Universidad de las Américas

Facultad de Ingeniería y Ciencias Agropecuarias

# Framework generador de formularios HTML Web 2.0

Andrés Torres Russo.

2009

# Universidad de las Américas

Facultad de Ingeniería y Ciencias Agropecuarias

Ingeniería en Sistemas

**Framework generador de formularios HTML Web 2.0**

Trabajo de titulación presentado en conformidad a los requisitos para obtener el título de Ingeniero de Sistemas.

Profesor Guía: Ing. Juan José León

**Autor: Andrés Torres Russo**

**2009**

## Declaración Profesor  Guía

Declaro que le presente trabajo fue desarrollado completamente