quality Factor	Category
Integrity	External Quality Factors (i.e. usability factors)
Reliability	
Usability	
Correctness	
Efficiency	
Interoperability	
Maintainability	Internal Quality Factors
Testability	
Flexibility	
Reusability	
Portability	

Fuente: Curso de Calidad de Software UDA Marzo-Julio 2006

Gráfica 13: Separación de características Gráfica 14: Modelo McCall según Fitzpatrick

Relación entre factores de calidad Algunos factores son útiles para la evaluación de varias características, existen modelos:

- **Con Solapamiento** McCall, Boehm, algunos niveles 9126-1.
- **Sin Solapamiento** FURPS.

Hablamos de solapamiento cuando un factor de calidad forma parte de la descomposición de otros a nivel superior, sin embargo demasiadas dependencias pueden comprometer la utilidad del modelo, dicho factor puede evaluarse con métricas diferentes.

Funciones de las Métricas

Entre las funciones de las métricas están: permitir proveer la base para estimaciones, realizar un seguimiento del avance, determinar la complejidad, ayudarnos a entender cuando hemos alcanzado el nivel de calidad esperado, analizar defectos, validar experimentalmente las mejores prácticas.

Software Quality Factors	Efficiency	Flexibility	Integrity	Interoperability	Maintainability	Portability	Reliability	Survivability	Correctness	Reusability	Verifiability	Usability	Expandability
Efficiency													
Flexibility	X												
Integrity	X												
Interoperability	X	O	X										
Maintainability	X	O											
Portability	X			O									
Reliability	X				O								
Survivability	X	X	X										
Correcteness		O											
Reusability	X		X		O	O	O	X	O				
Verifiability	X				O	O	O		O	O			
Usability	X	O			O		O	O			X		
Expandability	X		X		O			X	O				

X means the factors conflict
O means the factors support one another
A blank space means there is not relationship

Fuente: Sitio UPC Departamento de Ingeniería de Servicios y Sistemas de Información (http://www.essi.upc.edu/~carvallo/Sesion2CS.pdf)

Gráfica 15: Interrelaciones entre factores de calidad IEEE 1061-1992.

Clasificación de los modelos de calidad de software

Existen propuestas en relación a

Estructura

- Número de capas
- Número de relaciones entre capas

Alternativa de construcción

- Fijo, medida o mixto
- A la medida subdividido en forma directa o indirecta

Aplicación

- General
- Específico para un producto

Cuadros comparativos de modelos de calidad de software según la clasificación realizada:

Modelo	Tipo de Modelo	Separación elementos internos y externos	Relación entre los elementos	Medidas aplicadas para
McCall	Fijo	No	Jerárquico	Criterios
Bohem	Fijo	No	Jerárquico, métricas de dependencia	Características primitivas
FURPS	Fijo	No	Jerárquico	Los atributos de calidad en el menor nivel jerárquico
CQM	Medida	No	Jerárquico	Preguntas
Gilb	Medida	No	Jerárquico	Atributos bajo y de alto nivel
ISO	Medida	No	Jerárquico	Atributos de calidad
IEEE 1061-1998	Medida	No	Jerárquico, conflicto y el apoyo a las relaciones entre los factores	Cualquier elemento en cualquier capa (definibles por el usuario)
Dromey	Medida	Yes	Vínculos entre los componentes y atributos de alto nivel	No incluidos en la propuesta
PORE	Fijo	No	Jerárquico	No incluidos en la propuesta
OTSO	Mixto		Jerárquico	Criterios de evaluación establecidos
SQUID	Medida	Yes	Jerárquico, vínculos entre atributos internos y externos	Atributos de calidad
ADEQUATE	Mixto	No	Jerárquico, relaciones directas, inversas y neutral entre los elementos	Factores clave de la calidad y factores locales definidos

Fuente: Autor

Modelo	Tipo de elementos del modelo	Número de capas	Número de elementos de cada tipo
McCall	Factores, Criterios, Métricas	4 (1 principal subdivisión, 1 capa de factores, 1 criterios, 1 métricas)	Factores 14 (3 principal subdivisión,11 en la segunda capa), 23 criterios, métricas no fijas (usuario las define)
Bohem	Características de alto nivel, características primitivas, métricas	4 (2 capa características de alto nivel 1 características primitivas 1 de métricas)	Características de alto nivel :9,Características primitivas: 15,métricas no fijas (usuario las define)
FURPS	Atributos de calidad, Métricas	3 (2 capas de atributos de calidad y uno de métricas)	Primera capa de atributos de calidad:5, Segunda capa 27,métricas no fijas (usuario las define)
GQM	Metas,preguntas,métricas	3 (metas, preguntas y métricas)	Metas: no fijas (usuario las define)
Gilb	Características de alto nivel, características de bajo nivel, métricas	No fijos (definidos por el usuario)	Atributos de alto nivel, atributos de bajo nivel, métricas cada elemento no fijo (el usuario las define)
ISO	Características, sub-características, atributos de calidad, métricas	2 capas superiores (características y subcaracterísticas), capas de atributos no son fijos son definibles, capas métricas están asociadas a todas las capas de atributos	Características:6,Subcaracterísticas: 27,atributos de calidad, métricas no fijas (usuario las define) pero el estándar y las métricas son proporcionales
IEEE 1061-1998	Factores,Subfactores,Métricas	No fijos (definidos por el usuario)	Factores,subfactores,métricas no fijas (usuario las define)
Dromey	Características de alto nivel, componentes	2 capas superiores (los atributos de alto nivel y la calidad de las propiedades de transporte) no se fija en los niveles más bajos	Atributos de alto nivel, calidad propiedades de transporte, componentes no fijo (el usuario las define)
PORE	Características, criterios product-requirement	6 procesos genéricos	No fijos (definidos por el usuario)
OTSO	Fases,factores,métricas	6 fases (5factores)	Métricas: no fijas (usuario las define)
SQUID	Características de calidad, atributos de calidad, métricas	Las capas superiores no son fijas, son definibles por el usuario. Una capa de atributos de calidad	Características de calidad, atributos de calidad una capa, métricas no fijas (usuario las define)

Fuente: Autor

Conclusión

Finalmente, para concluir la calidad debe ser definida y medida, si la mejora se quiere lograr. Sin embargo, un problema importante en la ingeniería de calidad y de gestión es que el término calidad es ambiguo, de modo que se entiende mal comúnmente. La confusión se puede atribuir a varias razones.
En primer lugar, la calidad no es una sola idea, sino más bien un concepto multidimensional. Las dimensiones de la calidad incluye la entidad de interés, el punto de vista sobre esa entidad, y los atributos de calidad de esa entidad. En segundo lugar, hay niveles de abstracción, cuando se habla de calidad, una de las partes podría estar refiriéndose a ella en un sentido más amplio, mientras que otro podría estar refiriéndose a su significado específico. En tercer lugar, el término calidad es una parte de nuestro lenguaje cotidiano y los usos populares y profesionales pueden ser muy diferentes.

El desarrollo o la selección de productos software de alta calidad son, por este motivo, de gran importancia. La especificación y la evaluación extensiva de la calidad de los productos software es un factor clave para asegurar una calidad adecuada. Esto se puede alcanzar definiendo las características de calidad apropiadas, teniendo en cuenta el propósito y uso del producto software. Es importante que cada característica relevante de calidad del producto software se especifique y se evalué, usando dentro de lo posible métricas que estén validadas o ampliamente aceptadas.

CAPITULO II

MODELO DE CALIDAD PARA FRAMEWORK

Introducción

Actualmente existen grandes cambios en la forma en la que se desarrollan las aplicaciones software. La creciente necesidad de realizar sistemas complejos en cortos períodos de tiempo, a la vez que con menores esfuerzos tanto humanos como económicos, está favoreciendo el desarrollo de aplicaciones con la ayuda de frameworks.

Los principales esfuerzos de la comunidad de software se fundamentan en el análisis funcional de los frameworks, es decir, en las características que ofrecen. Sin embargo, por lo general se han venido obviando muchos de los aspectos de calidad que intervienen a la hora de seleccionar frameworks y ensamblar componentes propios, para así construir aplicaciones que satisfagan los requisitos del cliente.

Es por esta razón que este capítulo abordará, una breve introducción acerca de la definición de framework, exponiendo sus principales características y un modelo que facilita su comprensión dentro del proceso de evaluación de calidad, luego se expondrá las ventajas de usar frameworks, así como los criterios, que pueden ser consideradas para su evaluación. Finalmente se escogerá un método de calidad adaptable a la evaluación de frameworks de desarrollo, para así dar inicio a la comparación de frameworks, y a su vez respaldar las ventajas de usarlos.

2.1 Definición de framework

De forma breve, un framework es una estructura conceptual y tecnológica compuesta por librerías, componentes y clases que facilitan el desarrollo ágil, seguro y escalable de una aplicación. La estructura básica de todos los proyectos web es muy parecida. Esto hace que los programadores deban destinar mucho tiempo en realizar tareas repetitivas que aportan escaso o nulo valor a las aplicaciones y es aquí en donde el uso de un framework puede facilitar el desarrollo.
Además los frameworks proporcionan servicios que soportan un modelo de componentes los mismos que se pueden utilizar tanto en entornos de desarrollo como de producción, los componentes considerados patrones que permiten interactuar entre sí de acuerdo al problema que resuelven y permiten la extensibilidad del framework y su funcionalidad, estos son aplicados a dominios específicos.

En el transcurso del capítulo se analizará la calidad como la totalidad de aspectos y características de un producto o servicio , y la habilidad de satisfacer las necesidades declaradas o implícitas. La calidad es el factor decisivo para la selección de la herramienta que entrara a ser parte de un nuevo sistema, es decir, si dos componentes son candidatos para hacer parte de una aplicación y muestran la misma funcionalidad, el nivel de calidad que implemente cada uno de ellos será el elemento sobre el cuál se base la decisión final.

2.2 Ventajas de usar frameworks

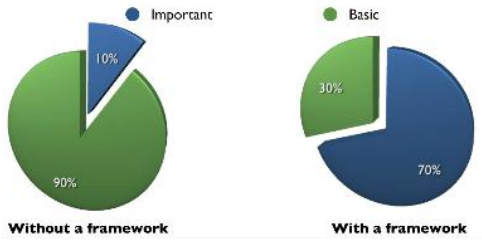
La principal ventaja que ofrece un framework es la reducción del costo en el proceso de desarrollo de aplicaciones software para dominios específicos, y la mejora de la calidad del producto final. Sin embargo otra ventaja es la facilidad para incorporar nuevos desarrolladores a la empresa, dado que el método de trabajo no depende de la empresa en la que se trabaja, sino del framework que se utilice. De esa forma el proceso de adaptación de una nueva persona al equipo de desarrolladores resulta más sencillo y rápido, como en todo, hay desarrolladores pro-framework y anti-frameworks, sin embargo entre las ventajas del uso de frameworks se encuentran:

- Mantenimiento y escalabilidad
- Independencia interfaz gráfica y lógica de negocio
- Menos bug, menos complejidad
- Implementa MVC (Modelo, vista controlador)
- Desarrollo rápido y mejor
- Librerías y códigos ya realizados
- Más beneficios, menos costo
- Extensa documentación e incluso en español
- Independencia de base de datos
- Suelen seguir la mayoría de las mejores prácticas y patrones de diseño web.
- Comunidades online de ayuda y asesoramiento
- Menor cantidad de código
- Menos tiempo, Más productividad

2.3 Desventajas de usar frameworks

A veces surgen limitantes, ya que al utilizar un framework no se puede modificar algo del núcleo, es aquí donde surgen interrogantes como ¿Si yo hubiese programado desde el inicio lo tendría más claro?, otras de las desventajas con mayor peso son:

- El aprendizaje resulta costoso
- Puede implicar la utilización de un gran número de recursos para de esta forma forzar a obtener mejores resultados.



Fuente: Sitio CAMON (http://www.tucamon.es/contenido/ventajas-de-usar-frameworks-en-php)

Gráfica 16: Desventajas de usar frameworks

- Al no utilizar frameworks se tiende a no tener organizado los datos tanto conexión a base de datos, diseño y lógica del negocio.
- Se tiende a crear un concepto equivocado de framework, sin embargo hay que tener presente que es una metodología con herramientas, pero no lo soluciona todo.

2.4 Criterios de evaluación

Después de tener claro que las ventajas de usar frameworks son mayores, y además al ser considerado como un modelo de componentes puesto que sus funcionalidades son fijas, es decir no los podemos cambiar totalmente.

El proceso de evaluación parte con la definición de los criterios basados en factores o características técnicas, no técnicas, funcionales y no funcionales entre los que se podría definir los siguientes:

- Licencias con los vendedores o proveedores.
- Consultas a usuarios del producto.
- Páginas Web o comunidades que respalden el funcionamiento del producto.
- Revisión de la documentación de los productos.
- Revisión de demos.
- Curva de aprendizaje.
- Experimentar sobre los productos.
- Sesiones de demostración.
- Certificaciones de los productos.
- Funcionalidad.
- Interfaz.
- Asegurar la capacidad en función de los requisitos.
- Confiabilidad.
- Seguridad.
- Independencia de base de datos.
- Usabilidad.
- Eficiencia.
- Portabilidad.

Cada uno de los criterios de evaluación debe considerarse bajo las siguientes perspectivas:
Dominio: Representa en términos generales todos los aspectos relacionados a los problemas del programador relacionados a su funcionalidad.
Programa: Este enfoque es el que más varia, ya que muestra en forma más detallada información técnica del framework, estructura de los API'S de información, definición de

la interface de datos, los tipos de parámetros, definiciones de las bases de datos, información, estructura del framework etc.
Situación: Se basa en el conocimiento del diseño y comportamiento, en función de los requerimientos con las capacidades funcionales y no funcionales del framework.

2.5 Seleccionar modelo de calidad para evaluar frameworks

Como se detalló en el capítulo I los modelos de calidad brindan la posibilidad de describir requisitos de calidad en una forma estructurada, es por ello que la elección de un framework es un proceso de ingeniería de software del que depende el producto final. La mala elección de éste conllevara a un desarrollo complejo y fuera del domino.

El modelo de calidad apto para la evaluación de frameworks debe determinar la calidad en función de los requisitos establecidos por el desarrollador, también debe permitir la evaluación de criterios en relación a los DBSC propios del framework así como los factores técnicos, no técnicos, funcionales y no funcionales.

Dentro de los modelos de calidad estudiados, se realizara el análisis de acuerdo al tipo de modelo al que pertenecen para un selección más clara, los modelos fijos como en el caso de McCall, Boeh, FURPS, PORE aportan ventajas comunes como lo son proporcionar una vista fija y comparable que se reutiliza en cada proyecto, debido a que sus factores de calidad siempre son estables. Ahora bien, tiene como inconveniente su poca flexibilidad, lo cual es una característica fundamental dentro del análisis de frameworks dado que los requisitos funcionales y no funcionales van a variar dependiendo del tipo de framework a evaluar, es por esta razón que los modelos no cumplen con el objetivo; de igual forma Boeh identifica características de calidad desde el punto de vista del usuario, mientras que PORE es considerado un estándar para la selección de componentes, y si bien es cierto la calidad se asegura con una correcta selección, la principal limitación de esta propuesta, que es también común a la mayoría de las que existen al nivel de proceso, se centra en la evaluación de los componentes, pero sin llegar a detallar los atributos concretos que se han de medir, o las métricas a ser utilizadas.

Al analizar los modelos de calidad a medida no existe ningún catálogo de factores de partida, y dichos factores deben ser identificados para cada proyecto. La idea que guía la construcción de estos modelos es el partir de la identificación de los objetivos a alcanzar lo cual se consideraría un problema y además no cumple con las expectativas del tema planteado, debido a que cada framework debe ser comparado y esta comparación sería nula, al poseer objetivos propios; la idea es establecer criterios comunes y según dichos criterios realizar las comparaciones. Una de las ventajas es la adaptabilidad pero resulta mayor la cantidad de inconvenientes ya que su construcción es muy complicada comparada con el de los modelos fijos, y como se había analizado la reutilización de modelos de un proyecto a otro es difícil, dado que los factores identificados para un proyecto no tienen por qué ser

adecuados además de entre los modelos que entran en esta clasificación se encuentran IEEE 1061 que no fija ningún factor de calidad.

Pero si una clasificación de los factores de los que debe constar un modelo en un nivel más alto y abstracto de factores, que deben descomponerse en subfactores, y a su vez se descomponen en métricas, en este modelo todos los factores a evaluar están a consideración subjetiva, y se fundamenta más en los requisitos funcionales, además se propone la construcción de modelos de calidad adaptados a cada proyecto, y el objetivo es centrarse en los requisitos no funcionales y orientados al área de desarrollo. El tipo de modelo que se ajusta a nuestras necesidades es el mixto puesto que combina las ventajas de los dos tipos anteriores de modelos. La idea es que, exista un conjunto de factores de calidad más abstractos que sean reutilizados, y que puedan ser editados y operacionalizados para un proyecto particular, además se requiere un modelo o estándar que proponga una jerarquía de factores de calidad clasificados como características, sub-características y atributos, la elección se fundamenta en estos tipos de modelos entre los que tenemos; Gilb el que posee para la evaluación características sub-características o métricas detalladas. Para cada una de ellos se debe especificar conceptos de evaluación, es por esta razón que este modelo se asocia con la filosofía QFD (*Quality Function Deployment*) para la gestión de la calidad industrial, El estándar OTSO facilita la evaluación, búsqueda y selección de software reutilizable, considerando el desarrollo del producto final y el uso del producto es por esta razón que los requisitos de partida se presenten de una manera estructura definiendo criterios de evaluación en función del costo beneficio; aparentemente seria el modelo ideal a pesar de que sus factores de calidad no están definidos pero si existe fases estructuradas sobre las que se definen los requisitos, el problema radica al descomponer los criterios puesto que se usa un proceso llamado AHP, esta técnica, en muchos casos debe exigir experiencia y estudios amplios y los cálculos son propios del método, el estándar ISO/IEC 9126 presenta todas las características requeridas puesto que ofrecer un conjunto de factores de calidad que tienen sentido para cualquier dominio de componente software, y por tanto contiene factores de calidad que serán reusados entre distintos proyectos, además de permitir el análisis de calidad de componentes OTS al permitir el análisis de factores no técnicos relativos al proveedor, al costo , etc., la única desventaja que por así decirla es el solapamiento puesto que en exceso puede afectar el análisis.

Conclusión

Para concluir, este capítulo analiza el concepto de framework, junto con sus ventajas y desventajas, además de realizar un análisis de los modelos de calidad, luego de definir el framework y conceptualizarlo como una aplicación genérica incompleta y configurable sobre la cual podemos construir una aplicación concreta y al saber que se lo considera un modelo de componentes, así como de establecer en breves rasgos los tipos de criterios que utilizan los modelos y los métodos posibles a considerar en función de los requerimientos funcionales, no funcionales técnicos y no técnicos.

Se pasó a la fase de selección del modelo ajustable al concepto de framework establecido y debido a los temas abordados dentro del capítulo I y II , además de seleccionar el estándar ISO IEC 9126 como método de evaluación de calidad.

Es un modelo mixto en el que su principal característica es definir factores de calidad más abstractos que sean reutilizados virtualmente en todos los dominios posibles, facilitando así la comparación de los frameworks, además de no ser un modelo rígido, facilitando su personalización tanto como sea requerido dentro de un proyecto particular y otra de las ventajas sobre los demás modelos analizados, es el permitir el análisis de calidad de componentes OTS, al permitir determinar factores no técnicos relativos al proveedor, al costo , etc., la única desventaja que por así decirla es el solapamiento (factor de calidad que participa en la descomposición jerárquica de otros en niveles superiores) puesto que en exceso puede afectar el análisis, sin embargo se mantendrá un análisis de cada uno de los factores sin caer en la redundancia de su análisis en las subcategorías.

CAPITULO III

FRAMEWORK CAKE

Introducción

El siguiente capítulo realiza una descripción del Framework CakePhp, para posteriormente examinar aspectos como características generales, estructura, convenciones o reglas tanto de la base de datos, modelos, controladores, vistas, parámetros de configuración del core y requerimientos que deberán considerarse antes de efectuar el caso práctico, para finalizar se abordarán temas basados en el manejo de componentes, vistas, controladores estableciendo los pasos para su creación y utilización y se explicara en que consiste el scaffolding además se establecerán las tareas comunes requeridas por los desarrolladores mediante la explicación de las clases.

3.1 Framework CakePhp

CakePHP es un framework, libre, de código abierto sin pérdida de flexibilidad, reutilizable y estructurado desde los nombres de archivo, tablas de bases de datos, las cuales poseen su propia lógica de funcionamiento, además tiene un equipo de desarrolladores y una comunidad activa, la cual ofrece capacitación en línea con costos variables según el nivel requerido.
Su documentación se encuentra estructurada en 6 sitios según el tipo de información, sin embargo 4 de ellos poseen información clara y exacta; entre ellos se encuentran http://www.cakephp.org , sitio oficial el cual cuenta con enlaces a herramientas, videos, descargas y oportunidad de contribuir a la comunidad, http://book.cakephp.org manual en línea, http://bakery.cakephp.org casos de uso y ejemplos de código, http://api.cakephp.org/ se encuentra documentación más completa que explica los detalles internos del funcionamiento del framework.
CakePhp lanza actualizaciones constantes, las cuales se encuentran probadas y son tolerante a fallos y estables.

3.2 Características de CakePhp

Entre sus características se encuentran

- Comunidades activas y manuales en línea.
- Licencia flexible MIT, la cual es libre de modificar, distribuir y publicar el código fuente con la condición de que los avisos de copyright se mantienen intactas; se puede incorporar CakePHP en cualquier aplicación comercial o de código cerrado.
- Compatible con PHP4 y PHP5
- Soporte de aplicación Scaffolding

- Arquitectura Modelo Vista Controlador (MVC)
- URLs y rutas personalizadas
- Validación integrada
- Sintaxis de PHP, con ayudantes (helpers)
- Ayudantes para AJAX, JavaScript, formularios HTML
- Componentes de Email, Cookie, Seguridad, Sesión y Manejo de solicitudes
- Limpieza de datos

3.3 Estructura de los Archivos

Para entender el funcionamiento e implementación de cake se debe entender como está estructurado

- app: Los archivos del usuario son colados
- cake: Core propio del framework.
- vendors: Biblioteca de componentes adicionales.
- plugins
- .htaccess
- index.php
- README

Todo el desarrollo se realiza dentro de la carpeta /app, la cual utiliza una división de carpetas.

Config: mantiene la configuración de pocos archivos de CakePhp, entre los que se encuentran: detalles de base de datos, configurar rutas de acceso hacia los controladores.
Console: consola de la aplicación y sus componentes.
Lib: contiene bibliotecas propias definidas por el programador, evitando se mezclen con las bibliotecas de los proveedores.
Locate: almacena archivos para la internalización.
Model: contiene modelos de la aplicación, es decir las tablas de datos mapeadas.
Plugin: contiene paquetes de plugins.
Tmp: almacena datos temporalmente como datos de sesiones, registros etc.
Vendor: clases o bibliotecas de terceros, útiles para configuración de múltiples aplicaciones y de sistemas complejos.
View: almacena archivos de presentación, elementos, páginas de error, ayudantes y diseños de vistas.
Webroot: configuración de producción almacena hojas de estilos CSS, imágenes y archivos JavaScript.

3.4 Convenciones de la Base de Datos, Modelos, Controladores, y Vistas

La configuración de la Base de Datos en CakePhp requiere cumplir ciertas condiciones

- El nombre de las tablas definidas en plural.
- El campo primario de cada tabla deberá ser definido como id.
- Si existe relación de ∞ a ∞ la clave principal de la tabla débil se forma por las claves principales de las tablas regulares en singular seguidas de _id y en orden alfabético.
- En la relación 1 a ∞ el campo primario de la tabla principal definida con relación 1, forma parte de los campos de la tabla principal con relación ∞, el valor del campo deberá estar definido en singular y seguido de _id.
- Las claves foráneas, llevan el nombre de la tabla de relación en singular seguida por _id.

CakePHP no soporta claves primarias compuestas. Si deseas manipular directamente los datos de la tabla de unión, usa llamadas directas a query's.

Modelos: Los nombres de las clases de modelos están en singular y en formato CamelCase.

Controladores: Los nombres de las clases de los controladores son en plural, con formato CamelCase y Terminan en Controller. El método creado por defecto es índex, cuando se especifica el nombre del controlador pero no de la acción.

Para utilizar métodos que no sean accesibles directamente desde la web, sino solo para uso interno anteponer guión bajo _.

Vistas: Los archivos de plantillas de Vistas (*Views*), deben antes ser definidas en el controlador dentro de una carpeta que agrupe todas las vistas relacionadas y su extensión termina en .ctp.

3.5 Desarrollo con CakePhp

Dentro de esta estructura se determinaran diversos aspectos de CakePHP, temas como los requerimientos, configuración, y estructura de cakephp las cuales faciliten el desarrollo de una aplicación.

3.5.1 Requerimientos

1.- Servidor Web:
2.- Servidor de Base de Datos.
3.- Conocimiento básico de PHP

4.- Por último, se necesita un conocimiento básico del patrón MVC (Patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica del negocio en tres componentes distintos).

3.5.2 Configuración del Core

Este archivo es una colección de definiciones de variables Configure y definiciones de constantes que determinan como ha de comportarse la aplicación.

A pesar de que muy pocas cosas necesitan configuración, para el correcto funcionamiento del CakePhp se debe cambiar lo siguiente:

1.-Editar módulo de seguridad dentro del /app/config/core del framework, línea Security.salt, Security.cipherSeed cambiando sus números originales.

2.-Activar el modo módulo mod_rewrite, del apache en el archivo htpd.conf dentro de la línea #LoadModule rewrite_module modules/mod_rewrite.so.

Clase App

Carga clases adicionales mediante App:import (), asegurando que la clase ha sido cargada sólo una vez apropiadamente, resuelve las rutas de ubicación automáticamente en la gran mayoría de los casos.

Importando librerías del core, y controladores sin embargo luego la clase debe ser inicializada.

3.5.3 Controladores

Un controlador maneja la lógica de la aplicación los cuales se extiende de la clase principal AppController definida en /app/app_controller.php considerada clase estándar al desarrollar Cakephp, dentro de los controladores se definen los métodos o acciones a usar, por lo general interactúan con un solo modelo.

Dentro de la clase controller se deben definir por lo menos una función para cada vista que interactúa con el usuario

Atributos del Controlador

$name: define el nombre del controlador, o nombre del helper.

params: Funciona con los parámetros del controlador disponibles en $this->params cuyo objetivo es obtener acceso a información que ha sido entregada al controlador a través de las operaciones POST o GET

Métodos

Set(): Es una manera rápida de asignar un conjunto de información a la vista, cuya estructura es set ($variable, $valor).

Render (): Es llamado automáticamente al final de cada acción de controlador pedida, llevando a cabo la lógica de la vista (usando los datos proporcionados con el método set ().

Redirect(): El método de control de flujo utilizado con mayor frecuencia, redirige el browser hacia una acción de un controlador establecida.

3.5.4 Modelos

Un modelo es un mapeo de la tabla asociada convirtiéndola en un objeto, se extiende de AppModel entre sus funcionalidades se encuentran:

Find: Facilita las búsquedas dentro del modelo

```
$this->Post->find('list', array('fields'=>'Post.title'));
```

Find (first): Es el tipo find por defecto, y devolverá un solo resultado.

```
$this->Article->find('first', array('conditions' => array('Article.id' => 1)));
```

```
$this->Article->find('count', array('conditions' => array('Article.status' => 'pending')));
```

Find (count): Devuelve un entero con el número de registros.
Find (all): Devuelve un array de resultados, usado para paginar.

```
$this->Article->find('all', array('conditions' => array('Article.status' => 'pending')));
```

Find (list): Devuelve un array indexado, usado en selects del formulario.

```
$this->Article->User->find('list', array('conditions' => array('Article.status !=' => 'pending')));
```

Query(): Se pueden realizar sql personalizados, utilizando el nombre de la tabla en la consulta como clave del array, en vez del nombre del modelo, cake posee recomendaciones para armar sql.

```
$this->Compra ->query("SELECT * FROM compras LIMIT 2;");
```

3.5.5 Componentes

Paquetes de lógica que son compartidos entre los controladores se encuentran dentro del core del proyecto en la carpeta cake/libs.

- Sesiones: Librería cake_session.php.
- Seguridad : Ubicada dentro cake\libs\controller\components como auth.php, sin embargo es un componente que se puede utilizar a nivel de proyecto para personalizar es decir dentro de app/controller/components/Auth.php, para su personalización
- file.php:Componente encargado de la carga de archivos.
- Controller: Componente encargado de la lógica, del que heredan los controladores propios del sitio, contiene scaffolding.php.
- Model: Componente manejo de datos de su validez, interacción y la evolución del flujo.
- View: Contienen las clases de salida y presentación como son elements, helpers, layouts propios de cakephp.

3.5.5.1 Crear componentes

Se crean componentes cuando se requieren cierta acción en diferentes controladores, el cual posee su propia lógica

1. Crear una clase dentro de /app/controllers/components/nombrecomponente.php
2. Añadir al nombre del componente la palabra"*Component*"
3. La clase hereda de Object
4. Definir las funciones propias

```
<?php
class ValorComponent extends Object {
    function doOperation($cant1, $cant2) {
      return $amount1 + $amount2;
    }
?>
```

3.5.5.2 Añadir Componente:

Podemos usarlo en los controladores de la aplicación añadiendo su nombre (excepto la parte "*Component*", de esta manera podremos acceder a una instancia del mismo:

```
var $components = array('Math', 'Session');
```

Los componentes definidos no necesitan declararse dos veces puesto que se incluye los componentes a un controlador.

3.5.6 Vistas

Las vistas o ficheros están en php plano y tienen la extensión .ctp (CakePHP *Template*) por defecto, contienen toda la lógica de representación recibidos del controlador.

Se almacenan en /app/views/, en una carpeta nombrada según el controlador está formada por diferentes de partes.

- ***layouts*** (diseños): ficheros de vista que contienen el código de presentación.
- ***helpers*** (ayudantes): estas clases encapsulan lógica de vista, ayuda a construir formularios, construir funcionalidad AJAX, paginar los datos del modelo.

Para implementar una vista se deben definir los siguientes parámetros $this->Form->create('Cliente'); y echo $this->Form->end(__('Grabar', true)); que nos permite tener la creación del formulario, para mostrar cada uno de los datos de la base o sobre los cuales las acciones se desplegaran, se debe definir una lógica propia de cake de la siguiente manera:

1. $this->Form->input ('nombres'); nombre del campo de la base de datos,
2. Para definir el tipo de campo, como en caso de ser un campo tipo file se define así $form->input ('img_file', array('type' => 'file'));
3. Si se desea enviar parámetros hacia otra vista se debe definir de la siguiente manera: $this->Html->link($cliente['Tarjetacredito']['id'], array('controller' => 'tarjetacreditos', 'action' => 'view', $cliente['Tarjetacredito']['id']));

3.5.7 Scaffolding

Otra de las ventajas ofrecidas es la configuración y fácil desarrollo mediante la aplicación de datos por consola, configurándose de la siguiente manera.

1. Configurar dentro de las variables de entorno del sistema en la variable del Path colocando al final.
    a. El directorio donde se encuentra el php.exe
    b. El directorio hacia la consola de cake cake/console separado por ;

2. Realizada esta configuración, dentro de la consola de Windows en el directorio app, escribimos la instrucción cake bake.

3. Se podrá trabajar desde la consola, comenzando desde el mapeo de las tablas de la base en los llamados modelos, implementar la lógica del sistema dentro del directorio controller y facilitar la interacción con el usuario al editar las vistas.

3.6 Tareas Comunes

Dentro de las tareas comunes se encuentra la validación de campos en la cual la palabra valide nos permite especificar qué criterios debe poseer ese campo utilizado en el formulario de creación ya que cada tabla de la base de datos posee un mapeo o modelo y para utilizar los formularios de interacción con los usuarios se deberá realizar validaciones en cada campo entre los tipos de validación más frecuente se encuentran.

1. notempty(valida que un dato no sea ingresado como nulo), alphaNumeric(Solo letras), comparison(comparaciones bajo condicionantes), range(establecer rangos) en la cual dentro del array se debe definir el nombre del campo y la regla que está relacionada; message es un texto que se desplegara al efectuarse una validación errónea, esta implementación se la realiza dentro del modelo.

```
public $validate = array(
        'campomodel1' => array(
        'notempty' => array(
        'rule' => array('notempty'),
        'message' => 'Ingrese valores un valor no se aceptan datos nulos',
         ),
        'campomodel2' => array(
                'comparison' => array(
              'rule' => array('comparison', '>=', 0),
               'message' => 'Ingrese valores positivos',
        ),
        'campomodel3' => array(
                'range' => array(
                'rule' => array('range',-1,101),
                'message' => 'Valor entre 0 y 100'    ),
        )
```

Configuración de conexión a la base de datos.

1. La configuración se encuentra en el archivo app/config/database.php.default
    a. Copiar dicho archivo
    b. Renombrarlo a database.php
    c. Configurarlo, de acuerdo a los datos de la base que vayamos utilizar, cambiar datos como 'driver' => 'mysql', 'host' => 'localhost', 'login' => 'root', 'password' => 'admin', 'database' => 'tiendavirtual', en este caso utiliza dbo_mssql.php

CakePhp posee varios datasources específicos (enlace entre modelos y la fuente de datos) para según la base de datos cuyo listado se encuentra en cake/libs/model/datasources/dbo/

Manejo de Sesiones

1.-El manejo de Sesiones en este framework se realiza de la siguiente manera:
$this->Session->read ();
2.-Para especificar un valor en especial
$this->Session->read('Auth2.Cliente.id');
3.-El envío de mensajes o avisos al finalizar un proceso en el controlador se especifica con:
$this->Session->setFlash (__ ('Invalid compra', true));

Conclusión

Una vez realizado el estudio del framework CakePhp se determinó el funcionamiento, requerimientos, estructura, configuraciones, componentes, modelos,vista,controladores y los beneficios que estos ofrecen así también se presentó la estructura de las tareas más comunes, scaffolding de esta manera se tendrá bases sólidas para efectuar el caso práctico mediante dicha herramienta.

CAPITULO IV

ONLINE SHOP DESARROLLADO EN CAKEPHP

Introducción

El siguiente capítulo pretende ser una guía en el desarrollo de una aplicación, generada con CakePhp dentro de la cual se abordarán temas como instalación, conexión hacia la base de datos, generación de modelos, mapeo de datos, relaciones entre los modelos, estructura de la lógica de negocio implementación de componentes utilizados, seguridad, manejo de imágenes, generación de vistas utilizas en el caso práctico, para el cual se ha tomado como ejemplo la implementación de una tienda virtual, puesto que representa el intento de trasladar la "operativa" comercial habitual de un comercio tradicional a Internet, debido a que el comercio electrónico no solo es una novedad tecnológica, sino más bien ha revolucionado el funcionamiento de las empresas para finalizar el caso práctico posee casos de uso, y además se mostrar la interfaz de la aplicación, generada en función de la herramienta y de los requerimientos.

4.1 Instalación de CakePhp

- 1.-Para la generación de un proyecto en cake primero se debe descomprimir CakePhp 2.0.5, en una carpeta o directorio accesible vía web .
- 2.-Renombrar la carpeta con el nombre del proyecto a implementar en cake.

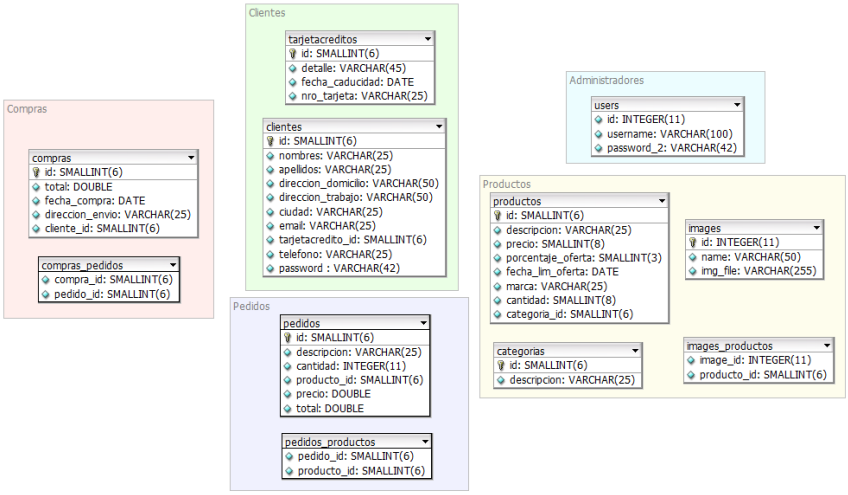
4.2 Conexión hacia la Base de Datos

Editar el archivo de configuración app\config\database.php

```
class DATABASE_CONFIG {
        var $default = array(
                'driver' => 'mysql',
                'persistent' => false,
                'host' => 'localhost',
                'login' => 'root',
                'password' => 'admin',
                'database' => 'tiendavirtual',
                'prefix' => '',
                //'encoding' => 'utf8',);
        );
```

4.3 Modelo Base de Datos

Según las convenciones de CakePhp



Fuente: Autor

4.3.1 Mapear Datos

La clase para cada tabla mapeada se extiende de AppModel, implementando un ejemplo:

```
<?php
class User extends AppModel {
var $name = 'User';
        public $validate = array( 'username' => array('notempty' => array(
        'rule' => array('notempty'),
        'message' => 'Ingrese Nombre de Usuario',          ),
        ),
        'password' => array('notempty' => array('rule' => array('notempty'),
        'message' => 'Ingrese Password',
        ),
        )                                                          );
}
?>
```

4.3.2 Relaciones entre los modelos

Las relaciones de las tablas dentro de los modelos los cuales son consideradas un punto de acceso a la base de datos, definiendo reglas de validación:

Relación	Asociación
Uno a uno	HasOne
Uno a muchos	HasMany
Muchos a uno	belongsTo
Muchos a muchos	HasAndBelongsToMany

La clase se implementaría de la siguiente manera:

```
var $belongsTo = array(
                'Categoria' => array(
                        'className' => 'Categoria',
                        'foreignKey' => 'categoria_id',
                        'conditions' => '',
                        'fields' => '',
                        'order' => '')
        );
var $hasAndBelongsToMany = array(
                'Image' => array(
                        'className' => 'Image',
                        'joinTable' => 'images_productos',
                        'foreignKey' => 'producto_id',
                        'associationForeignKey' => 'image_id',
                        'unique' => true,
                        'conditions' => '',
                        'fields' => '',
                        'order' => '',
                        'limit' => '',
                        'offset' => '',
                        'finderQuery' => '',
                        'deleteQuery' => '',
                        'insertQuery' => '')
        );
```

4.4 Generación de Lógica de Negocio

A continuación se definirá un mantenimiento básico de un controlador del Modelo User, para que un controlador sea válido debe poseer la siguiente estructura users_controller.php debe estar en plural en relación al nombre del modelo y al final _controller.php.

```
<?php
class UsersController extends AppController {
var $name = 'Users';
var $components = array('Auth');
var $helpers    = array('Html','Form');
function index() {
                $this->User->recursive = 0;
                $this->set('users', $this->paginate());
}
function view($id = null) {
                if (!$id) {
                        $this->Session->setFlash(__('Invalid user', true));
                        $this->redirect(array('action' => 'index'));}
                $this->set('user', $this->User->read(null, $id));
}
```

```
function add() {
      if (!empty($this->data)) {
                  $this->User->create();
          if ($this->User->save($this->data)) {
                  $this->Session->setFlash(__('The user has been saved', true));
                  $this->redirect(array('action' => 'index'));}
           else {
            $this->Session->setFlash(__('The user could not be saved. Please, try again.', true));
}}}
function edit($id = null) {
       if (!$id && empty($this->data)) {
                  $this->Session->setFlash(__('Invalid user', true));
                  $this->redirect(array('action' => 'index'));
       }
       if (!empty($this->data)) {
          if ($this->User->save($this->data)) {
                  $this->Session->setFlash(__('The user has been saved', true));
                  $this->redirect(array('action' => 'index'));}
         else {
          $this->Session->setFlash (__('The user could not be saved. Please, try again.', true));
          }}
        if (empty($this->data)) {
          $this->data = $this->User->read(null, $id); }}
  function delete($id = null) {
                  if (!$id) {
                          $this->Session->setFlash(__('Invalid id for user', true));
                          $this->redirect(array('action'=>'index'));
                  }
                  if ($this->User->delete($id)) {
                          $this->Session->setFlash(__('User deleted', true));
                          $this->redirect(array('action'=>'index'));
                  }
                  $this->Session->setFlash(__('User was not deleted', true));
                  $this->redirect(array('action' => 'index')); }
          }
```

4.5 Componentes Utilizados

Para que la autentificación funcione, dentro del controller/component se debe implementar las opciones de seguridad las cuales se encuentra en http://api.cakephp.org/class/auth-component API de cake, sobre la cual existen componentes ya definidos y propios de cake.

4.6 Implementar Seguridad

1.-Para aplicar seguridad en cada uno de los formularios se debe implementar una variable que contenga un arreglo de nombre de la clase que implemente la autentificación quedando de la siguiente manera var $components = array('Auth');

2.-La implementación de la clase que maneje la seguridad ubicada en app\controllers\components, debe heredar de Objetc, sobre la que se define parámetros como base y que deben estar correctamente definidos, como los siguientes.

```
var $components = array('Session', 'RequestHandler');
var $userModel = 'Cliente';
var $fields = array('nombres' => 'nombres', 'password' => 'password');
```

3.-Entre las funciones que necesariamente deben estar definidas están los ejemplos de seguridad propios de cake se encuentran en el siguiente repositorio http://api.cakephp.org/view_source/auth-component/#line-745

```
function login($data = null) {
        $this->__setDefaults();
        $this->_loggedIn = false;

        if (empty($data)) {
                $data = $this->data;
        }

        if ($user = $this->identify($data)) {
                $this->Session->write($this->sessionKey, $user);
                $this->_loggedIn = true;
        }
        return $this->_loggedIn;
    }
/**
 *Todo login debe tener una function de cierre de sesión
*/
    function logout() {
        $this->__setDefaults();
        $this->Session->delete($this->sessionKey);
        $this->Session->delete('Auth2.redirect');
        $this->_loggedIn = false;
        return Router::normalize($this->logoutRedirect);
    }
```

Dentro de la Autentificación al validarse los campos definidos en la clase auth.php tales como password y nombres se pasara al login redirect que este a su vez permite que la autentificación se concrete permitiendo ingresar a las vistas.

4.7 Manejo de Imágenes

Para la visualización de imágenes se debe añadir phpThumb dentro de la carpeta vendors y un código genérico de respaldo se encuentra bajo el repositorio http://phpthumb.sourceforge.net .

Para que este módulo se implemente dentro de la vista que utiliza la visualización de imágenes, se deber añadir $html->image('uploads' . DS . 'images' . DS.$image['Image']['img_file']); que lo que hace es, tomar el path de la ubicación de la imagen la cual antes debe estar subida en ..\app\webroot\img\uploads.

Una vez ubicada la clase para la implementación y utilización se debe añadir el siguiente código , el cual debe contener el tipo de archivo de carga que se aceptara, el tamaño de la imagen, el directorio sobre el cual se guardara, cabe recalcar que esta función debe ser implementada sobre el modelo que poseerá dicha función.

```
var $actsAs = array(
      'MeioUpload' => array(
            'img_file' => array( 'dir' => 'img{DS}uploads{DS}images',
            'create_directory' => false,
            'allowed_mime' => array('image/jpeg', 'image/pjpeg', 'image/png'),
            'allowed_ext' => array('.jpg', '.jpeg', '.png'),'zoomCrop' => true,
            'thumbsizes' => array('normal' => array('width' => 400, 'height' => 300),),
             'default' => 'default.jpg'))
);
```

4.8 Generación Vista

Para que una vista reciba datos, el envío desde el controlador debe estar registrado bajo la estructura de link $this->Html->link en la cual para que se envíe el dato se debe definir ‘controller’=> ‘nombredelcontrolador’,’action’=>’nombrevistarecibedato’,’datoenviado’

1.-Para capturar los datos enviados con set en el caso del controller view se debe aplicar la siguiente sintaxis. $user['User']['username']; donde $user indica el objeto de la clase User definida en el modelo, seguido del nombre de campo.

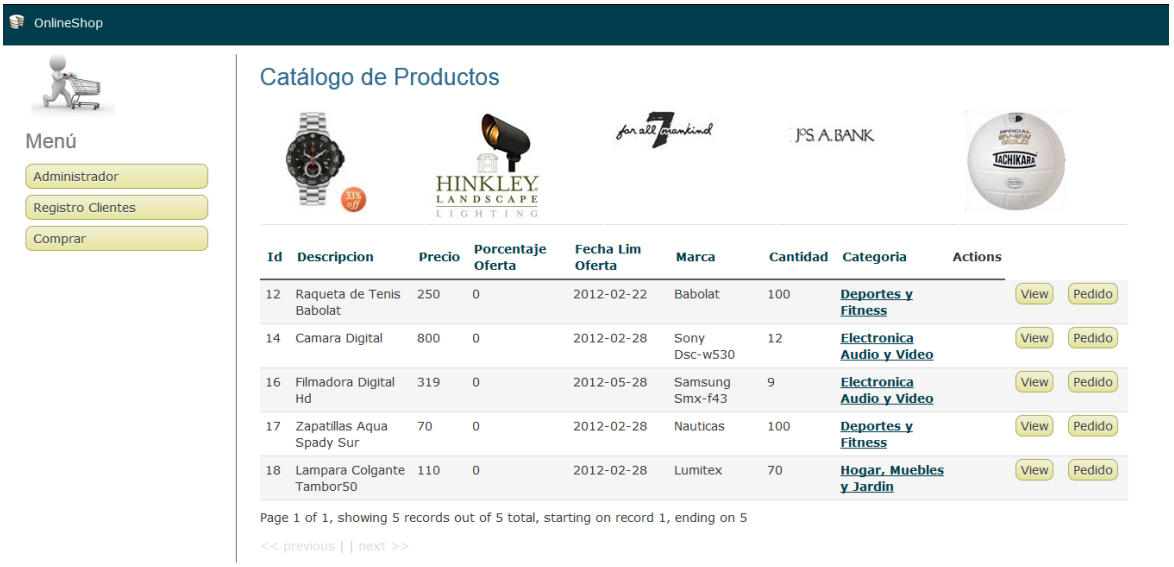
2.-Para implementar la autentificación en la vista se debe implementar $session->flash('auth'); antes de definir el la creación del formulario, es decir antes de la sentencia $form->create('User',array('action'=>'login')); de la siguiente manera.

4.9 Casos de Uso

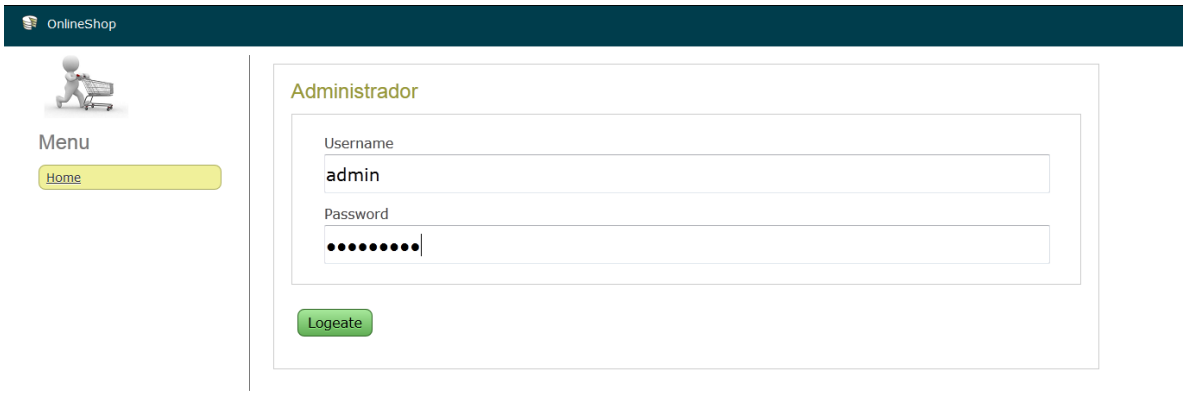
Revisar Anexo 1

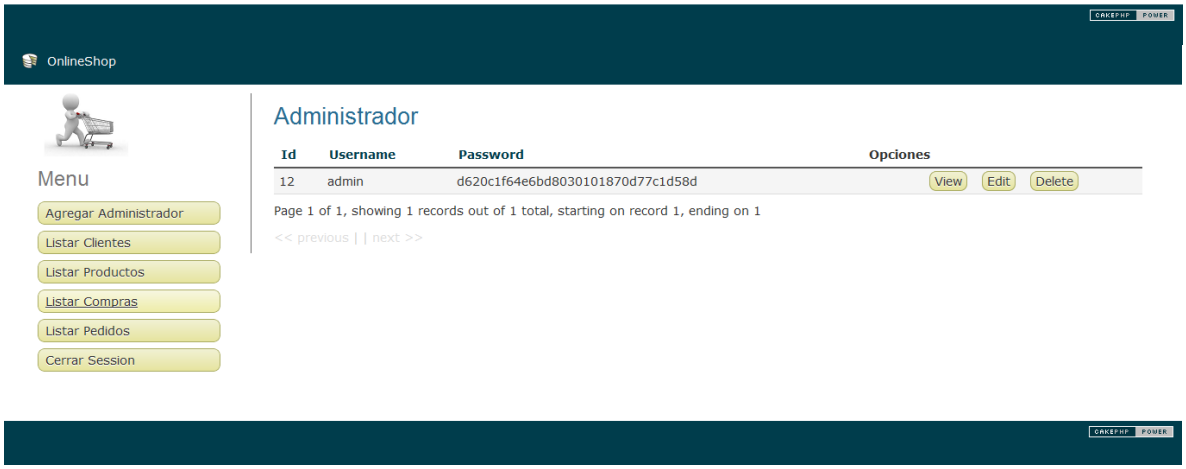
4.10 Interfaz de la Aplicación

Usuario Online

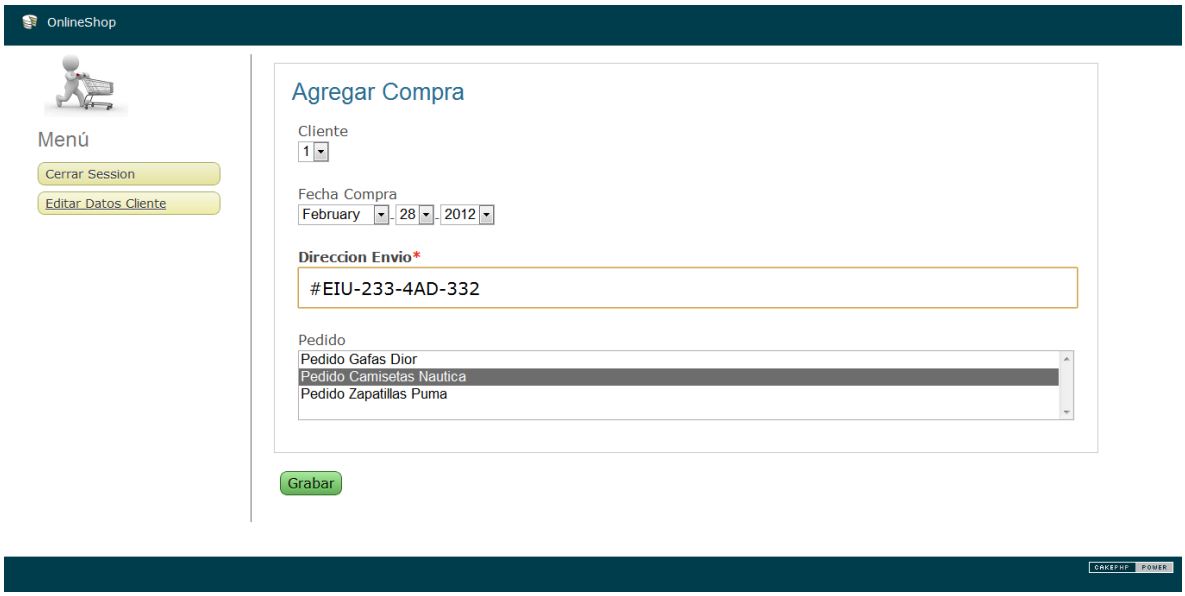


Administrador





Cliente Registrado



Conclusiones

Finalmente, para concluir la elaboración de este capítulo, CakePhp posee un diseño rígido no muy adaptable a la lógica del programador, además de ello existe gran cantidad de convenciones tanto a nivel de base de datos como de la estructura que utiliza, los componentes no se encuentran dentro del core del framework sino más bien, deben programarse, o en el mejor de los casos reutilizarse.

Al implementar imágenes, el manejo resulta complicado; la seguridad es rígida y centraliza para todo el controlador, forzando su utilización en todas las vistas asociadas, sin embargo dentro de sus beneficios se encuentra el scaffolding, el cual reduce en gran medida las líneas de programación, maneja componentes de manera predeterminada, dicha característica se visualiza en su diseño al ejecutar sus propias grillas y diseño estable, Cake se convierte en una opción sólida en proyectos pequeños convirtiéndose en una herramienta poderosa.

CAPITULO V
FRAMEWORK PRADO

Introducción

En el siguiente capítulo se realizará una descripción del Framework Prado, para posteriormente examinar aspectos como características generales, estructura, convenciones o reglas tanto de la base de datos, activerecord, controladores, vistas, configuración e integración API, dado que prado posee sus propios controles y requerimientos, que deberán considerarse antes de efectuar el caso práctico, para finalizar se abordarán temas basados en el manejo de componentes, eventos, plantillas mediante la explicación de las clases, para luego establecer los pasos para su creación y utilización y se explicara en que consiste el Scaffolding.

5.1 Framework Prado

Framework basado en componentes gracias a la creación de instancias, ofreciendo una programación orientada a eventos a causa de las actividades generadas por los usuarios, las mismas que son capturas en los eventos del servidor, para el desarrollo de aplicaciones web en PHP 5, facilitando la interacción con los componentes, como se mencionó en el capítulo II (Un componente es una pieza de programa que es autocontenido), además los métodos y funciones pueden asociarse con los eventos reduciendo significativamente la codificación repetitiva de bajo nivel.

5.2 Características de Prado

- Reutilización: no sólo de código propio, sino también código generado por otros, de una manera fácil.
- Soporte de I18N: crea aplicaciones con múltiples idiomas.
- Ofrece seguridad.
- Programación por evento: las actividades del usuario final, son capturadas como eventos del servidor mejorando las interacciones del usuario.
- Soporte de bases de datos mediante Active Record o el mapa completo de los objetos del negocio SqlMap.
- Soporte de AJAX: sin escribir una sola línea de código JavaScript.
- Integración de equipo, la capa de presentación y la capa lógica son almacenados por separado.
- Compatibilidad XHTML
- Potente manipulación de errores ,excepciones y registro de mensajes; manejo de errores personalizable y localizable.
- Extensible.

5.3 Estructura de Prado

Assets: Directorio de almacenamiento de archivos de recursos (como imágenes, sonidos, videos, hojas de estilo CSS, JavaScript, etc) guardados como archivos privados pero que sin embargo se hacen públicos por instrucción en el código fuente, de manera que serán visibles por todos a través del navegador.
Protected: Es el directorio principal de la aplicación, utilizado para guardar scripts privados e información configurada como inaccesible a los usuarios finales.
Runtime: Archivo de información del tiempo de ejecución de las aplicaciones de almacenamiento, tales como estado, datos almacenados en caché, etc; este directorio no debe ser modificado. Todo el desarrollo se realiza dentro de la carpeta /protected, sobre este directorio se encuentran las carpetas.
Pages: Ruta de acceso a las páginas, almacena todas las páginas prado, dentro de protected.
application.xml: Archivo sobre el cual se realiza la configuración para el sitio a desarrollar.

5.4 Convenciones, ActiveRecord y Base de Datos

Para entender cómo se realiza el mapeo de datos se debe partir de analizar el funcionamiento del application.xml.

- En el <module class="System.Data.ActiveRecord.TActiveRecordConfig" ConnectionID="db" /> permite mapear las tablas de datos, pero aquí el nombre que recibe cada archive mapeado es, ActiveRecord.php donde según el nombre de la base debe ser cambiado quedando así usersRecord.php.
- La sección de path, permite crear un path sobre el cual se emitirán acciones, como por ejemplo namespace="Application.database.* que define, sobre el archivo database se emitan ActiveRecord.php según lo defina el usuario, de manera predeterminada, para la generación de active records se la puede efectuar mediante consola que, lo que hace es devolver el ActiveRecord.php mapeado para la implementación posterior.

```
<paths>
        <using namespace="Application.database.*" />
</paths>
```

Convenciones Base de Datos

1.-Las claves primarias deben estar definidas como id, cada una de las relaciones existentes en la base de datos se debe implementar un procedimiento en el que se debe declarar el nombre de la relación a implementar, si se tiene una relación de uno a muchos, la variable es de tipo array y todas estas variables de relación deben ser declaradas como públicas.

```
 public $categoria;
 public $productos=array();
 public $pedidos=array();
public static $RELATIONS=array (
    'categoria' => array(self::BELONGS_TO, 'CategoriasRecord', 'categoria_id'),
    'productos' => array(self::MANY_TO_MANY, 'ProductosRecord', 'image_id'),
    'pedidos' => array(self::HAS_ONE, 'clientesRecord', 'compra_id'),
);
```

5.5 Desarrollo con Prado

Ubicado Prado en una carpeta o directorio accesible vía web.

- La generación de un proyecto se realiza vía consola, para lo cual partimos de implementar dentro del path de las variables de entorno de Windows, el path del php.exe, ubicado dentro del servidor.
- Accedemos a la consola y la ejecutamos con permisos de administrador.
- Nos ubicamos en el directorio accesible vía web, para factor de ejemplo lo definiremos como localhost\www y accedemos a la carpeta \prado\framework sobre dicha ubicación ejecutamos el comando prado-cli.php –c nombreproyecto.

5.5.1 Configuración integración API de desarrollo

Para que el desarrollo sea más intuitivo prado ofrece la ventaja de integrarse con dreamweaver CS5.5, gracias a su extensión PRADO.mxp que se encuentra en la carpeta \editors\Dreamweaver\.Sin embargo este archivo ya no viene junto en las versión 3.10 en adelante, pero se lo puede obtener en el repositorio de prado.

Dentro Dreamweaver CS5.5 se deben configurar las siguientes opciones:

1. En la ruta C:\ Archivos de Programa\ Adobe \Adobe Dreamweaver CS3\ Configuration\ dentro del archivo Extensions.txt añadir
    a. ,PAGE,SKIN: Todos los Documentos
    b. , PAGE, SKIN: Documentos HTML respetando las comas y además sin espacios.
2. Abrir C:\Archivos de Programa \ Adobe \Adobe Dreamweaver CS3 \ Configuration\ DocumentTypes\ MMDocumentTypes.xml Agregar page y skin, en documenttype id =”HTML” dentro de las opciones winfileextension, macfileextension.
3. Habilitar los archivos .tpl (para el uso de MasterPages o templates) y .skin, dentro del DWCS3 en el menú Edición->Preferencias->Tipos de archivo / Editores.
    a. Clic en el (+) de Extensiones y agregar .tpl .skin y luego clic en el (+) de editores y buscar el archivo *Dreamweaver.exe*, finalmente clic en OK y reiniciar el DW.

4. Luego abrir la aplicación Adobe Extensión Manager CS5.5, sobre el que se trabajara.

5.5.2 Active Record (Modelo de Datos)

Permite mapear las tablas de datos, pero aquí el nombre que recibe cada archivo mapeado es ActiveRecord.php, donde según el nombre de la base debe ser cambiado quedando así nombretablasRecord.php.
Entre sus funcionalidades se encuentran crear, recuperar, actualizar y eliminar registros, facilita la búsqueda SQL, además de definir relaciones de dependencia asociadas.
Se puede generar Activer Record por consola mediante la siguiente instrucción:

```
C:\wamp\bin\php\php5.3.5\php.exe C:\wamp\www\prado\framework\prado-cli shell .
generate contacto Application.database.contactoRecord
```

Entre sus métodos se encuentran:

FindAll (): Retorna todos los datos de salida asociados al ActiveRecord, este método está relacionado con el finder () utilizado para filtrar datos.

```
nombreRecord::finder()->findAll()
```

FindByPk(): Realiza el filtro directamente a la clave principal, por lo que no necesita especificar el campo sobre el filtro.

```
nombreRecord = $finder->findByPk($primaryKey);
```

Criterio: Se pueden definir criterios para el filtro de datos

```
$criteria = new nombreRecord;
$criteria->Condition = 'username = :name AND password = :pass';
$criteria->OrdersBy['name'] = 'asc';
$criteria->Limit = 10;
```

Sql: Para implementar sentencias sql propias se debe antes definir una clase adicional heredada del nombreRecord sobre el que se desea obtener los datos.

```
class UserRecord2 extends UserRecord
{
   public $another_value;
}
```

Luego generar el sql

```
$sql = "SELECT *  FROM users";
nombreRecord::finder('UserRecord2')->findAllBySql($sql);
```

DeleteByPk(): Como en su nombre lo describe permite eliminar registros directamente indicando su clave primaria.

```
$finder->deleteByPk(array($key1,$key2));
```

DeleteAllByPks(): Borra multiples records

```
$finder->deleteAllByPks (array ($key1, $key2), array ($key3, $key4)...);
```

5.5.3 Componentes

Las librerías o componentes que utiliza prado se encuentran en el core, fuera del proyecto generado dentro de ../framework, entre los más usados se encuentran:
Data:Componente de manejo ActiveRecord.php y clases de enlace php con la base de datos.
Dentro de la librería se encuentra definido scaffold(generación código rápido),relations(implementa relación entre modelos).
Web: Este componente se encarga de la interacciones del usuario, es por ello que hablamos de un framework orientada a eventos, el componente utilizado es JavaScript ubicado en framework\Web\Javascripts\source\prado aquí se encuentran los recursos usados por los controles de presentación usados.
Los controles se encuentran en framework\Web\UI\WebControls, estos elementos se encuentran mapeados dentro del core de framework para su utilización, a nivel de vista.
Un componente es una instancia de TComponent o su clase hija, los cuales proseen sus propiedades que describe un aspecto específico del componente, como el color de fondo, el tamaño de fuente, etc y eventos. La creación de un componente es similar a instanciar un objeto en donde toda la lógica del componente se encuentra en la clase.

```
$component = new ComponentClassName;
$component = Prado::createComponent('ComponentType');
```

5.5.4 Controles

Un control es una instancia de TControl clase o la subclase, y la relación padre hijo entre los controles , cada control de los padres puede tener controles secundarios de uno o varios, además un control es considerado un componente con interfaz de usuario,para diferenciarlos cada control posee un ID que lo diferencian de su control padre existe.
Prado posee sus propios controles, entre los más usados se encuentran:

<com:TTextBox ID="Id" visible="false"/>	(similar al control input type=text html)

<com:TActiveImage ID="image1" ImageAlign="absmiddle" ImageUrl="imagenes/noFoto.jpg" Width="120px" Height="90px"/>	(similar al control img)
<com:TActiveFileUpload ID="fileupload1" AutoPostBack="true" OnFileUpload="fileUploaded" />	(simula un input tipo file)
<com:TActiveButton ID="subir" OnCallBack="buttonClicked" Text="Grabar" />	
<com:THyperLink NavigateUrl="<%= $this->Service> constructUrl('productos.ListImagen') %>"Text="Imágenes"/>	(simula etiqueta a de html)
<com:TLabel ID="Id" />	(similar a un input type=”label”)
<com:TRepeater ID="Repeater" ItemRenderer = "Application.pages.clientes.PostRendererTarjetaCredito" AllowPaging="true" AllowCustomPaging="true" PageSize="5"/>	
<com:TListBox ID="lstItems" width="150"/>	
<com:TPager ControlToPaginate="Repeater" OnPageIndexChanged ="pageChanged" />	

Cada Control posee propiedades propias y eventos asociados, entre los eventos comunes se encuentran OnClick, onTextChanged para mayor información acerca de sus eventos se encuentra, el API de prado, bajo la siguiente dirección http://www.pradosoft.com/docs/manual/li_System.Web.UI.ActiveControls.html.

5.5.5 Eventos

Como se mencionó antes, los eventos están asociados a los componentes con métodos especiales como sus valores.

Un evento de un control se define por la existencia de un método, para obtener más información acerca de todos los eventos asociados con los componentes buscar información en http://www.pradosoft.com/docs/manual/System/PradoBase.html , el siguiente ejemplo explica la lógica de los eventos asociados a los controles.

```
<com:TTextBox ID=”Username” />
<com:TCustom Validator
ForeColor=”#9999CC”
ControToValidate=”Username”
ErrorMessage=”Sorry, your username is taken by someone else.”
OnServiceValidate=”checkUsername”
Display=”Dynamic”
```

Dentro de la clase asociada al .page

```
public function checkUsername($sender,$param){
 $param->IsValid=NombreRecord::finder->findByPk($this->Username->Text)===null;
 }
```

5.5.6 Plantillas

Prado utiliza templates los mismos que son colocados, dentro de una carpeta en /protected/templates, los cuales posee para su diseño dos extensiones .tpl y .php.
El archivo .php como mínimo debe implementar dicha clase, en la cual se puede definir el cierre de sesión.

```
<?php
class MasterPage extends TTemplateControl
{
        public function logoutButtonClicked($sender,$param)
          {
            $this->Application->getModule('auth')->logout();
            $url=$this->Service->constructUrl($this->Service->DefaultPage);
            $this->Response->redirect($url);
   }
}
?>
```

Archive .tpl y su estructura básica, las páginas que utilicen este contenedor debe declararse al principio <%@ MasterClass="MasterPage" %> <com:TContent ID="cphCuerpo"> </com:TContent> utilizando dentro del TContent el ID definido en el MasterPage.

```
<com:THead>
<com:TContentPlaceHolder ID="cphHead" />
</com:THead>
<com:TForm>
<com:TContentPlaceHolder ID="cphCuerpo" />
</com:TForm>
</body>
</html>
```

5.5.7 Páginas (View)

Se visualizan por los usuarios finales, sin embargo para poder visualizar el .page debe tener antes definido toda su lógica, para interacción con el usuario en él .php, tanto la página de visualización y la clase asociada deben poseer el mismo nombre.
Su funcionamiento es simple, el envío de la página que contiene el formulario se llama devolución de datos y se la puede considerar un caso que sucedió en el lado del cliente, planteada por el usuario.
Prado genera una división llamada page service, para procesar las solicitudes de los usuarios, almacenándolas en un directorio especificado por la propiedad BasePath del servicio de la página, este directorio se encuentra ubicado en \protected\application.xml dentro de la configuración de los servicios.

```
<services><service id="page" class="TPageService" DefaultPage="Home" /> </services>
```

O a su vez cada directorio, puede haber un archivo de configuración de la página llamado config.xml, que contiene las configuraciones eficaces.

```
<?xml version="1.0" encoding="utf-8"?>
  <configuration>
      <authorization>
        <allow roles="admin" />
        <deny users="*" />
      </authorization>
</configuration>
```

5.5.8 Scaffolding

Prado ofrece la posibilidad de generar mantenimientos funcionales, eficiente

1.-Se implementa los controles com: TScaffold dentro del .page.

```
<com:TScaffoldView RecordClass="UserRecord" />
```

Genera una lista de registros asociados al RecordActive.

2.- Si por lo contrario se desea editar los archivos usados se debe implementar.

```
<com:TScaffoldListView RecordClass="UserRecord" >
   <prop:List.ItemTemplate>
   </prop:List.ItemTemplate>
</com:TScaffoldListView>
```

3.-Además permite adaptar la visualización de la interfaz al combinar.

```
<com:TScaffoldEditView ID="edit_view" RecordClass="UserRecord" />
<com:TScaffoldListView EditViewID="edit_view" RecordClass="UserRecord" />
```

Cada una de los controles posee asociado un TScaffoldSearch el cual por sí solo no cumple su objetivo que es la búsqueda de registros, actuando como una consulta.

Conclusión

Una vez realizado el estudio del framework Prado se determinó el funcionamiento, características, requerimientos, estructura, configuraciones, componentes,activerecord,vista,controladores,plantillas y los beneficios que estos ofrecen así también se presentó las convenciones comunes tanto a nivel de base de datos, como a nivel de modelado o active récord, además se definieron los controles que manejan el scaffolding y el uso de eventos de esta manera se tendrá bases sólidas para efectuar el caso práctico mediante dicha herramienta.

CAPITULO VI

ONLINE SHOP DESARROLLADO EN PRADO

Introducción

El siguiente capítulo pretende ser una guía en el desarrollo de una aplicación, generada con Prado dentro de la cual se abordarán temas como instalación, conexión hacia la base de datos, mapeo de datos, relaciones entre los modelos, estructura de la lógica de negocio, implementación de componentes, manejo de imágenes, seguridad, generación de vistas gracias a la utilización de eventos en el desarrollo del caso práctico, antes ya mencionado, para finalizar el caso práctico posee casos de uso, y además se muestra la interfaz de la aplicación, generada en función de la herramienta y de los requerimientos.

6.1 Instalación de Prado

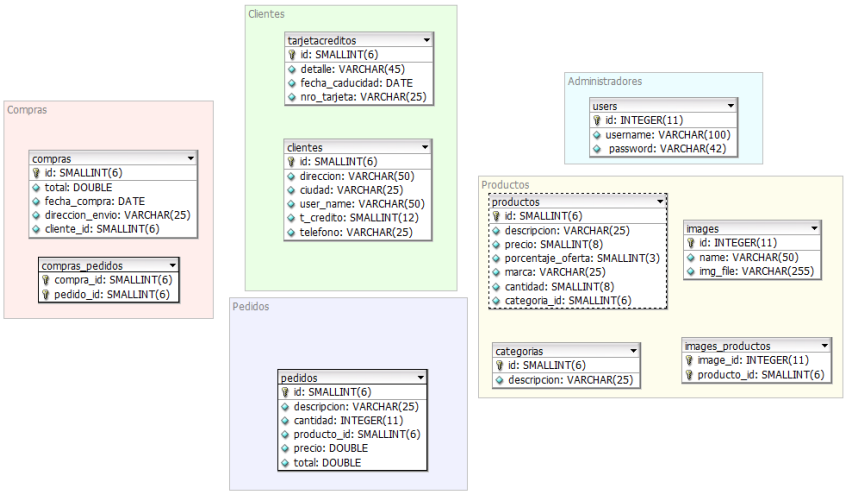
- Descomprima el archivo de PRADO en una carpeta o directorio accesible vía web.
- La generación de un proyecto se realiza vía consola, para lo cual partimos de implementar dentro del path de las variables de entorno de Windows, el path del php.exe, ubicado dentro del servidor.
- Accedemos a la consola y la ejecutamos con permisos de administrador.
- Nos ubicamos en el directorio accesible vía web, para factor de ejemplo lo definiremos como localhost\www y accedemos a la carpeta \prado\framework sobre dicha ubicación ejecutamos el comando prado-cli.php –c nombreproyecto.

6.2 Conexión hacia la Base de Datos

Para la conexión se debe modificar el archivo application.xml, dentro de la carpeta /protected, donde las variable definidas en el ConnectionString, cambian según el tipo de base de datos en la cual la modificación es la siguiente.

```
<?xml version="1.0" encoding="utf-8"?>
<application id="pradocontact" mode="Debug">
<paths>
    <using namespace="Application.database.*" />
</paths>
<modules>
<module id="db" class="System.Data.TDataSourceConfig">
  <database ConnectionString="mysql:host=localhost;dbname=pradocontact" username="root"
password= "admin"/>
</module>
<module class="System.Data.ActiveRecord.TActiveRecordConfig"ConnectionID="db"/>
</modules>
 </application>
```

6.3 Modelo Base de Datos



Fuente: Autor

6.3.1 Relaciones entre los Modelos

Las relaciones se deben declarar para cada uno de los modelos de datos con la estructura antes establecida, cambiando solo el tipo de relación, para que la relación quede definida, se debe identificar en las tablas relacionadas es decir si existe una relación MANY_TO_MANY entre productos e imagen en el ProductosRecord.php se debe definir 'images' => array(self::MANY_TO_MANY, 'imagesRecord', 'producto_id'), y en el ImageRecord.php se debe implementar la siguiente estructura 'productos' => array(self::MANY_TO_MANY, 'ProductosRecord', 'image_id') de esta manera la relación queda establecida.

6.4 Generación de Lógica de Negocio

La lógica del negocio a desarrollar se encuentra en el .page es decir cada mantenimiento o función requerida posee .page y a su vez todo .page se extiende de TPage.
A continuación se explicara el mantenimiento realizado a un ActiveRecord, desde el ingreso, modificación, listado y eliminación:
Ingreso: Se realiza la instanciación del objeto con la clase del Record mapeada, la sintaxis para guardar es .save().

```
<?php
class NewPedido extends TPage
{
   public function createButtonClicked($sender,$param)
   {
      if($this->IsValid)         {
         $userRecord=new pedidosRecord;
```

```
            $userRecord->descripcion=$this->descripcion->Text;
            $userRecord->cantidad=$this->Cantidad->Text;
            $userRecord->producto_id=$this->Producto_id->Text;
            $userRecord->precio=$this->Precio->Text;
            $userRecord->total=$this->Total->Text;
            $userRecord->save();
            $this->Response->redirect($this->Service->DefaultPageUrl);}}
 }
?>
```

Editar

```
 public function saveButtonClicked($sender,$param)
  {
      if($this->IsValid)
      {
         $userRecord=$this->UserRecord;
         $userRecord->id=$this->Id->Text;
         if(!empty($this->descripcion->Text))
         $userRecord->descripcion=$this->descripcion->Text;
          $userRecord->descripcion=$this->descripcion->Text;
         $userRecord->save();
         $url=$this->Service->constructUrl('productos.Categorias');
         $this->Response->redirect($url);}
   }
   protected function getUserRecord()
   {
      if( $this->Request['id']!==null)
      $username=$this->Request['id'];
      $userRecord=categoriasRecord::finder()->findByPk($username);
      if(!($userRecord instanceof categoriasRecord))
         throw new THttpException(500,'Username is invalid.');
      return $userRecord;
   }
```

En donde la function getUserRecord(), devolverá el registro a editar según el código enviado, y la función saveButton obtiene 2 parámetros donde $senter son los datos y $param, identifica el tipo de control luego se actualizan los datos en el Record.

Eliminar

Se establece dentro del archivo .tpl que permite visualizar los multiregistros mostrando el índex del Record seleccionado, dentro de este se implementa.

```
<com:TControl Visible="true">
    <com:TLinkButton Text="Eliminar"
       OnClick="deletePost"
       Attributes.onclick="javascript:if(!confirm('Are you sure?')) return false;" />
</com:TControl>
```

```
private $_productos;
 public function onInit($param)
   {
      parent::onInit($param);
      $userID=(int)$this->Request['id'];
      $this->_productos=productosRecord::finder()->findAll($userID);
      if($this->_productos===null)
         throw new THttpException(500,'Unable to find the specified post.');
   }
 public function deletePost($sender,$param)
  {
     $userID=$this->id->Text;
      productosRecord::finder()->deleteAll('id = ?',$userID);
      $url=$this->Service->constructUrl('productos.Productos');
      $this->Response->redirect($url);
  }
 public function getUser()
 {
      return $this->_productos;
 }
```

Dentro del evento OnInit se realiza el filtrado de datos con la instrucción productosRecord::finder()->findAll($userID);

Para luego pasar a la función deletePost donde sobre el registro se procede a borrarlo con la instrucción productos.

Record::finder()->deleteAll('id = ?',$userID);para después direccionar el browser con $url=$this->Service->constructUrl('productos.Productos'); $this->Response->redirect($url);

Listar Datos

El listar implementa 4 páginas 2 .php 1.page y 1.tpl, .page es la principal sobre la que se mostraran los datos, los cuales permiten manejar multiregistro, además de indicar la ubicación del archivo multiregistro que debe ser añadido.

```
 <com:TRepeater ID="Repeater"
    ItemRenderer="Application.pages.clientes.PostRendererTarjetaCredito"
    AllowPaging="true"
    AllowCustomPaging="true"
    PageSize="5" />
<com:TPager ControlToPaginate="Repeater" OnPageIndexChanged="pageChanged" />
```

El control TPage maneja la paginación así también dentro del primer .php debe contener la función de filtrado de datos , sobre el Record, para luego cargarse en el OnInit.

```
public function onInit($param){
      parent::onInit($param);
      $userID=(int)$this->Request['id'];
      $this->_clientes=clientesRecord::finder()->find('id = ?',$userID);
      if($this->_clientes===null)
         throw new THttpException(500,'Unable to find the specified post.');
          if(!$this->IsPostBack)  {
                $this->Repeater->VirtualItemCount=tarjetacreditosRecord::finder()->count();
                $this->populateData();}}
protected function getPosts($offset, $limit){
$userID=(int)$this->Request['id'];
$categorias=clientesRecord::finder()->findByPk($userID);
return tarjetacreditosRecord::finder()->findAll('id = ? ', $categorias->t_credito);}
```

El .tpl que contiene los multiregistros, primero comienza definiendo la captura de datos según el nombre del campo y datos a recibir debe realizarse de la siguiente manera:

```
<div class="post-box">
<com:TLiteral Text="<%# $this->Data->marca %>" />
<-div>
```

El siguiente .php que se relaciona directamente con este .tpl, el cual hereda datos de TRepeaterItemRenderer, dentro del siguiente ejemplo se implementó un eliminar en el .page y la clase posee las funciones para eliminarlo, queriendo dejar en claro que se puede efectuar cualquier operación sobre el multiregistro.

```
<?php
class PostRendererProducto extends TRepeaterItemRenderer
{
private $_productos;
public function onInit($param)
  {
       parent::onInit($param);
       $userID=(int)$this->Request['id'];
          $this->_productos=productosRecord::finder()->findAll($userID);
       if($this->_productos===null)  // if post id is invalid
          throw new THttpException(500,'Unable to find the specified post.');
  }
public function deletePost($sender,$param){
       $userID=$this->id->Text;
       productosRecord::finder()->deleteAll('id = ?',$userID);
       $url=$this->Service->constructUrl('productos.Productos');
       $this->Response->redirect($url);
    }
   public function getUser() {
       return $this->_productos;}}
?>
```

6.5 Componentes Utilizados y Manejo de Imágenes

Para la generación de imágenes dentro del .php implementar la función

```
public function fileUploaded($sender,$param)
	{
		if($this->fileupload1->HasFile && $sender->IsValid)
		{
			$foto=$this->fileupload1->FileName;
			$this->label1->Text="cargo";

			$path='imagenes/'.$foto;
			$val=$this->fileupload1->saveAs($path);

			if($val){$this->image1->ImageUrl=$path;}
		}
		else
		{
			$this->label1->Text="Error al subir imagen";
		}
	}
```

En la cual se utiliza el componente para carga de archivo definido en assets\97ab35ad\prado\activefileupload.

Para que este archivo sea llamado utilizar dentro del .page.

```
<com:TActiveImage ID="image1" ImageAlign="absmiddle"
ImageUrl="imagenes/noFoto.jpg" Width="120px" Height="90px" />
<com:TActiveFileUpload ID="fileupload1" AutoPostBack="true"
OnFileUpload="fileUploaded" />
<com:TActiveLabel ID="label1" Text="..." />
 <com:TActiveButton ID="subir" OnCallBack="buttonClicked" Text="Grabar" />
```

6.6 Implementar Seguridad

Para que el campo se encripte se debe utilizar

```
<com:TCustomValidator
  ControlToValidate="Password"
   ForeColor="#9999CC"
  ErrorMessage="Your entered an invalid password."
  Display="Dynamic"
  OnServerValidate="validateUser" />
<com:TTextBox ID="Password" TextMode="Password" />
```

Dentro de la página .php que maneje la autentificación, deberá implementar la función de validación de datos.

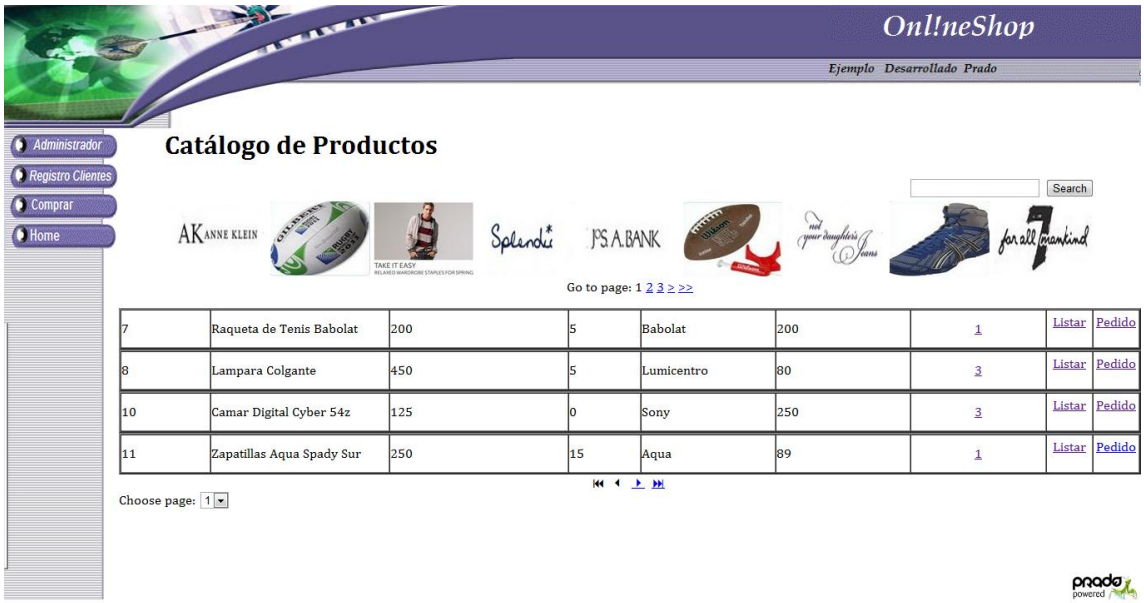
```
public function validateUser($sender,$param)
    {
        $authManager=$this->Application->getModule('auth');
        if(!$authManager->login($this->Username->Text,md5($this->Password->Text)))
            $param->IsValid=false;  // tell the validator that validation fails
    }
```

6.6 Casos de Uso

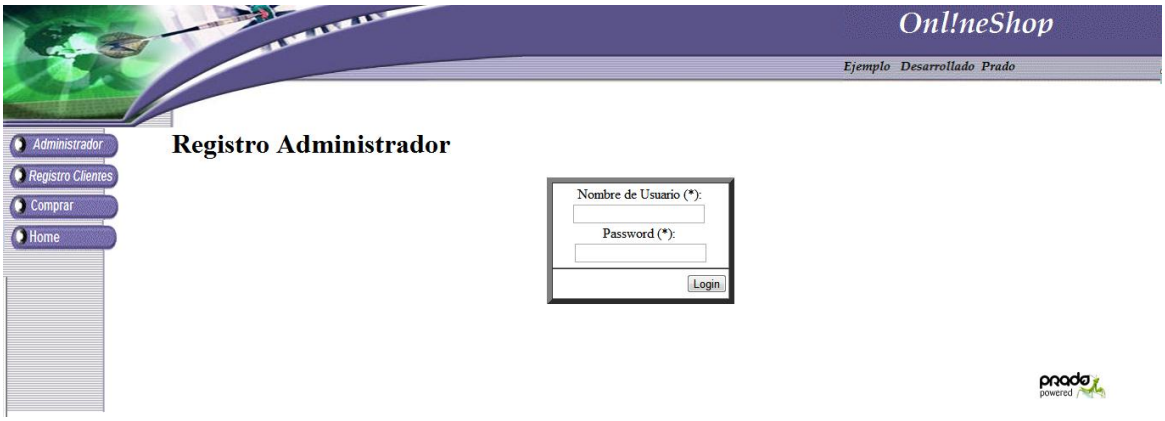
Ver Anexo 1

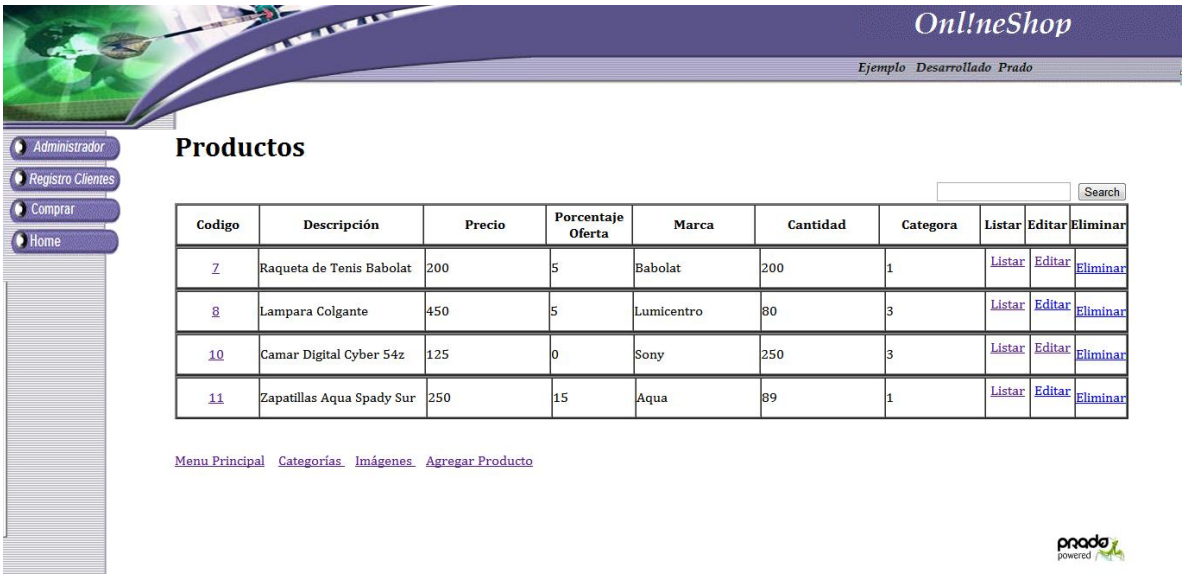
6.8 Interfaz de la Aplicación

Usuario Online

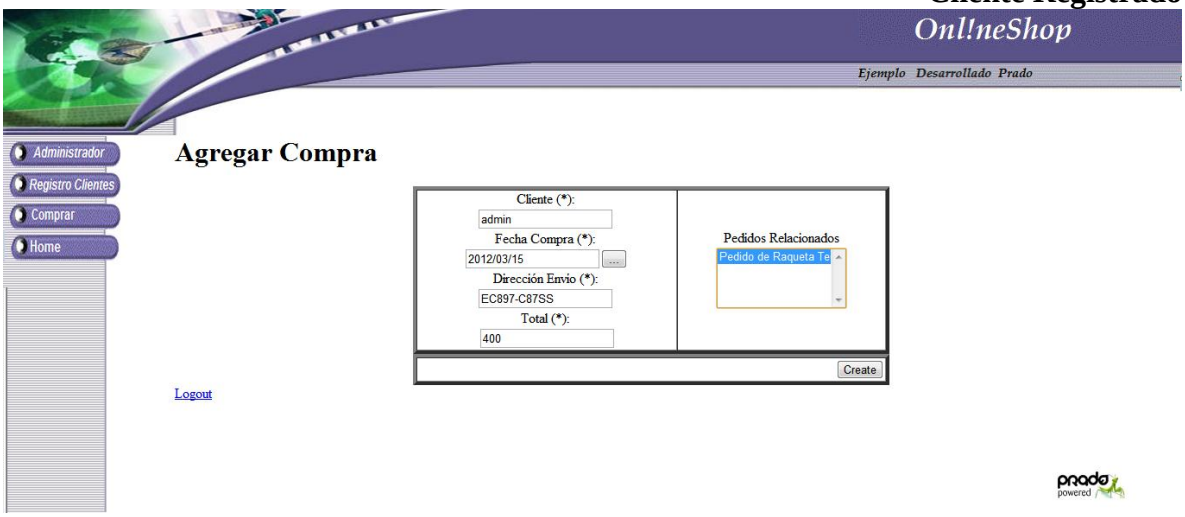


Administrador





Cliente Registrado



Conclusión

Prado posee un diseño flexible adaptable a la lógica del programador, además de ello, ofrece componentes ya integrados que permiten el manejo de eventos y permiten facilidad en la interacción sin embargo, el código se repite y es excesivo, puesto que no presenta una estructura donde se definirá su lógica.

El problema surge al utilizar antes otros frameworks debido que prado posee una estructura de programación diferente, hasta el punto de poseer su propios controles, razón por la cual presentan beneficios al facilitar al programador el uso de controles potentes y estables, entre sus ventajas se encuentran la utilización de las librerías, las cuales poseen incorporados componentes que reducen el tiempo en el desarrollo como en el caso del manejo de imágenes, desde mi perspectiva prado resulta una herramienta poderosa al poseer un amplio dominio de la misma puesto que su programación es propia.

CAPÍTULO VII

FRAMEWORK ZEND

Introducción

El siguiente capítulo pretende ser una guía en el desarrollo de una aplicación, generada con Zend dentro de la cual se abordarán temas como instalación, conexión hacia la base de datos, mapeo de datos, relaciones entre los modelos, estructura de la lógica de negocio, implementación de componentes, manejo de imágenes, seguridad gracias a la utilización de eventos en el desarrollo del caso práctico, antes ya mencionado, para finalizar el caso práctico posee casos de uso, y además se mostrar la interfaz de la aplicación, generada en función de la herramienta y de los requerimientos.

7.1 Framework Zend

Zend Framework considerado open source para PHP5 desarrollado por Zend, empresa encargada de la mayor parte de las mejoras hechas a PHP, por lo que se podría decir que es el framework "oficial".
Está diseñado para ser liviano, modular y extensible, posee un diseño minimalista a de más de permite libertad, proporcionando estructuras, convenciones y layouts de códigos reutilizables.

7.2 Características de Zend

- Totalmente orientado a objetos.
- Soporta la internalización y localización de aplicaciones.
- Facilita el setup de los proyecto mediante línea de comandos.
- Componentes con bajo acoplamiento, por lo que los puedes usar en forma independiente.
- Posee Zend_Test para facilitar el testing de nuestra aplicación.
- Programación estandarizada.
- Soporta adapters para múltiples tipos de bases de datos.
- Componentes para la autenticación y autorización de usuarios, envío de mails, cache en varios formatos, creación de web services, etc.

7.3 Estructura de Zend

Zend posee una estructura de archivos, de las cuales las carpetas principales son

Application: Crea subcarpetas que contendrán los modelos, vistas y controladores de nuestra aplicación.

Public: Almacena imágenes, scripts y archivos CSS.

Library: Contiene archivos descargados del Zend Framework y otras librerías propias.

Test: Su objetivo es hacer pruebas fáciles de leer, escribir y depurar.

.zfproject.xml: Contendrá su estado de estructura de la aplicación en formato XML, al ser generadas ya sea mediante la consola.

La estructura generada es la siguiente:

Application
- Configs
- Controllers
- Forms
- Layouts
- Models
- views
- Bootstrap.php

Library
Public
Tests
- Application
- Library
- bootstrap.php
- phpunit.xml

.zfproject.xml

7.4 Convenciones de la Base de Datos, Modelos, Controladores, y Vistas

El framework organizada las páginas las cuales deben antes estar definidas como acción en los controladores, cuyo nombre tiene el sufijo “Action”, sobre el igual cake posee una acción por defecto llamada índex; cada vista contiene un método antes referenciado en el controlador, caso contrario no se permite su visualización.
Cabe destacar que dentro de la convenciones con mayor importancia en Zend se encuentran a nivel de base de Datos, sin embargo para facilitar la interacción con la base de datos se

utilizan los modelos mapeados en forma de clase la cual hereda de diversas clases propias de zend cuya función es el manejo de dados, aunque la clase Zend_Db_Table, es la más utilizada puesto que insertar, actualizar y eliminar registros de una tabla en la base de datos.

Para usar Zend_Db_Table, necesitamos decir que base de datos usar junto con un usuario y una contraseña. Para no escribir código de más se utilizará un archivo de configuración para guardar estos datos, ubicado en application/configs/application.ini y agregamos.

```
resources.db.adapter = PDO_MYSQL
resources.db.params.host = localhost
resources.db.params.username = root
resources.db.params.password = admin
resources.db.params.dbname = tienda_virtual
```

De esta manera quedara establecida la conexión.

7.5 Desarrollo con Zend

Para comenzar a utilizar zend se debe configurar tres archivos.
htacces: ubicado en public, redirige todas las peticiones al index.php.

```
RewriteRule ^.*$ index.php [NC,L]
```

index.php: Decide que página mostrar haciendo referencia a un archivo application.ini.

```
$application = new Zend_Application(
  APPLICATION_ENV, APPLICATION_PATH . '/configs/application.ini' );
  $application->bootstrap()
              ->run();
```

application.ini: el cual contiene la configuración del sitio.
Bootstrap: registrando en el autoloader el namespace de nuestra aplicación para poder instanciar las clases que usemos sin necesidad de hacer antes un include de dicho archivo.

```
class Bootstrap extends Zend_Application_Bootstrap_Bootstrap
{
protected function _initAutoload(){
$moduleLoader = new Zend_Application_Module_Autoloader(array('namespace' => '',
'basePath' => APPLICATION_PATH);
 return $moduleLoader;
}
```

Para la creación de un proyecto se utiliza la siguiente sintaxis:

zf create project zf-tiendavirtua

7.5.1 Requerimientos

Los requisitos para implementar Zend Framework son los siguientes:

- PHP 5.1.4 (o más actual).
- Un servidor Web que soporte funcionalidades de mod_rewrite del apache ubicado, en el archivo htpd.conf dentro de la línea #LoadModule rewrite_module, para desactivarlo se debe borrar el numeral al inicio de dicha fila.
- Apache debe estar configurado para soportar archivos .htaccess, en el caso de no estar configurado se cambia la configuración de AllowOverride None a AllowOverride All.

7.5.2 Controladores

Un controlador, se extienden de Zend_Controller_Action, además un controlador posee una estructura muy simple utilizando el método init(), sobre el que se define tareas de inicialización.

Al ejecutar, zf automáticamente se genera una action llamado índex relacionado el nombre del action con una vista que será renderizada.

```
$this->view->render('user/index');
```

7.5.3 Modelos

Dentro del modelo se define la lógica del negocio, generando los mantenimientos mediante insertar, actualizar y eliminar. En general, los datos que requieren un proceso se cargan en el modelo.

Se podría afirmar que todo proceso de datos va en el modelo, de esta manera el controlador sólo funciona como intermediario entre la vista y el modelo cuando sea necesario, pretendiendo tener el código mucho más organizado y limpio.

Para facilitar la gestión de datos los modelos se extienden de Zend_Db_Table asumiendo que la tabla tiene un primary key llamado id.

```
<?php
class Application_Model_Register
{
  public function createUser()
  {
    $dbTableUser =Application_Model_DbTable_User();  //instanciar un objeto
    $dbTableUser->insert();
  }
}
?>
```

Entre las funciones que son implementadas se encuentran:

```
$sth=$this->db->prepare('select * from users');
 $sth->execute();
 return $sth->fetchAll();
```

7.5.4 Componentes

Zend Admite la utilización de varios componentes o librerías, entre los más usados se encuentran:

Zend_Layout: Fácil de proporcionar diseños en todo el sitio para sus aplicaciones MVC además de decorar Zend_View, que heredan las capacidades de dicho componente proporcionando una variedad de ayudantes y plugins para acceder al objeto de diseño desde el interior de otros componentes.

Zend_Form: Encargado de crear y manejar los formularios, por lo que es un componente de uso casi obligado en cualquier desarrollo web y uno de los que más llaman la atención al empezar a trabajar con este framework.

Es una herramienta realmente simple y potente que nos permitirá ahorrar tiempo al trabajar con formularios y generar un código mucho más preciso.

Zend_Auth: Esta librería se encarga de la autentificación de usuario, por lo general se valida nombre de usuario y contraseña.

Zend_Db: Librería base utilizada para conectar la base de datos con el php, utiliza varias base de datos la diferencia está en el tipo de adaptador.

Zend_File_Transfer: Librería utilizada para carga y descargas de archivos, posee filtros propios y validadores.

ZendX_JQuery: Librería extra no se encuentra dentro del core común del zend, considerada ayudantes de forma, considerada una alternativa adicional de Dojo.

Zend_Dojo: Considerado un componente de manejo de widgets o etiquetas, las cuales utilizan una variedad de diseños con la finalidad de proporcionar características de accesibilidad y estandarización.

Zend_Paginator: Componente flexible para paginar colecciones de datos y su presentación al usuario.

Zend_Session: Para definir una opción de configuración de sesiones.
Zend_Validate: Conjunto estándar de clases de validación, los cuales están listos para su uso. Entre las más comunes se encuentra isValid ()
Zend_Controller: Dentro de este componente se encuentra la lógica de la aplicación, considerado el patrón de diseño y de este componente las demás clases se extiende utilizando las interfaces y clases abstractas inclusive aquí se incluyen plugins o ayudantes para mejor el funcionamiento.
Zend_View: Maneja la vista, maneja el script de vista independiente del modelo utiliza filtros de salida y ayudantes de mejora es independiente de la plantilla.

7.5.4.1 Crear Componentes

1.- Al decir que un componente es un objeto; crear un componente resulta una tarea fácil puesto que comienza con la instanciación.

```
$form=new Zend_Form;
```

Facilitando el manejo de componentes se utiliza la consola, con la ayuda de la sentencia zf, en el caso de requerir la creación del layout se utiliza Zf enable layout.
Con ello se acaba de crear la carpeta application/layouts/scripts/ y dentro de la misma un archivo llamado layout.phtml. También actualizó el archivo application.ini y agregó la línea resources.layout.layoutPath = APPLICATION_PATH "/layouts/scripts/" a la sección [production].
Zend ofrece la posibilidad de adaptar el uso de sus componentes según las necesidad del programador haciéndolo flexible a modificaciones, como lo explica el siguiente ejemplo:

```
$db = Zend_Db_Table_Abstract::getDefaultAdapter();
$db->setFetchMode(Zend_Db::FETCH_OBJ);
 $sql = "select * from pedidos p, compras_pedidos cp where cp.compra_id='$id' and
 cp.pedido_id=p.id limit 0,10";
        $this->view->selection = $db->fetchAll($sql,Zend_Db::FETCH_BOTH);
```

Zend permite que sus componentes se modifiquen; como en el caso de Zend_Db al generar un sql personalizado; sin embargo mantiene su lógica interna, como lo es el uso de FETCH_OBJ, el cual maneja arrays de objetos y fetchAll cuya función es la comunicación del adaptador y el objeto al representarlo como cadena select.

7.5.5 Elementos

Zend tiene por default como elementos

- Button
- Checkbox / multiCheckbox
- Hidden
- Image

- Password
- Radio
- Reset
- Select / multi-select
- Submit
- Text
- Textarea

Como se mencionó anteriormente la utilización de componentes facilita el desarrollo, sin embargo para que la implementación del componente Zend_Form se realiza con éxito, se utilizan elementos los cuales poseen su propia estructura, para agregarlos existen 2 formas:

1.-instanciar el elemento y pasárselo al form

```
$form->addElement(new Zend_Form_Element_Text('username'));
```

2.- pasar el tipo del elemento al form y que este lo instanciar

```
$form->addElement('text', 'username');
```

7.5.6 Vista

Una vista tiene la extensión phml y es un objeto (Zend_View)**,** al que accedemos desde el controlador con $this->view**.**
Permitir separar el código que muestra la página del código que está en las funciones de las acciones.

Zend Framework provee de un Action Helper (una función que se utiliza muchas veces) llamada ViewRenderer. Este se ocupa de inicializar la vista en el controlador ($this->view).

7.5.7 Scaffolding

Existe un componente que aún está en desarrollo (Zend_Controller_Fron_Scaffold), seguramente. El scaffolding es un método para construir aplicaciones basadas en bases de datos, esta técnica está soportada por algunos frameworks del tipo MVC tales como prado, cakephp antes estudiados.
La información acerca del scaffolding en zend se encuentra en la siguiente página http://framework.zend.com/wiki/display/ZFPROP/Zend_Controller_Front_Scaffold, la cual describe en qué nivel de desarrollo se encuentra el scaffold y presenta demos de prueba.

7.6 Configurar Testing

Una de las características de zend frente a otros frameworks de desarrollo es el manejo de testing, para lo cual se debe realizar las siguientes configuraciones.
1.-Dentro del archivo ..zend\application\config\application.ini agregar
SetEnv APPLICATION_ENV development.
2.-Comprobar que dentro de ..zend\application\configs\application.ini se encuentran definidas las siguientes secciones.

```
[testing : production]
phpSettings.display_startup_errors = 1
phpSettings.display_errors = 1

[development : production]
phpSettings.display_startup_errors = 1
phpSettings.display_errors = 1
resources.frontController.params.displayExceptions = 1
```

3.-Al encontrase activada esta opción se podrá revisar donde se ocasiona el error mediante un log generado en la aplicación.

7.7 Tareas Comunes

Un sitio web que se encuentra en producción, debería tener configurado un virtual host para tener seleccionada la entrada al sitio directamente a la carpeta public.
Para ello se debería implementar el siguiente cambio:
1.-Cambiar la siguiente sintaxis en el apache http.conf.

```
<VirtualHost 127.0.0.1>
        DocumentRoot "C:\wamp\www\zend\zend\public"
        ServerName zend
        <Directory "C:\wamp\www\zend\zend\public">
       AllowOverride AuthConfig FileInfo Indexes Limit Options
       Order Deny,Allow
          Allow from all
        </Directory>
</VirtualHOst>
```

2.-Dentro de Windows en el área C:\Windows\System32\drivers\etc\hosts cambiar el fichero a 127.0.0.1 localhost zend jream sandbox zend.
3.-Ahora si escribimos en el navegador zend aparecerá directamente el sitio.

Manejo de Sessiones

1.-Para el manejo de sesiones se instancia un objeto asignando un nombre relacionado al componente de sesión.

```
Zend_Session::start();
$app = new Zend_Session_Namespace('app');
$app->filtro =$idfiltro;
```

2.-Recuperar sesión, con referencia al objeto instanciado.

```
$app = new Zend_Session_Namespace('app');
$id =$app->filtro;
```

Conclusión

Al finalizar este capítulo se realizó, el estudio del framework Zend definiendo su funcionamiento, requerimientos, estructura, configuración del testing, manejo de componentes, modelos,vista,controladore,elementos y los beneficios que estos ofrecen así también se presentó la estructura de las tareas más comunes, scaffolding de esta manera se tendrá bases sólidas para efectuar el caso práctico mediante dicha herramienta.

CAPITULO VII
ONLINE SHOP DESARROLLADO EN ZEND

Introducción

En el siguiente capítulo se realizará una descripción del Framework Zend, para posteriormente examinar aspectos como características generales, estructura, convenciones o reglas tanto de la base de datos, modelos, controladores, vistas, parámetros de configuración a nivel de herramienta así como la configuración del testing y requerimientos que deberán considerarse antes de efectuar el caso práctico, para finalizar se abordarán temas basados en el manejo de componentes, vistas, controladores, elementos estableciendo los pasos para su creación y utilización y se explicara en que consiste el scaffolding además se establecerán las tareas comunes requeridas por los desarrolladores mediante la explicación de las clases.

8.1 Instalación de Zend

- Descargar la versión mínima de http://framework.zend.com/download/stable.
- Ubicar la dirección, en este caso se encuentra en la carpeta pública.
- Revisar que el include_path hacia las librerías C:\wamp\www\zendFramework\library se encuentre definido en application.init.
- Registrar dentro de las variables del sistema la ruta de zf.bat en este caso se encuentra bajo C:\wamp\www\zendFramework\bin.
- Comprobamos abriendo la consola de cmd y colocamos zf.

Sobre la cual ya aparecen las opciones propias del framework, de esta manera se tendrá el zend instalado en nuestro sitio, al ubicarnos en la carpeta pública, en mi caso http://localhost/zend/zend/public/ donde fue colocada la carpeta pública tiene todos los datos que serán visibles.

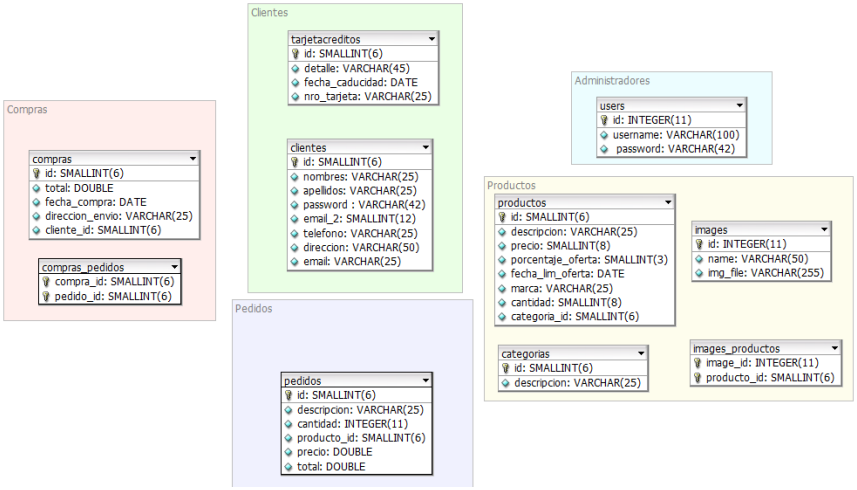
8.2 Conexión hacia la Base de Datos

Para implementar la conexión hacia la base de datos, como se explicó en el capítulo VII debe implementar su configuración, para conectar la base se la puede efectuar vía consola, gracias a la siguiente sintaxis:
zf configure db –adapter “adapter PDO-MYSQL”

```
C:\wamp\www\zend\zend>zf configure db-adapter "adapter-PDO MYSQL"
                        An Error Has Occurred
 At least one name value pair is expected, typcially in the format of
"adapter=Mysqli&username=uname&password=mypass&dbname=mydb"

Zend Framework Command Line Console Tool v1.11.11
Details for action "Configure" and provider "DbAdapter"
  DbAdapter
    zf configure db-adapter dsn section-name[=production]
```

8.3 Modelo Base de Datos



Fuente: Autor

8.3.1 Mapear Datos

Dentro del Mapeo de Datos, es recomendable utilizar la consola con la instrucción.

```
zf db-table nombretabla
nombremodelo
```

Se constanta en..\zend\application\models, dentro de dicho directorio se constanta que la clase fue creada, como se mencionó anteriormente la clase generada posee la siguiente estructura.

```
class Application_Model_User extends
Zend_Db_Table_Abstract
{
    protected $_name = 'users'; // definir nombre de clase
    protected $_primary='id'; //clave primaria
}
```

Dentro del modelo se definirán los métodos a utilizar por el controlador ejemplo:

```
public function addCategorias($id,$descripcion){
                $data = array('id' => $id, 'descripcion'
=>$descripcion,);
                $this->insert($data);
```

```
}
```

8.4 Generación de Lógica de Negocio

La lógica del negocio se centra en los modelos, sin embargo como se mencionó anteriormente, el controlador sirve de intermediario entre el modelo y la vista por esta razón es aquí en donde se implementa toda la programación, a continuación se explicar cómo se generan los mantenimientos.

Listar Datos: Al trabajar con programación orientada a objetos, se parte con la instanciación del modelo, luego se establece el método fetchAll(), para retorno de valores.

```
 public function indexAction()
  {
    $libros = new Application_Model_DbTable_Tarjetacreditos();
     $posts=$libros->fetchAll();
     $paginator=Zend_Paginator::factory($posts);

Zend_View_Helper_PaginationControl::setDefaultViewPartial('/paginator/items.phtml');
     if ($this->_hasParam('page')){
                 $paginator->setCurrentPageNumber($this->_getParam('page'));
     }
               $this->view->paginator=      $paginator;
  }
```

Dentro de la vista se debería establecer el siguiente código.

```
<?php foreach($this->libros as $libro) :
echo htmlentities($this->escape($libro->id)) ;
echo htmlentities($this->escape($libro->detalle));
echo htmlentities($this->escape($libro->fecha_caducidad));
echo htmlentities($this->escape($libro->nro_tarjeta));
endforeach; ?>
```

De esta manera se optimizara el uso de redundancia en el código, generando un código claro y preciso.

Agregar: Para el agregar se requiere la toma de los valores por parte del usuario, es decir que los datos obtenidos sean extraídos por post, con el uso del componente Zend_Form, se obtendrán los datos propios del usuario para ello se debe instanciar el form con el nombre de clase asociado, como se indique en el ejemplo.

Luego se realizara la validación del Request, generado al enviar los datos mediante el método getRequest() y para obtener los datos mediante post usar el método getPost(), de este modo extraer cada uno de los datos enviados desde el formulario, mediante el método getValue('nombre_elemento') y el proceso finaliza al realizar el almacenamiento el cual debe estar implementado dentro del modelo, la sintaxis usada es $libros->addTarjeta($param1,$param2);

```
Public function agregarAction(){
                $form = new Application_Form_Tarjetacreditos();
                $this->view->form = $form;
                if ($this->getRequest()->isPost()){
                        $formData = $this->getRequest()->getPost();
                        if ($form->isValid($formData)){
                                $detalle = $form->getValue('detalle');
                                $fecha_caducidad = $form->getValue('fecha_caducidad');
                                $nro_tarjeta = $form->getValue('nro_tarjeta');
                                $id= $form->getValue('id');
                                $libros = new Application_Model_DbTable_Tarjetacreditos();
                                $libros->addTarjeta($id,$detalle,$fecha_caducidad, $nro_tarjeta);
                                        return $this->_helper->redirector('index');
                        }
                }
}
```

Dentro del Modelo se define el método addTarjeta, para luego implementa la clase Zend_Db_Table_Abstract con el método insert($data), el cual se encuentra ubicado en \library\Zend\Db\Table\Abstract.

```
public function addTarjeta($id,$detalle,$fecha_caducidad, $nro_tarjeta){
$data = array('id' => $id, 'detalle' =>$detalle,'fecha_caducidad' => $fecha_caducidad,
'nro_tarjeta' =>$nro_tarjeta,);
$this->insert($data);
}
```

Para finalizar con la parte del insertar datos dentro de la vista debe establecerse la siguiente sintaxis donde los datos del controlador sean recibidos en este caso el componente form.

```
<form  action="./agregar" method="POST">
<?php echo $this->form ;?>
</form>
```

Eliminar: Dentro del controlador definir el método borrarAction(), el cual actúa con la misma lógica, es decir trabaja mediante getRequest()->isPost() obtenido todos los datos desde el formulario.

```
public function borrarAction(){
   if ($this->getRequest()->isPost()) {
      $del = trim($this->getRequest()->getPost('del'));
      if ($del == 'Si') {
            $id = $this->_getParam('id', false);
            $libros = new Application_Model_DbTable_Tarjetacreditos();
            $libros->deleteTarjeta($id);
       }
       return $this->_helper->redirector('index'); }
   }
```

En el modelo se implementa el método delete propio de Zend_Db_Table_Abstract, según la estructura establecida, es decir siempre enviando como parámetro el criterio de eliminación como se establece en el ejemplo.

```
Public function deleteTarjeta($id){
  $this->delete('id =’. $id);
}
```

En la vista se implementa el manejo de los controles en este caso se establecerá, dos elementos submit dentro de la vista que contengan un valor para su comparación.

```
Está seguro que desea eliminar
<form action="<?php echo $this->url(array('action'=>'borrar')); ?>" method="post">
<input type="hidden" name="id" value="<?php echo $this->libro['id']; ?>" />
<input type="submit" name="del" value="Si"  />
<input type="submit" name="del" value="No" />
</form>
```

Editar: A diferencia del agregar el editar carga los datos en los elementos para que estos sean editados, para ello se utiliza el método propio de las clases _getParam() ,el cual retorna los valores asociados al filtro, luego se procede con la instanciación del modelo para llamar al método updateTarjeta definido en el modelo.

```
public function editarAction(){
$form = new Application_Form_Tarjetacreditos();
$this->view->form = $form;
if ($this->getRequest()->isPost()) {
$formData = $this->getRequest()->getPost();
if ($form->isValid($formData)) {
$id =$this->_getParam('id', false);
$detalle = $form->getValue('detalle');
$fecha_caducidad = $form->getValue('fecha_caducidad');
$nro_tarjeta = $form->getValue('nro_tarjeta');
$albums = new Application_Model_DbTable_Tarjetacreditos();
```

```
$albums->updateTarjeta($id,$detalle,$fecha_caducidad, $nro_tarjeta);
                    $this->_helper->redirector('index'); }
else { $form->populate($formData);}}
else {
$id = $this->_getParam('id', 0);
if ($id > 0) {
$albums = new Application_Model_DbTable_Tarjetacreditos();
$form->populate($albums->getTarjeta($id));}}}}
```

Dentro del modelo se define.

```
public function updateTarjeta($id,$detalle,$fecha_caducidad, $nro_tarjeta){
  $data = array('id' => $id, 'detalle' =>$detalle,
                    'fecha_caducidad' => $fecha_ caducidad, 'nro_tarjeta' =>$nro_tarjeta);
 $this->update($data, 'id = '. (int)$id);
  }
```

En la vista se define el siguiente código.

```
<form action="<?php echo $this->url(array('action'=>'editar'));?> method="post">
    <?php echo $this->form;?>
</form>
```

Cabe mencionar que los datos enviados vía get son recuperados $this->libro['id'] facilitando el manejo de los datos.

Si se pretende establecer consultas con mayor complejidad se deberá combinar tanto el armado mediante los métodos propios de los modelos como find() y su estructura antes considerada o se procederá al armado de query(), según el grado de complejidad, todo ello implementado en el controlador, es por esto que zend brinda las herramientas y es el desarrollador quien las explota a su conveniencia y necesidad.

8.5 Componentes Utilizados

El componente que más se ha utilizado dentro del desarrollo del proyecto es el Zend_Form, para la creación de formularios, al definirse un componente puedes ser instanciado como objeto se lo puede generar en la consola propia de zend, mediante la instrucción:

```
zf create form  nombreform
```

El component form se ubica dentro ..\zend\application\forms sobre el que se colocan cada uno de los form creados. Otro de los componentes utilizados es el Zend_Paginator, para su uso se deben seguir los siguientes pasos:

1.-instanciar el controlador

```
$paginator=Zend_Paginator::factory($valor);
```

2.- usar el Zend_View_Helper_PaginationControl que son ayudantes de vista luego implementar el set con la ruta de la vista del paginador.

```
public function indexAction(){
  $tarjeta = new Application_Model_DbTable_Tarjetacreditos();
  $posts=$tarjeta->fetchAll();
  $paginator=Zend_Paginator::factory($posts);
  Zend_View_Helper_PaginationControl::setDefaultViewPartial('/paginator/items.phtml');
       if ($this->_hasParam('page')){
                    $paginator->setCurrentPageNumber($this->_getParam('page'));
       }
      $this->view->paginator=$paginator;
}
```

3.-Cabe destacar que se debe antes generar el paginador en la ruta zend\application\views\scripts\ ,definir una carpeta en este caso /paginator/ítems.phtml la vista .phtml contendrá la siguiente información, sin embargo existe varios modelos definidos en zend dentro de su página oficial.

```
<?php if ($this->pageCount): ?>
<div class="paginationControl">
<!-- Previous page link -->
<?php if (isset($this->previous)): ?>
  <a href="<?php echo $this->url(array('page' => $this->previous)); ?>">&lt; Previous
</a><?php else: ?>
  <span class="disabled">&lt; Previous</span> |
<?php endif; ?>
 <!-- Numbered page links -->
<?php foreach ($this->pagesInRange as $page): ?>
  <?php if ($page != $this->current):?>
    <a href="<?php echo $this->url(array('page' => $page)); ?>"><?php echo $page;
?></a>  <?php else: ?>
    <?php echo $page; ?> |
  <?php endif; ?>
<?php endforeach ?>
<!-- Next page link -->
<?php if (isset($this->next)): ?><a href="<?php echo $this->url(array('page' =>
$this>next)); ?>"> Next &gt; </a>
<?php else: ?>
  <span class="disabled">Next &gt;</span>
<?php endif; ?></div>
<?php endif; ?>
```

8.6 Implementar Seguridad

Para que la autentificación se efectúe el Zend_Auth_Adapter_DbTable constructor, requiere una instancia de Zend_Db_Adapter_Abstract, que sirve como la conexión de base de datos al que está enlazado la instancia del adaptador de autentificación, conocido como

Zend_Db_Adapter clase básica que se utiliza para conectar el PHP de aplicaciones a una BDD, para ello se requiere la utilización de Zend_Db_Table que más que una clase es una interfaz orientada a objetos a las tablas de bases de datos. La clase base es extensible, por lo que se puede agregar lógica personalizada.
Los métodos de instancia a la clase son setTableName('users') la cual especifica la tabla a utilizar, setIdentityColumn('username'). Este es el nombre de la columna de la tabla base de datos utilizada para representar la identidad, setCredentialColumn('password').
Este es el nombre de la columna de la tabla base de datos utilizada para representar la credencial propiamente dicha.

```
private function getAuthAdapter(){
    $authAdapter=new
    Zend_Auth_Adapter_DbTable(Zend_Db_Table::getDefaultAdapter());
   $authAdapter->setTableName('users')
                    ->setIdentityColumn('username')
                    ->setCredentialColumn('password');
    return $authAdapter;
}
```

Dentro del Controlador crear el método login, el cual realizará la redirección en caso de efectuarse la autentificación mediante su comprobación realizada al instanciara el objeto Zend_Auth::getInstance ().

```
public function loginAction() {
   if (Zend_Auth::getInstance()->hasIdentity()){//redirecciona}
      $request= $this->getRequest();
      if ($request->isPost()){
         $loginForm = new Application_Form_Auth_Login();
         $this->view->form = $loginForm;
         $Form = $this->getRequest()->getPost();
        if ($loginForm->isValid($Form)){
                $loginForm = new Application_Form_Auth_Login();
                 $valor1=(String)$this->_getParam('username', false);
                $valor2=md5($this->_getParam('password', false));
                 $authAdapter=$this->getAuthAdapter();
                 $authAdapter->setIdentity($valor1)
                                ->setCredential($valor2);
                 $auth   = Zend_Auth::getInstance();
                 $result=$auth->authenticate($authAdapter);
                if ($result->isValid()) {
                        $identity=$authAdapter->getResultRowObject();
                        $authStorage=$auth->getStorage();
                        $authStorage->write($identity);
                       return $this->_helper->redirector('menu');}
```

```
else{ return $this->_helper->redirector('index'); }}
else { return $this->_helper->redirector('index'); } }}
```

8.7 Manejo de Imágenes

Para la implementar de imágenes, se debe utilizar Zend_File_Transfer, el cual ofrece un amplio soporte para carga de archivos y descargas además posee validadores de los archivos, en este caso se utiliza isImage(), además de la funcionalidad para cambiar los archivos con filtros.
Se compone de dos partes el Zend_File_Transfer_Adapter_Http hace la carga, mientras que el Zend_File_Transfer se encarga de los archivos subidos.

```
public function uploadAction(){
    $adapter = new Zend_File_Transfer_Adapter_Http();
    $adapter->setDestination('C:\wamp\www\zend\public');
if (!$adapter->receive()) {
            $messages = $adapter->getMessages();
            echo implode("\n", $messages);
    }
$upload = new Zend_File_Transfer();
    if ($upload IsImage()){
        $files = $upload->getFileInfo();
        $upload->receive();
        $names = $upload->getFileName();}
    }
```

Para mayor comodidad, se podría utilizar Zend_Form_Element_File en vez de construir el HTML manualmente dentro de las vistas.

8.8 Casos de Uso

Ver Anexo 1

8.9 Interfaz de la Aplicación

Usuario Online

Pantalla Principal

Agregar Compra

Listado de Compras

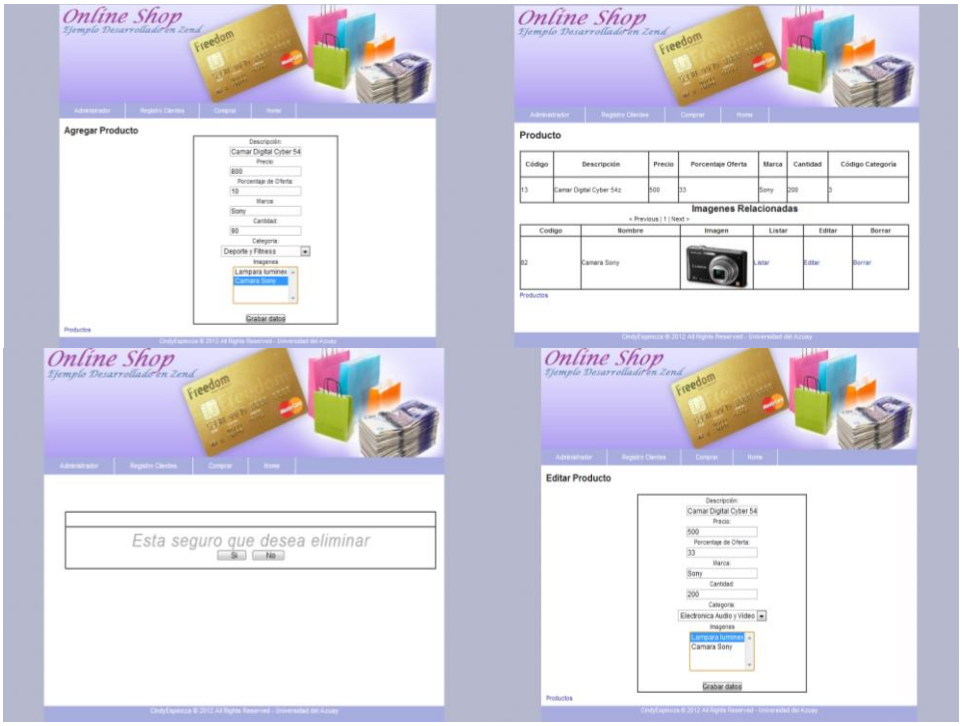
Administrador

Registro Administrador

Menú de Mantenimientos

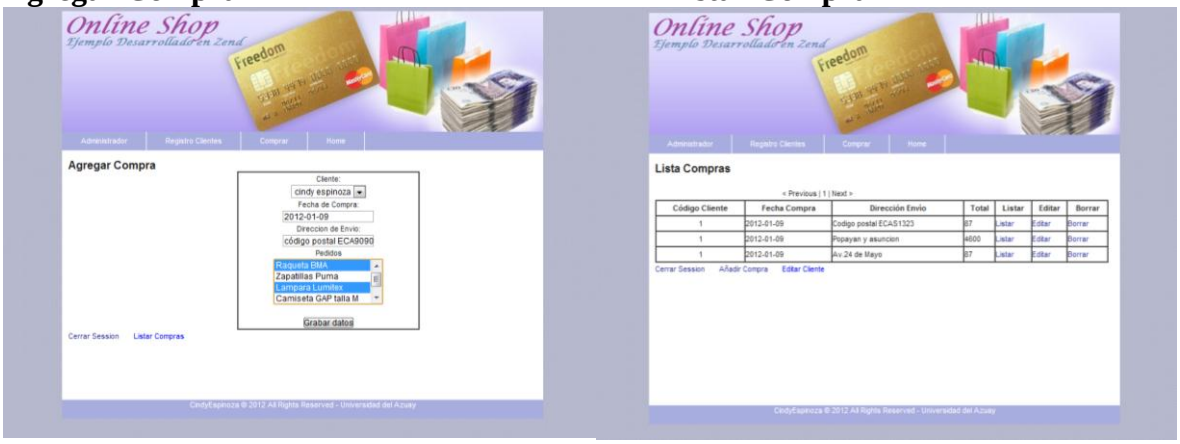
Menú Administrador

El ejemplo posee mantenimientos de cada una de las tablas, para el ejemplo se presenta la interfaz de mantenimiento de Productos.



Cliente Registrado

Agregar Compra **Listar Compra**



Conclusión

Zend posee un diseño organizado adaptable a la lógica del programador, además de ello no existen muchas convenciones, zend ofrece componentes que facilitan el desarrollo, sin embargo posee sus propios componentes y librerías que no son flexibles sino más bien es el programador, que las debe ajustar a sus necesidades, es un framework totalmente orientado a objetos por lo cual presenta todas sus ventajas de la programación orientada a objetos. Otra de sus ventajas es la cantidad de librerías existentes y las prestaciones brindadas, dentro del caso práctico se consideraron las más utilizadas y para concluir el código se encuentra totalmente distribuido, una de las desventajas con mayor impacto es la falta de Scaffolding.

CAPITULO IX
COMPARATIVA DE FRAMEWORK DE DESARROLLO

Introducción

En el siguiente capítulo se generará la evaluación comparativa del framework CakePhp,Prado,Zend mediante el estándar ISO/IEC 9126 seleccionado en el capítulo II, para luego establecer las comparaciones bajo métricas comunes, considerando los atributos de cada herramienta y los requerimientos planteados, para finalizar se pretende establecer un cuadro comparativo de resultados según la evaluación generada bajo el estándar ISO/IEC 9126 y la experiencia en la realización del caso práctico se obtendrán las ventajas y desventajas de cada herramienta.

9.1 Definición ISO/IEC 9126

La ISO/IEC 9126, describe un modelo en dos partes para la calidad de los productos software: a) calidad interna y externa y b) calidad en uso. La primera parte del modelo especifica seis características para la calidad interna y externa, que se subdividen posteriormente en subcaracterísticas. La calidad en uso es el efecto combinado para el usuario de las seis características de calidad del producto software.

9.2 Métricas utilizadas

Las escalas de medición para las métricas usadas para los requisitos de calidad se pueden dividir en categorías correspondientes, a los diferentes grados de satisfacción de los requisitos. Cada una de las métricas posee un valor en un rango de 10, que va de menor a mayor calidad, denotando que 0 no posee calidad mientras que 10 indica calidad total frente a la característica planteada.

Las siguientes métricas serán consideradas dentro de la evaluación las cuales tiene asignado un valor de medición.

- satisfactoria 10
- insatisfactoria 0

- cumple 10
- aceptable 7
- mínima 4
- inaceptable 0

- complejo 10
- simple 0

- si 10
- no 0

La obtención de los resultados se basa en establecer un promedio entre las subcaracterísticas consideradas y con los datos obtenidos generar un nuevo valor promedio por característica, cabe recalcar que el promedio posee un rango de 1 a 10, de esta manera se establecerá datos comparativos entre los frameworks antes estudiados.

9.3 Características a Evaluar

Funcionalidad

Esta característica se refiere a lo que el framework hace para satisfacer necesidades, mientras que otras características se refieren principalmente a cuando y como se satisfacen estas necesidades.

Confiabilidad

Capacidad del framework para mantener un nivel especificado de prestaciones cuando se usa bajo condiciones requeridas.

Usabilidad

Capacidad del framework para ser entendido, aprendido, usado y ser atractivo para el usuario, cuando se usa bajo condiciones especificadas.

Eficiencia

Capacidad del framework para proporcionar prestaciones apropiadas, relativas a la cantidad de recursos usados, bajo condiciones determinadas.

Mantenibilidad

Capacidad del framework para ser modificado. Las modificaciones podrían incluir correcciones, mejoras o adaptación del software a cambios en el entorno, y requisitos y especificaciones funcionales.

Portabilidad

Capacidad del producto framework para ser para ser adaptado a diferentes entornos especificados.

9.4 Aplicación del Modelo de Calidad para CakePhp

Framework CakePHP					
Características / Subcaracterísticas	Attribute Level 1	Attribute Level 2	Métrica	Resultado	Total
1 Funcionalidad					
1 Idoneidad					
1 Idoniedad Reportes					
	1 Generación de Reportes				
		1.1 Manejo de Componentes	(si/no)	10	
		Utiliza Paginación	(satisfactoria, insatisfactoria)	10	
		Calidad del Reporte	(satisfactoria, insatisfactoria)	10	
		Flexible a cambios	(cumple, aceptable, minima, inaceptable)	7	
		Facilidad de Implementación	(satisfactoria, insatisfactoria)	10	
		Genera varios Formatos	(si/no)	10	
		1.2 Generación Automática	(si/no)	10	
	2 Reporte de Error				
		2.1 Generación Log de Errores	(si/no)	10	
		Log Preciso	(cumple, aceptable, minima, inaceptable)	4	
		2.2 Configuración reportes de error	(complejo,simple)	10	
		Dificultad de implementación	(complejo,simple)	10	9,18
2 Idoniedad Alertas					
	1 Implementar Mensajes				
		1.1 Avisos de Error	(si/no)	10	
		1.2 Avisos de Validación	(cumple, aceptable, minima, inaceptable)	10	
		1.3 Redirección	(cumple, aceptable, minima, inaceptable)	10	
		1.4 Personalización de Mensajes	(satisfactoria, insatisfactoria)	10	
		1.5 Librerias y Plugins	(cumple, aceptable, minima, inaceptable)	0	
		1.6 Mensajes de Confirmación	(si/no)	10	
	2 Comentarios en Clases				
		2.1 Comentarios	(si/no)	10	
	3 Mensajes Correo				
		3.1 Integración correo electrónico y Framework	(cumple, aceptable, minima, inaceptable)	4	
		3.2 Enviar y Recibir Mensajes	(cumple, aceptable, minima, inaceptable)	4	
		3.3 Filtrar envio	(cumple, aceptable, minima, inaceptable)	4	
		3.4 Importar contactos	(satisfactoria, insatisfactoria)	0	
	4 Generar Mensajes comunidad online				
		4.1 Envio y Recepción de Mensajes	(cumple, aceptable, minima, inaceptable)	4	
		4.2 Foros de ayuda	(cumple, aceptable, minima, inaceptable)	7	6,38
2 Precisión					
1 Verificabilidad					
	1 Control de Versiones				
		1.1 Información respaldo	(cumple, aceptable, minima, inaceptable)	4	
		1.2 Integrar mejoras	(cumple, aceptable, minima, inaceptable)	4	
		1.3 Notificación de Mejoras	(cumple, aceptable, minima, inaceptable)	4	
	2 Capacidad Registro Comunidades				
		2.1 Generación de Información	(satisfactoria, insatisfactoria)	0	
	3 Componentes Estables				
		3.1 Cambios segun mejoras	(cumple, aceptable, minima, inaceptable)	4	
		3.2 Requieren modificación	(complejo,simple)	10	4,33
3 Interoperabilidad					
1 Interoperabilidad directa					
	1 Uso de Protocolos				
		1.1 http(s), ftp(s), file, news, y gopher.	(cumple, aceptable, minima, inaceptable)	10	
	2 Integracion API's desarrollo con IDE				
		2.1 IDE reconoce API's	(cumple, aceptable, minima, inaceptable)	0	
		2.2 Actualización extensiones	(satisfactoria, insatisfactoria)	0	
		2.3 Documentación Existente	(cumple, aceptable, minima, inaceptable)	7	4,25
4 Seguridad					
1 Aplicación de seguridad					
	1 Manejo de Seguridad				
		1.1 Implementa Componentes de Seguridad	(cumple, aceptable, minima, inaceptable)	10	
		1.2 Nivel de Seguridad	(cumple, aceptable, minima, inaceptable)	7	
		1.3 Manejo de Autentificación	(satisfactoria, insatisfactoria)	10	
	2 Mejoras en la Seguridad				
		2.1 Manejo de Sessiones	(cumple, aceptable, minima, inaceptable)	10	
		2.2 Cookies	(si/no)	10	
		2.3 Asignación de Roles	(si/no)	10	
	3 Envio de Datos				
		3.1 Seguridad al transferir datos	(cumple, aceptable, minima, inaceptable)	4	8,71
5 Cumplimiento en el Funcionamiento					
	1 Covenciones funcionamiento				
		1.1 Convenciones Base de Datos	(complejo,simple)	0	
		1.2 Convenciones Vistas	(complejo,simple)	0	
		1.3 Convenciones Clases	(complejo,simple)	10	
		1.4 Convenciones Modelo	(complejo,simple)	0	
		1.5 Convenciones Controlador	(complejo,simple)	0	2,00 6,04
2 Confiabilidad					
1 Madurez					
1 Historicos del producto					
		1.1 Aceptación en el Mercado	(cumple, aceptable, minima, inaceptable)	7	
	2 Deteccion de fallas y correccion				
		2.1 Fallas detectadas por versión	(complejo,simple)	10	
		2.2 Fallas corregidas por parche	(cumple, aceptable, minima, inaceptable)	7	8,00
2 Tolerancia a fallos					
1 Nivel de tolerancia					
		1.1 Presentación de errores	(cumple, aceptable, minima, inaceptable)	10	
		1.2 Parametrizacion valores por defecto	(si/no)	10	10,00
3 Capacidad de recuperacion					5,17
1 Recuperabilidad del sistema					
		1.1 Recuperación en función de error	(cumple, aceptable, minima, inaceptable)	10	
		1.2 Log Errores	(satisfactoria, insatisfactoria)	10	
		1.3 Tiempo de caida	(cumple, aceptable, minima, inaceptable)	7	9,00
2 Recuperabilidad de datos					
	1 Datos del sistema				
		1.1 Facilidades de backup y recuperacion	(cumple, aceptable, minima, inaceptable)	4	
		1.2 Tipo de backup	(satisfactoria, insatisfactoria)	0	
		1.3 Configuración backup	(complejo,simple)	0	1,33 7,72

3 Usabilidad							
Usability	1 Capacidad para ser entendido						4,58
	1 Comprensibilidad de interfaz						
			1.1 Clara Navegabilidad	(cumple, aceptable, minima, inaceptable)	10		
			1.2 Fácil implementación	(satisfactoria, insatisfactoria)	10		
			1.3 Soporta varios Idiomas	(cumple, aceptable, minima, inaceptable)	0	6,67	
	2 Estructura global						
			2.1 Estandarización de las interfaces	(complejo,simple)	10		
			2.2 Componentes Manejo de interfaz	(satisfactoria, insatisfactoria)	0		
			2.3 Manejo de Templates	(complejo,simple)	0		
			2.4 Implementación de Interfaz	(si/no)	0	2,50	
	2 Capacidad para ser aprendido						6
	1 Entrenamiento						
			1.1 Curva de aprendizaje	rango(1..10)	8		
			1.2 Ayudas Online,videos etc	rango(1..10)	5	6,5	
	2 Documentación						
			2.1 Documentacion proporcionada	(cumple, aceptable, minima, inaceptable)	7		
			2.2 Documentacion y manuales	(cumple, aceptable, minima, inaceptable)	7		
			2.3 Foros y Comunidades Activas	(cumple, aceptable, minima, inaceptable)	4		
			2.4 Demos	(cumple, aceptable, minima, inaceptable)	4	5,5	
	3 Capacidad para ser manipulado						4,52
	1 Sistema hecho a la medida						
		1 Flexibilidad a cambios					
			1.1 Framework Flexible a cambios	(cumple, aceptable, minima, inaceptable)	7		
			1.2 Desarrollo organizado y distribuido	(satisfactoria, insatisfactoria)	10		
		2 Manejo de Recursos					
			2.1 Programación Módular	(cumple, aceptable, minima, inaceptable)	10		
			2.2 Soporta grupos de desarrollo	(cumple, aceptable, minima, inaceptable)	10		
			2.3 Amplio número de recursos	(si/no)	10		
		3 Evaluación de estructura interna					
			3.1 Generar modificaciones a nivel interno	(complejo,simple)	0		
			3.2 Parámetros de configuración	(cumple, aceptable, minima, inaceptable)	7	7,71	
	2 Apariencia hecha a medida						
		1 Soporte Plantillas					
			1.1 Cambio de Estilos	(si/no)	0		
			1.2 Configuración de plantillas	(complejo,simple)	0		
			1.2 Manejo Componentes de diseño	(cumple, aceptable, minima, inaceptable)	4	1,33	5,04
4 Eficiencia							
	2 Utilizacion de recursos						
	1 Implementación						
		1 Recursos Software requeridos					
			1.1 Cantidad de Recursos requerida	rango(1..10)	8		
			1.2 Desarrollo Organizado	rango(1..10)	9		
			1.3 Modelo Vista Controlador	(si/no)	10		
			1.2 Cantidad de Código	rango(1..10)	8	8,75	8,75
5 Mantenibilidad							
ainability	1 Analizabilidad						5,33
	1 Datos analizables						
		1 Testing					
			1.1 Implementa testing	(si/no)	0		
			1.2 Manejo de errores personalizables	(si/no)	10	5	
	2 Analisis de capacidades						
		1 Componentes,plugins,elementos,controles probados					
			1.1 Maneja versiones estables	(cumple,aceptable,minima,inaceptable)	7	5,67	
	2 Variabilidad						5,92
	1 Estorno de desarrollo						
		1 Bibliotecas API					
			1.1 Actualización de bibliotecas	(si/no)	10		
			1.2 Documentación de mejora	(cumple,aceptable,minimo,inaceptable)	7		
			1.2 Versiones estables Biblioteca	(rango 1..10)	6		
		2 Parches o versiones de mejora					
			2.1 Implementar parches	(si/no)	10		
			2.2 Varios entornos de desarrollo	(si/no)	10		
			2.2 Configuración de parches	(complejo,simple)	0	7,17	
	2 Apariencia hecha a medida						
		1 Soporte Plantillas					
			1.1 Cambio de Estilos	(si/no)	0		
			1.2 Configuración de plantillas	(complejo,simple)	0		
			1.2 Manejo Componentes de diseño	(cumple, aceptable, minima, inaceptable)	4	1,33	5,04
4 Eficiencia							
	2 Utilizacion de recursos						
	1 Implementación						
		1 Recursos Software requeridos					
			1.1 Cantidad de Recursos requerida	rango(1..10)	8		
			1.2 Desarrollo Organizado	rango(1..10)	9		
			1.3 Modelo Vista Controlador	(si/no)	10		
			1.2 Cantidad de Código	rango(1..10)	8	8,75	8,75
5 Mantenibilidad							
ainability	1 Analizabilidad						5,33
	1 Datos analizables						
		1 Testing					
			1.1 Implementa testing	(si/no)	0		
			1.2 Manejo de errores personalizables	(si/no)	10	5	
	2 Analisis de capacidades						
		1 Componentes,plugins,elementos,controles probados					
			1.1 Maneja versiones estables	(cumple,aceptable,minima,inaceptable)	7	5,67	
	2 Variabilidad						5,92
	1 Estorno de desarrollo						
		1 Bibliotecas API					
			1.1 Actualización de bibliotecas	(si/no)	10		
			1.2 Documentación de mejora	(cumple,aceptable,minimo,inaceptable)	7		
			1.2 Versiones estables Biblioteca	(rango 1..10)	6		
		2 Parches o versiones de mejora					
			2.1 Implementar parches	(si/no)	10		
			2.2 Varios entornos de desarrollo	(si/no)	10		
			2.2 Configuración de parches	(complejo,simple)	0	7,17	

Mainta	2 Desarrollo de documentación						
			1.1 Documentación biblioteca API	(cumple,aceptable,minimo,inaceptable)	10		
			1.2 Documentación componentes	(cumple,aceptable,minimo,inaceptable)	4		
			1.3 Documentación de plugins	(cumple,aceptable,minimo,inaceptable)	4		
			1.4 Documentación de controles	(cumple,aceptable,minimo,inaceptable)	0		
			1.5 Documentación de elementos	(cumple,aceptable,minimo,inaceptable)	0		
			1.6 Documentación versiones	(cumple,aceptable,minimo,inaceptable)	10	4,67	
	3 Estabilidad					5,75	
	2 Estabilidad Operacional						
		1 Tiempo Promedio real de las versiones		(rango 1..10)	3		
		2 Componentes mejoras en cache		(cumple,aceptable,minimo,inaceptable)	10		
		3 Volumen de cache		(cumple,aceptable,minimo,inaceptable)	7		
		4 Reporte Parches cambios		(rango 1..10)	3	5,75	
	4 Capacidad de Prueba					3,33	
		1 Implementar correcciones					
			1.1 Tiempo para realizar correcciones	(rango 1..10)	5		
			1.2 Facilidad de implementación	(rango 1..10)	5		
			1.3 componentes,plugins,controles soporta cambios	(complejo,simple)	0	3,33	5,08
6 Portabilidad							
	1 Adaptabilidad					10	
		1 Reutilizar el código existente					
			1.1 Reutilización código propio	(rango 1..10)	10		
			1.2 Reutilización código existente	(rango 1..10)	10	10	
		2 Estructura					
			2.1 Modelo Vista controlador	(si/no)	10	10	
	2 Instalación					9,67	
	1 Configuración Instalación						
		1 Volumen del archivo de instalación	10.3 Mb (proyecto generado)	(rango 1..10)	10		
		2 Fácil Instalación		(complej, simple)	10	10	
	2 Soportes de Instalación						
		1 Documentación y manuales		(cumple, aceptable, minima, inaceptable)	10		
		2 Demos		(cumple, aceptable, minima, inaceptable)	7		
		3 Ayuda Online		(cumple, aceptable, minima, inaceptable)	10	9	
	3 Compatibilidad con Sistemas Operativos						
		1 Utilizado en varios sistemas operativos		(si/no)	10	10	
	3 Coexistencia					7,67	
	1 Utiliza multiples base de datos						
		1 Utiliza multiples base de datos		(si/no)	10		
		2 Archivos de configuración		(si/no)	10		
		3 Multiples convenciones según la base		(rango 1..10)	5		
		4 Múltiples protocolos		(si/no)	10		
		5 Utiliza Parches		(cumple,aceptable,minima,inaceptable)	4		
		6 Nivel de Configuración		(cumple, aceptable, minima, inaceptable)	7	7,67	9,1

9.5 Aplicación del Modelo de Calidad para Prado

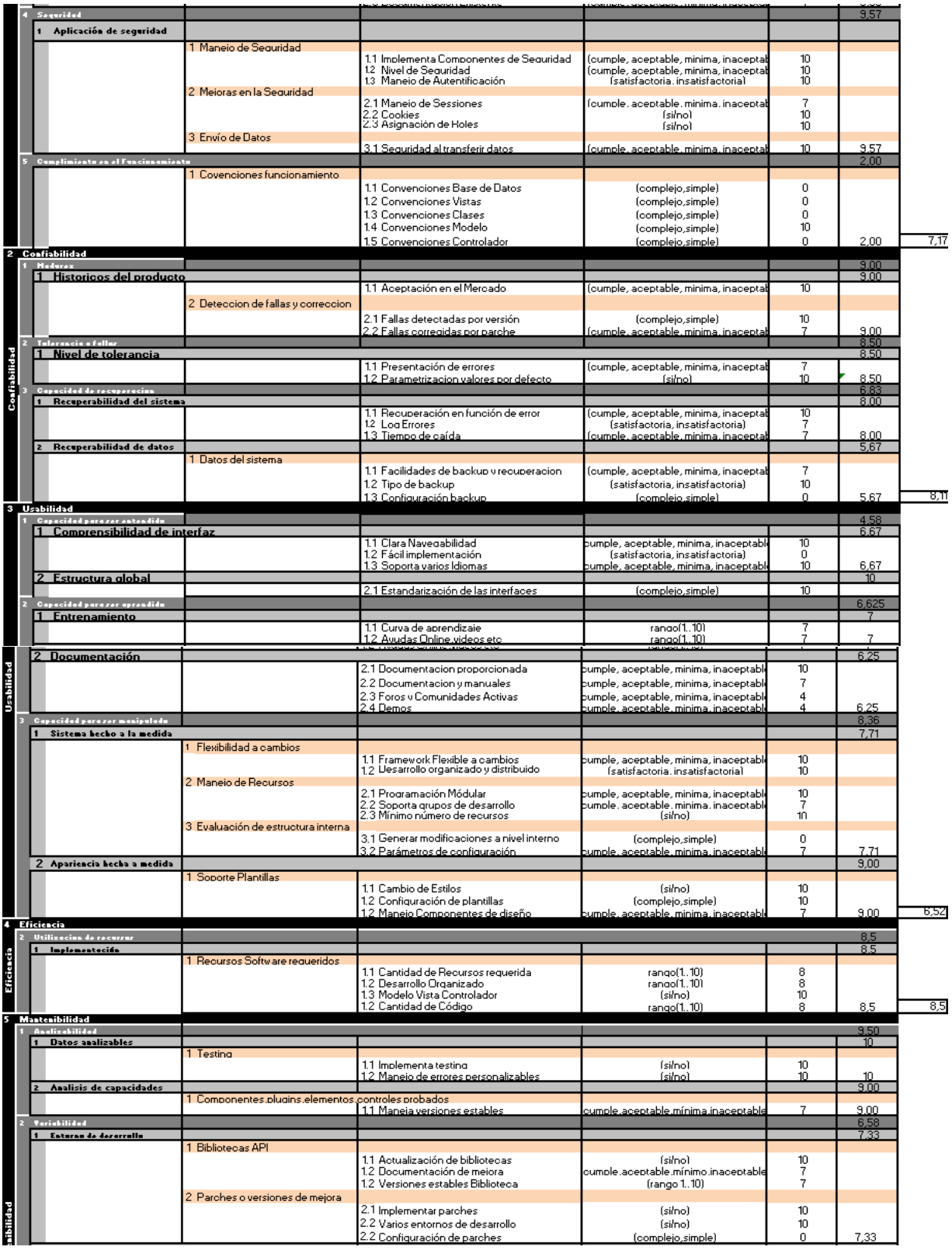
Framework CakePHP					
Características / Subcaracterísti	Attribute Level 1	Attribute Level 2	Métrica	Resultado	Total
1 Funcionalidad					
1 Idoneidad					6,49
1 Idoneidad Reportes					5,82
	1 Generación de Reportes				
		1.1 Maneio de Componentes	(si/no)	10	
		Utiliza Paginación	(satisfactoria, insatisfactoria)	0	
		Calidad del Reporte	(satisfactoria, insatisfactoria)	10	
		Flexible a cambios	(cumple, aceptable, minima, inaceptab	10	
		Facilidad de Implementación	(satisfactoria, insatisfactoria)	0	
		Genera varios Formatos	(si/no)	10	
		1.2 Generación Automática	(si/no)	10	
	2 Reporte de Error				
		2.1 Generación Log de Errores	(si/no)	0	
		Log Preciso	(cumple, aceptable, minima, inaceptab	4	
		2.2 Configuración reportes de error	(complejo,simple)	10	
		Dificultad de implementación	(complejo,simple)	0	5,82
2 Idoneidad Alertas					7,15
	1 Implementar Mensajes				
		1.1 Avisos de Error	(si/no)	10	
		1.2 Avisos de Validación	(cumple, aceptable, minima, inaceptab	7	
		1.3 Redirección	(cumple, aceptable, minima, inaceptab	10	
		1.4 Personalización de Mensajes	(satisfactoria, insatisfactoria)	10	
		1.5 Librerias y Plugins	(cumple, aceptable, minima, inaceptab	7	
		1.6 Mensajes de Confirmación	(si/no)	10	
	2 Comentarios en Clases				
		2.1 Comentarios	(si/no)	10	
	3 Mensajes Correo				
		3.1 Integración correo electrónico y Framework	(cumple, aceptable, minima, inaceptab	7	
		3.2 Enviar y Recibir Mensajes	(cumple, aceptable, minima, inaceptab	4	
		3.3 Filtrar envio	(cumple, aceptable, minima, inaceptab	4	
		3.4 Importar contactos	(satisfactoria, insatisfactoria)	0	
	4 Generar Mensajes comunidad online				
		4.1 Envio y Recepción de Mensajes	(cumple, aceptable, minima, inaceptab	7	
		4.2 Foros de ayuda	(cumple, aceptable, minima, inaceptab	7	7,15
2 Precisión					7,00
1 Verificabilidad					7,00
	1 Control de Versiones				
		1.1 Información respaldo	(cumple, aceptable, minima, inaceptab	7	
		1.2 Integrar mejoras	(cumple, aceptable, minima, inaceptab	4	
		1.3 Notificación de Mejoras	(cumple, aceptable, minima, inaceptab	4	
	2 Capacidad Registro Comunidades				
		2.1 Generación de Información	(satisfactoria, insatisfactoria)	10	
	3 Componentes Estables				
		3.1 Cambios según mejoras	(cumple, aceptable, minima, inaceptab	7	
		3.2 Requieren modificación	(complejo,simple)	10	7,00
3 Interoperabilidad					8,50
1 Interoperabilidad directa					
	1 Uso de Protocolos				
		1.1 http(s), ftp(s), file, news, y gopher.	(cumple, aceptable, minima, inaceptab	7	
	2 Integracion API's desarrollo con IDE				
		2.1 IDE reconoce API's	(cumple, aceptable, minima, inaceptab	10	
		2.2 Actualización extensiones	(satisfactoria, insatisfactoria)	10	
		2.3 Documentación Existente	(cumple, aceptable, minima, inaceptab	7	8,50

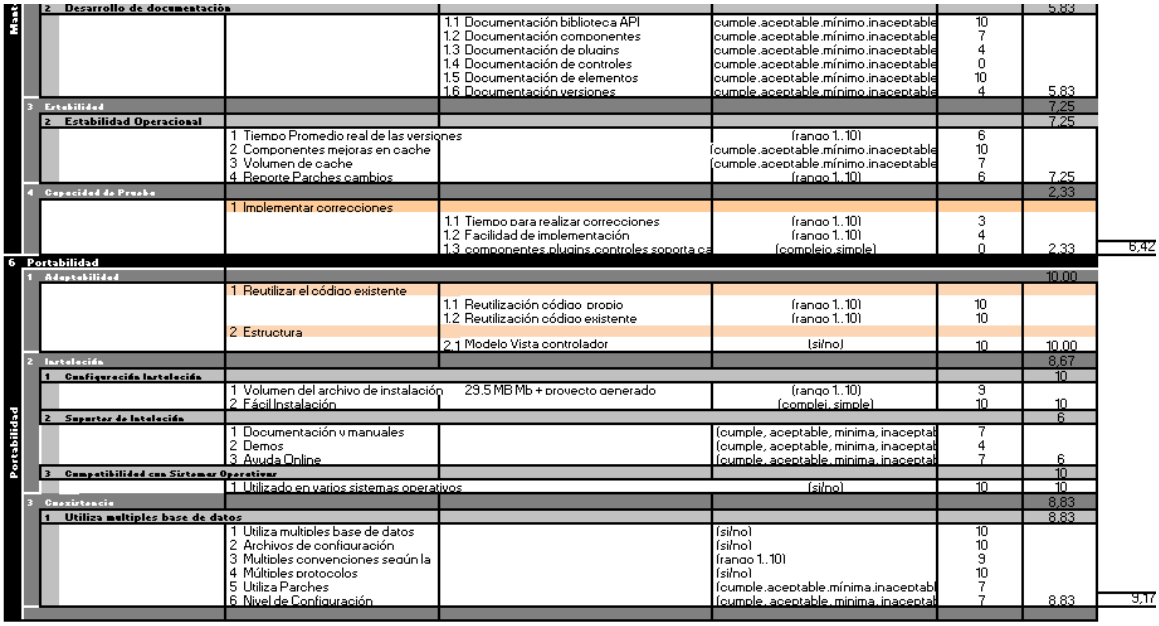
	4 Seguridad					8,29	
	1 Aplicación de seguridad						
		1 Manejo de Seguridad					
			1.1 Implementa Componentes de Seguridad	(cumple, aceptable, minima, inaceptab	10		
			1.2 Nivel de Seguridad	(cumple, aceptable, minima, inaceptab	7		
			1.3 Maneio de Autentificación	(satisfactoria, insatisfactoria)	10		
		2 Mejoras en la Seguridad					
			2.1 Maneio de Sessiones	(cumple, aceptable, minima, inaceptab	4		
			2.2 Cookies	(si/no)	10		
			2.3 Asignación de Holes	(si/no)	10		
		3 Envío de Datos					
			3.1 Seguridad al transferir datos	(cumple, aceptable, minima, inaceptab	7	8,29	
	5 Cumplimiento en el Funcionamiento					2,00	
		1 Covenciones funcionamiento					
			1.1 Convenciones Base de Datos	(complejo,simple)	0		
			1.2 Convenciones Vistas	(complejo,simple)	0		
			1.3 Convenciones Clases	(complejo,simple)	0		
			1.4 Convenciones Modelo	(complejo,simple)	0		
			1.5 Convenciones Controlador	(complejo,simple)	10	2,00	6,45
2 Confiabilidad							
	1 Madurez					7,00	
	1 Historicos del producto					7,00	
			1.1 Aceptación en el Mercado	(cumple, aceptable, minima, inaceptab	4		
		2 Deteccion de fallas y correccion					
			2.1 Fallas detectadas por versión	(complejo,simple)	10		
			2.2 Fallas corregidas por parche	(cumple, aceptable, minima, inaceptab	7	7,00	
	2 Tolerancia a fallar					8,50	
	1 Nivel de tolerancia					8,50	
			1.1 Presentación de errores	(cumple, aceptable, minima, inaceptab	7		
			1.2 Parametrizacion valores por defecto	(si/no)	10	8,50	
	3 Capacidad de recuperacion					5,17	
	1 Recuperabilidad del sistema						
			1.1 Recuperación en función de error	(cumple, aceptable, minima, inaceptab	10		
			1.2 Log Errores	(satisfactoria, insatisfactoria)	10		
			1.3 Tiempo de caída	(cumple, aceptable, minima, inaceptab	7	9,00	
	2 Recuperabilidad de datos					1,33	
		1 Datos del sistema					
			1.1 Facilidades de backup y recuperacion	(cumple, aceptable, minima, inaceptab	4		
			1.2 Tipo de backup	(satisfactoria, insatisfactoria)	0		
			1.3 Configuración backup	(complejo,simple)	0	1,33	6,89
3 Usabilidad							
	1 Capacidad para ser entendido					5,00	
	1 Comprensibilidad de interfaz					10,00	
			1.1 Clara Navegabilidad	cumple, aceptable, minima, inaceptable	10		
			1.2 Fácil implementación	(satisfactoria, insatisfactoria)	10		
			1.3 Soporta varios Idiomas	cumple, aceptable, minima, inaceptable	10	10,00	
	2 Estructura global					0	
			2.1 Estandarización de las interfaces	(complejo,simple)	0		
	2 Capacidad para ser aprendido					7,875	
	1 Entrenamiento						
			1.1 Curva de aprendizaje	rango(1..10)	9		
			1.2 Ayudas Online,videos etc	rango(1..10)	8	8,5	
	2 Documentación						
			2.1 Documentacion proporcionada	cumple, aceptable, minima, inaceptable	8		
			2.2 Documentacion y manuales	cumple, aceptable, minima, inaceptable	7		
			2.3 Foros y Comunidades Activas	cumple, aceptable, minima, inaceptable	7		
			2.4 Demos	cumple, aceptable, minima, inaceptable	7	7,25	
	3 Capacidad para ser manipulado					5,33	
	1 Sistema hecho a la medida						
		1 Flexibilidad a cambios					
			1.1 Framework Flexible a cambios	cumple, aceptable, minima, inaceptable	10		
			1.2 Desarrollo organizado y distribuido	(satisfactoria, insatisfactoria)	0		
		2 Maneio de Recursos					
			2.1 Programación Módular	cumple, aceptable, minima, inaceptable	4		
			2.2 Soporta grupos de desarrollo	cumple, aceptable, minima, inaceptable	7		
			2.3 Amplio número de recursos	(si/no)	10		
		3 Evaluación de estructura interna					
			3.1 Generar modificaciones a nivel interno	(complejo,simple)	0		
			3.2 Parámetros de configuración	cumple, aceptable, minima, inaceptable	4	5,00	
	2 Apariencia hecha a medida						
		1 Soporte Plantillas					
			1.1 Cambio de Estilos	(si/no)	10		
			1.2 Configuración de plantillas	(complejo,simple)	0		
			1.2 Maneio Componentes de diseño	cumple, aceptable, minima, inaceptable	7	5,67	6,07
4 Eficiencia							
	2 Utilizacion de recursos					3	
	1 Implementación						
		1 Recursos Software requeridos					
			1.1 Cantidad de Recursos requerida	rango(1..10)	5		
			1.2 Desarrollo Organizado	rango(1..10)	5		
			1.3 Modelo Vista Controlador	(si/no)	0		
			1.2 Cantidad de Código	rango(1..10)	2	3	3
5 Mantenibilidad							
	1 Analizabilidad					9,00	
	1 Datos analizables					10	
		1 Testing					
			1.1 Implementa testing	(si/no)	10		
			1.2 Maneio de errores personalizables	(si/no)	10	10	
	2 Analisis de capacidades					8,00	
		1 Componentes,plugins,elementos,controles probados					
			1.1 Maneja versiones estables	cumple,aceptable,mínima,inaceptable	4	8,00	
	2 Variabilidad					6,58	
	1 Entorno de desarrollo					7,50	
		1 Bibliotecas API					
			1.1 Actualización de bibliotecas	(si/no)	10		
			1.2 Documentación de mejora	cumple,aceptable,mínimo,inaceptable	7		
			1.2 Versiones estables Biblioteca	(rango 1..10)	8		
		2 Parches o versiones de mejora					
			2.1 Implementar parches	(si/no)	10		
			2.2 Varios entornos de desarrollo	(si/no)	10		
			2.2 Configuración de parches	(complejo,simple)	0	7,50	

Mant	2 Desarrollo de documentación					5,67	
			1.1 Documentación biblioteca API	cumple.aceptable.mínimo.inaceptable	10		
			1.2 Documentación componentes	cumple.aceptable.mínimo.inaceptable	0		
			1.3 Documentación de plugins	cumple.aceptable.mínimo.inaceptable	7		
			1.4 Documentación de controles	cumple.aceptable.mínimo.inaceptable	10		
			1.5 Documentación de elementos	cumple.aceptable.mínimo.inaceptable	0		
			1.6 Documentación versiones	cumple.aceptable.mínimo.inaceptable	7	5,67	
	3 Estabilidad					5,75	
	2 Estabilidad Operacional					5,75	
		1 Tiempo Promedio real de las versiones		(rango 1..10)	4		
		2 Componentes mejoras en cache		(cumple.aceptable.mínimo.inaceptable	10		
		3 Volumen de cache		(cumple.aceptable.mínimo.inaceptable	5		
		4 Reporte Parches cambios		(rango 1..10)	4	5,75	
	4 Capacidad de Prueba					3,00	
		1 Implementar correcciones					
			1.1 Tiempo para realizar correcciones	(rango 1..10)	4		
			1.2 Facilidad de implementación	(rango 1..10)	5		
			1.3 componentes,plugins,controles soporta ca	(complejo,simple)	0	3,00	6,08
6 Portabilidad							
	1 Adaptabilidad					6,67	
		1 Reutilizar el código existente					
			1.1 Reutilización código propio	(rango 1..10)	10		
			1.2 Reutilización código existente	(rango 1..10)	10		
		2 Estructura					
			2.1 Modelo Vista controlador	(si/no)	0	6,67	
	2 Instalación					9,67	
	1 Configuración Instalación					10	
		1 Volumen del archivo de instalación	70 Mb (proyecto generado)	(rango 1..10)	8		
		2 Fácil Instalación		(complej, simple)	10	10	
	2 Soporte de Instalación					9	
		1 Documentación y manuales		(cumple, aceptable, minima, inaceptab	7		
		2 Demos		(cumple, aceptable, minima, inaceptab	10		
		3 Ayuda Online		(cumple, aceptable, minima, inaceptab	10	9	
	3 Compatibilidad con Sistemas Operativos					10	
		1 Utilizado en varios sistemas operativos		(si/no)	10	10	
	3 Coexistencia					8,17	
	1 Utiliza multiples base de datos					8,17	
		1 Utiliza multiples base de datos		(si/no)	10		
		2 Archivos de configuración		(si/no)	10		
		3 Multiples convenciones según la		(rango 1..10)	8		
		4 Múltiples protocolos		(si/no)	10		
		5 Utiliza Parches		(cumple.aceptable.mínima.inaceptabl	7		
		6 Nivel de Configuración		(cumple, aceptable, minima, inaceptab	4	8,17	8,17

9.6 Aplicación del Modelo de Calidad para Zend

Framework CakePHP					
Características / Subcaracterísti	Attribute Level 1	Attribute Level 2	Métrica	Resultado	Total
1 Funcionalidad					
1 Idoneidad					8,26
1 Idoneidad Reportes					8,82
	1 Generación de Reportes				
		1.1 Manejo de Componentes	(si/no)	10	
		Utiliza Paginación	(satisfactoria, insatisfactoria)	10	
		Calidad del Reporte	(satisfactoria, insatisfactoria)	10	
		Flexible a cambios	(cumple, aceptable, minima, inaceptab	10	
		Facilidad de Implementación	(satisfactoria, insatisfactoria)	10	
		Genera varios Formatos	(si/no)	10	
		1.2 Generación Automática	(si/no)	10	
	2 Reporte de Error				
		2.1 Generación Log de Errores	(si/no)	10	
		Log Preciso	(cumple, aceptable, minima, inaceptab	7	
		2.2 Configuración reportes de error	(complejo,simple)	10	
		Dificultad de implementación	(complejo,simple)	0	8,82
2 Idoneidad Alertas					7,69
	1 Implementar Mensajes				
		1.1 Avisos de Error	(si/no)	10	
		1.2 Avisos de Validación	(cumple, aceptable, minima, inaceptab	7	
		1.3 Redirección	(cumple, aceptable, minima, inaceptab	10	
		1.4 Personalización de Mensajes	(satisfactoria, insatisfactoria)	10	
		1.5 Librerias y Plugins	(cumple. aceptable. minima. inaceptab	7	
		1.6 Mensajes de Confirmación	(si/no)	10	
	2 Comentarios en Clases				
		2.1 Comentarios	(si/no)	10	
	3 Mensajes Correo				
		3.1 Integración correo electrónico y Framework	(cumple, aceptable, minima, inaceptab	7	
		3.2 Enviar y Recibir Mensajes	(cumple, aceptable, minima, inaceptab	7	
		3.3 Filtrar envio	(cumple, aceptable, minima, inaceptab	4	
		3.4 Importar contactos	(satisfactoria, insatisfactoria)	7	
	4 Generar Mensajes comunidad online				
		4.1 Envio y Recepción de Mensajes	(cumple, aceptable, minima, inaceptab	7	
		4.2 Foros de ayuda	(cumple, aceptable, minima, inaceptab	4	7,69
2 Precisión					7,50
1 Verificabilidad					7,50
	1 Control de Versiones				
		1.1 Información respaldo	(cumple, aceptable, minima, inaceptab	7	
		1.2 Integrar mejoras	(cumple, aceptable, minima, inaceptab	7	
		1.3 Notificación de Mejoras	(cumple. aceptable. minima. inaceptab	4	
	2 Capacidad Registro Comunidades				
		2.1 Generación de Información	(satisfactoria, insatisfactoria)	10	
	3 Componentes Estables				
		3.1 Cambios según mejoras	(cumple, aceptable, minima, inaceptab	7	
		3.2 Requieren modificación	(complejo,simple)	10	7,50
3 Interoperabilidad					8,50
1 Interoperabilidad directa					
	1 Uso de Protocolos				
		1.1 http(s), ftp(s), file, news, y gopher.	(cumple. aceptable. minima. inaceptab	10	
	2 Integracion API's desarrollo con IDE				
		2.1 IDE reconoce API's	(cumple, aceptable, minima, inaceptab	7	
		2.2 Actualización extensiones	(satisfactoria, insatisfactoria)	10	
		2.3 Documentación Existente	(cumple, aceptable, minima, inaceptab	7	8,50





Mant.	2 Desarrollo de documentación					5.83
			1.1 Documentación biblioteca API	cumple.aceptable.mínimo.inaceptable	10	
			1.2 Documentación componentes	cumple.aceptable.mínimo.inaceptable	7	
			1.3 Documentación de plugins	cumple.aceptable.mínimo.inaceptable	4	
			1.4 Documentación de controles	cumple.aceptable.mínimo.inaceptable	0	
			1.5 Documentación de elementos	cumple.aceptable.mínimo.inaceptable	10	
			1.6 Documentación versiones	cumple.aceptable.mínimo.inaceptable	4	5.83
	3 Estabilidad					7.25
	2 Estabilidad Operacional					7.25
		1 Tiempo Promedio real de las versiones		(rango 1..10)	6	
		2 Componentes mejoras en cache		(cumple.aceptable.mínimo.inaceptable	10	
		3 Volumen de cache		(cumple.aceptable.mínimo.inaceptable	7	
		4 Reporte Parches cambios		(rango 1..10)	6	7.25
	4 Capacidad de Prueba					2.33
		1 Implementar correcciones				
			1.1 Tiempo para realizar correcciones	(rango 1..10)	3	
			1.2 Facilidad de implementación	(rango 1..10)	4	
			1.3 componentes plugins controles soporta ca	(complejo, simple)	0	2.33
6 Portabilidad						6.42
	1 Adaptabilidad					10.00
		1 Reutilizar el código existente				
			1.1 Reutilización código propio	(rango 1..10)	10	
			1.2 Reutilización código existente	(rango 1..10)	10	
		2 Estructura				
			2.1 Modelo Vista controlador	(si/no)	10	10.00
	2 Instalación					8.67
	1 Configuración Instalación					10
		1 Volumen del archivo de instalación	29.5 MB Mb + proyecto generado	(rango 1..10)	9	
		2 Fácil Instalación		(complej, simple)	10	10
	2 Soporte de Instalación					6
		1 Documentación y manuales		(cumple, aceptable, mínima, inaceptat	7	
		2 Demos		(cumple, aceptable, mínima, inaceptat	4	
		3 Ayuda Online		(cumple, aceptable, mínima, inaceptat	7	6
	3 Compatibilidad con Sistemas Operativos					10
		1 Utilizado en varios sistemas operativos		(si/no)	10	10
	3 Coexistencia					8.83
	1 Utiliza multiples base de datos					8.83
		1 Utiliza multiples base de datos		(si/no)	10	
		2 Archivos de configuración		(si/no)	10	
		3 Multiples convenciones según la		(rango 1..10)	9	
		4 Múltiples protocolos		(si/no)	10	
		5 Utiliza Parches		(cumple.aceptable.mínima.inaceptabl	7	
		6 Nivel de Configuración		(cumple, aceptable, mínima, inaceptat	7	8.83
						9.17

9.7 Cuadro comparativo de resultados

	CakePhp			Prado			Zend		
	Subcaracterista	Atributo	Caracteristica	Subcaracterista	Atributo	Caracteristica	Subcaracterista	Atributo	Caracteristica
1.Idoniedad	7.78			6.49			8.26		
Idoniedad Reportes		9.18			5.82			8.82	
Idoniedad Alertas		6.38			7.25			7.69	
2.Precisión	4.33			7.00			7.50		
Verificabiliad		4.33			7.00			7.50	
3.Interoperabilidad	4.25			8.50			8.50		
Interoperabilidad Directa		4.25			8.50			8.50	
4.Seguridad	8.71			8.29			9.57		
Aplicación de Seguridad		8.71			8.29			9.57	
5.Cumplimiento en el Funcionam	2.00			2.00			2.00		
Funcionalidad			5.42			6.45			7.17
1.Madurez	8			7			9		
Historicos del producto		8			7			9	
2.Tolerancia a fallos	10			8.50			8.50		
Nivel de tolerancia		10			8.50			8.50	
3.Capacidad de recuperacion	5.17			5.17			6.83		
Recuperabilidad del sistema		9			9.00			8	
Recuperabilidad de datos		1.33			1.33			5.67	
Confiabilidad			7.72			6.89			8.11
1.Capacidad para ser entendido	4.58			5.00			4.58		
Comprensibilidad de interfaz		6.67			10.00			6.67	
Estructura global		10			0			10.00	

2.Capacidad para ser aprendido	6			7.8			6.63		
Entrenamiento		6.5			8.50			7.00	
Documentación		5.5			7.25			6.25	
3.Capacidad para ser manipulado	4.52			5.33			8.36		
Sistema hecho a la medida		7.71			5.00			7.71	
Apariencia hecha a medida		1.33			5.67			9.00	
Usabilidad			5.04			6.07			6.52
1.Utilizacion de recursos	8.75			3			8.50		
Implementación		8.75			3			8.50	
Eficiencia			8.75			3			8.50
1.Analizabilidad	5.33			9.00			9.50		
Datos analizables		5			10.00			10	
Analisis de capacidades		5.67			8.00			9	
2.Variabilidad	5.92			6.58			6.58		
Entorno de desarrollo		7.17			7.50			7.33	
Desarrollo de documentación		4.67			5.67			5.83	
3.Estabilidad	5.75			5.75			7.25		
Estabilidad Operacional		5.75			5.75			7.25	
4.Capacidad de Prueba	3.33			3.00			2.33		
Mantenibilidad			5.08			6.08			6.42
1.Adaptabilidad	10			6.67			10		
2.Instalación	9.67			9.67			8.67		
Configuración Instalación		10			10			10	
Soportes de Intalación		9			9			6	
Compatibilidad con Sistemas Operativ		10			10			10	
3.Coexistencia	7.67			8.17			8.83		
Utiliza multiples base de datos		7.67			8.17			8.83	
Portabilidad			9.11			6.08			9.17
Resultado			6.85			5.76			7.65

9.8 Análisis de las ventajas y desventajas de cada herramienta

Al analizar los datos generados en el modelo ISO/IEC 9126 se obtuvieron resultados cuantitativos en función de los atributos propios de cada herramienta, estableciendo las ventajas y desventajas de su utilización.

9.8.1 Ventajas de CakePhp

1.-Dentro de la eficiencia en la sub-característica implementación se considera, el atributo cantidad de código sobre el cual, CakePhp ofrece amplias ventajas por varios factores entre ellos la implementación de Scaffolding, además la programación en el framework requiere menos código al concentrar su lógica en los controladores, facilitando la reutilización.
2.-Dentro de la funcionalidad en la sub-característica idoneidad podemos destacar el uso de Cakephp, puesto que su core, posee una distribución interna de sus reportes y

presentación; no hace falta implementar ningún componente o control de ayuda, solo invocar al tipo de presentación en el caso de ser en formato reporte se utiliza una llamada a view.
3.-Dentro de Confiabilidad en la sub-característica tolerancia a fallos, Cakephp ofrece una amplia ventaja puesto el framework despliega mensajes de presentación de errores confiable al indicar las clases que afectan dicho error y la línea exacta que lo desata, además de datos acerca del tiempo de respuesta del sistema ante el error producido.
4.-CakePhp resulta una herramienta poderosa dirigida a proyectos pequeños.
5.-Cakephp posee un desarrollo totalmente organizado, así también la cantidad de recursos requerida es mínima, dicha característica se encuentra dentro de eficiencia a nivel de utilidad de recursos.
6.-La generación de pruebas es otra de las ventajas generadas en CakePhp a nivel de mantenibilidad cuya sub-característica es la capacidad de pruebas.

9.8.2 Ventajas de Prado

1.-Dentro de la usabilidad en las sub-características capacidad para ser entendido, puesto que el captar el funcionamiento de su interfaz resulta sencillo al implementar extensiones que se enlazan al IDE de desarrollo, con todo sus API's propias, facilitando la programación en este aspecto.
2.- Dentro de la usabilidad en la sub-características capacidad para ser aprendido prado ofrece una amplia ventaja, debido a que su curva de aprendizaje es mejor en relación con los frameworks evaluados.
3.-Una de las ventajas de mayor peso en Prado se encuentran en el manejo de la interacción con el usuario al incorporar eventos en su desarrollo.
4.-A pesar de que la documentación no es completa, son de gran ayuda los demos y videos tutoriales.
5.-Dentro de la mantenibilidad sub-característica variabilidad; el entorno de desarrollo es considerado uno de los mejores puesto que la información de las bibliotecas API se actualiza constantemente con versiones estables.

9.8.3 Ventajas de Zend

1.-Dentro de la funcionalidad en la sub-característica idoneidad de alertas, zend presenta gran ventaja al permitir la configuración de sus alertas en los elementos propios del componente del formulario, a su vez facilitan la configuración total de sus alertas.
2.-Framework presenta mayor aceptación en el mercado por esta razón existe mayor información generada por los usuarios del framework, atributo clasificado dentro de confiabilidad en la sub-característica de madurez.
3.-A nivel de funcionalidad en la sub-característica de precisión zend ofrece ventajas al integrar mejoras gracias a la utilización de componentes.

4.-Dentro de la funcionalidad en la sub-característica de seguridad zend posee mayor seguridad al transferir datos, al manejar autentificación, cookies entre otros.
5.-La capacidad de recuperación a nivel de confiabilidad, zend posee mejor nivel en sus logs de error y produce menor tiempo en la recuperación en función del error.
6.-Otra de las características fundamentales de zend es la recuperación de datos, puesto que implementa componentes cuya función es generar backup's del sitio, este atributo está considerado en la confiabilidad sub-característica recuperabilidad de datos.
7. En la característica de usabilidad se considera la sub-característica capacidad de ser manipulado, donde zend es ajustable a cualquier requerimiento, dicha característica resulta fundamental para los desarrolladores.
8.-Asi pues un sitio debe poseer una interfaz amigable y ajustable sobre la cual se manejen estilos, plantillas y sobre todo su implementación sea sencilla, esta es otra de las ventajas ofrecidas por zend dicho factores está considerado en usabilidad apariencia hecha a la medida.
9.-La utilización de testing facilita la corrección y las mejoras de dicho factor dentro de la mantenibilidad sub-característica analizabilidad.
10.-El manejo de versiones estables y probadas reduce el tiempo en el desarrollo y evita errores factor implementado en estabilidad sub-característica estabilidad operacional.
11.-Zend no posee variedad de convenciones lo cual reduce el tiempo de desarrollo, dejando de lado características de bajo nivel, dicho factor se encuentra dentro de la característica portabilidad, sub-característica coexistencia.

9.8.4 Desventajas de CakePhp

1.-Presenta versiones de prueba (beta), la generación de información por parte de las comunidades es pobre, componentes inestables, factores evaluados en funcionalidad a nivel de precisión.
2.- No posee una integración API's desarrollo con IDE factor considerado dentro de funcionalidad a nivel de interoperabilidad.
3.-No presenta una estructura flexible a cambios, sino es el desarrollador quien debe adaptarse a su uso, factor evaluado en usabilidad sub-característica capacidad para ser manipulado.
4.-No presenta componentes estables sino que se lanzan versiones betas factor analizado dentro de mantenibilidad relacionado a variabilidad.
5.-La cantidad de convenciones es grande, lo cual retrasa al proyecto dicho atributo considerado al evaluar la portabilidad dentro de sub-característica de coexistencia.

9.8.5 Desventajas de Prado

1.-Presenta problemas al usar reportes, en especial en el control de paginación dificultando la idoneidad sub-característica de generación de reportes.

2.-Genera problemas en el uso del componente de autentificación al generarse solo sobre una tabla y su modificación requiere cambios en el componente dicho atributo esta evaluado dentro de seguridad en la característica de funcionalidad.
3.-No presenta mayor aceptación en el mercado lo cual genera una amplia desventaja al no poseer información.
4.-Su desarrollo no es organizado y el código es redundante puesto que su lógica es creada para cada vista dicho factor está considerado dentro de usabilidad sistema hecho a la medida.

9.8.6 Desventajas de Zend

1.-El tiempo empleado en la realización de correcciones es mucho mayor, a pesar de poseer componentes de testeo de datos, puesto que todo el desarrollo de la aplicación se mantiene distribuido en toda la estructura resulta complicado generar cambios este parámetro de evaluación, está considerado dentro de mantenibilidad en la sub-característica capacidad de pruebas.

2.-El tiempo para realizar mejoras es elevado en relación a los frameworks evaluados puesto que los componentes, plugins, controles no soportan amplios cambios, dicha característica se encuentra en mantenibilidad dentro de implementar correcciones.

3.-No posee demos y si bien es cierto la documentación es amplia esta se encuentra en inglés considerándose una desventajas para quienes no dominan el idioma, este factor se encuentra dentro de soporte instalación en portabilidad.

Conclusiones

Al finalizar este capítulo se establecieron comparaciones bajo métricas comunes, las cuales permiten generar un rango de valores cuantitativos comprendidos entre 1 y 10 donde a mayor valor, mejor calidad pose el factor, la evaluación considera todas la características que implican la utilización de recursos como son el uso de componentes,documentación,curva de aprendizaje, usabilidad, organización, cantidad de código, capacidad para reutilizar librerías,plugins,elementos,componentes etc; de esta manera al generar resultados en cada una de las características y dichos resultados se obtienen al aplicar un promedio de las sub-características y atributos analizados facilitando la comparación y el establecimiento de ventajas y desventajas de cada uno de los frameworks.

CONCLUSIONES

Al finalizar la monografía se establecieron las siguientes conclusiones:

-La especificación y la evaluación extensiva de la calidad del framework es un factor clave para asegurar una calidad adecuada.
-Al generar modelos de calidad los factores en función del framework evaluado poseen niveles de abstracción, puesto que al hablar de calidad una de las partes podría estar refiriéndose a ella en su sentido más amplio, mientras que otro podría estar refiriéndose a su significado específico, sin embargo la evaluación del framework de desarrollo genera reducción de recursos al realizarse el proyecto.
-Se puede considerar que la ventaja principal de un framework es la reducción del costo en el proceso de desarrollo de aplicaciones software para dominios específicos, y la mejora de la calidad del producto final, además de aprovechar al máximo la reutilización de componentes o códigos ya creados, de esta manera se pretende reducir el tiempo en requerimientos de bajo nivel.
-Dentro de los modelos o estándar de calidad, el estándar que se adapta a la evaluación del framework es el ISO IEC 9126 al ser considerado como modelo mixto, en el que su principal característica es definir factores de calidad más abstractos que sean reutilizados virtualmente en todos los dominios posibles, facilitando así la comparativa de los frameworks, además de no ser un modelo rígido, facilitando su personalización tanto como sea requerido dentro de un proyecto particular, por esta razón otra de las ventajas sobre los demás modelos analizados es el permitir el análisis de calidad de componentes OTS al permitir determinar factores no técnicos relativos al proveedor, y al costo.
-Al desarrollar una aplicación con cada una de las herramientas como lo son: CakePhp,Prado,Zend se analizó el funcionamiento de aspectos como características generales, estructura, convenciones o reglas tanto de la base de datos, modelos, controladores, vistas, parámetros de configuración del core, manejo de componentes, vistas, configuraciones del testing, elementos, controladores estableciendo los pasos para su creación y utilización se explicó en que consiste el scaffolding además se determinaron las tareas comunes requeridas por los desarrolladores mediante la explicación de las clases, de esta manera se obtuvo una visión completa del framework para su posterior evaluación al determinarse factores comunes.
-Según las comparaciones vertidas el framework con mejores prestaciones es zend framework y desde mi punto de vista este framework es el mejor al evaluar sus múltiples beneficios como flexibilidad, aceptación en el mercado,documentación,comunidades activas, capacidad de integrar componentes, reutilización de código existente,seguridad,implementa testing,backup de seguridad, ajustable a requerimientos, diseño de interfaz ajustable al programador y fácil de implementar, maneja versiones estables, reduce el desarrollo, no utilizaba variabilidad de convenciones, uso de librerías etc.

RECOMENDACIONES

-Para alcanzar las características de calidad apropiadas, se debe considerar el propósito y el uso del producto software.

- Es importante, que cada característica relevante de calidad del producto software se especifique y evalué; usando dentro de lo posible métricas cuantitativas que permitan establecer resultados medibles.

- Al finalizar esta monografía se recomienda el uso de frameworks, puesto que facilita el desarrollo de aplicaciones al cumplir los requisitos del cliente y generen mayores ganancias en menor tiempo, convirtiéndose en una ventaja competitiva en un mercado globalizado.

BIBLIOGRAFÍA

1. **Marín, Eyda.** (02/10/2008). Administración de Proyectos Informático. http://www.uhu.es/eyda.marin/apuntes/gesempre/Tema5_1IGE.pdf. Recuperado el 14 de enero de 2012.

2. **Ruiz, José**. (2009). ISO 9126 vs. SQuaRE. http://alarcos.inf-cr.uclm.es/doc/cmsi/trabajos/Joaquin%20Ruiz%20Expo.pdf.Recuperado el 16 de enero del 2012.

3. **Carvallo, Juan**. (s.f.). Evaluación de la calidad y selección de componentes de software. http://www.essi.upc.edu/~carvallo/Sesion1CS.pdf. Recuperado el 18 de enero del 2012.

4. **Verdoy, Alberto**. (2009, julio 15). Ventajas de usar frameworks para PHP. http://www.tucamon.es/contenido/ventajas-de-usar-frameworks-en-php. Recuperado el 18 de 13 de enero del 2012.

5. **García, Félix**. (2008). Proceso Software y Gestión del Conocimiento. http://alarcos.inf-cr.uclm.es/doc/psgc/doc/psgc-4a.pdf. Recuperado el 19 de enero del 2012.

6. **Ariza, Maribel & Molina, Juan**. (2004, Septiembre 30). Introducción y Principios Básicos del desarrollo de software basado en componentes. http://dis.um.es/~jsaez/dbc/curso1011/docs/art03_DSBC.pdf. Recuperado el 15 de enero del 2012.

7. **Fillottrani, Pablo R.** (2007). Calidad en el Desarrollo de Software. http://es.scribd.com/doc/75907374/clase6. Recuperado el 18 de enero del 2012.

8. **Carvallo, Juan.** (s.f.). Evaluación de la calidad y selección de componentes de software. http://www.essi.upc.edu/~carvallo/Sesion2CS.pdf. Recuperado el 17 de enero del 2012.

9. **Carvallo, Juan & Franch, Xavier**. (s.f.). Selección de Componentes; Off-the-shelf. http://www.essi.upc.edu/~franch/papers/libro-calidad-cap-17-jpc-xf-10-version-preliminar.pdf. Recuperado el 20 de enero del 2012.

10. **Carvallo, Juan & Franch, Xavier.** (s.f.). Análisis de Desajustes con Respecto a los Requisitos en la Selección de Componentes OTS.

http://www.essi.upc.edu/~franch/papers/ideas09-jpc-xf-camera-ready.pdf. Recuperado el 25 de enero del 2012.

11. **Bertoa, Manuel F & Troya, José M & Vallecillo, Antonio**. (s.f). Atributos de Calidad para Componentes COTS: Una valoración de la información ofrecida por los vendedores. http://www.lcc.uma.es/~av/Publicaciones/03/TICS03.pdf. Recuperado el 29 de enero del 2012.

12. **Carreón, María C.** (25/06/2008). Construcción de un catálogo de patrones de requisitos funcionales para ERP. http://upcommons.upc.edu/pfc/bitstream/2099.1/5452/1/TESINA_Carre%C3%B3n_Su%C3%A1rez%20del%20Real.pdf. Recuperado el 27 de enero del 2012.

13. ISO25000. [Online] 2012. http://iso25000.com/.

14. Prado. Component Framework for PHP 5. [Online] 2004, 2012. http://www.pradosoft.com/

15. Zend Framework. [Online] 2006 - 2012.http://framework.zend.com/.

16. **Ayala, Claudia & Franch, Xavier. (s.f).** Gestión Sistemática de la Calidad de la Información en los Procesos de Selección de Componentes de Software. http://www.lsi.upc.edu/~cayala/Papers/Ideas09-ca-xv-CameraReadyVersionFinal-Final.pdf. Recuperado el 10 de febrero.

17. Norma ISO/IEC 12007: International Organization for Standardization. [Online] s.f. http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_ics_browse.htm?ICS1=31.

18**. Mena, Gonzalo.** (2006). ISO 9126-3: Métricas Internas de la Calidad del Producto de Software.http://mena.com.mx/gonzalo/maestria/calidad/presenta/iso_9126-3/. Recuperado el 18 de Febrero del 2012.

19. **Kan,Stephen.** Metrics and Models in Software Quality Engineering. Addison Wesley. September 20, 2002

20. Cake Software Foundation Inc. [Online] 2005- 2012.http://www.cakephp.org/.

21. PHP Frameworks. [Online] 2011.http://www.phpframeworks.com/.

22. **Zend Technologies Inc.** (03/02/2012).Programmer's Reference Guide. http://packages.zendframework.com/docs/latest/manual/en/zend.db.html#zend.db.adapter

ANEXOS

1.-Casos de Uso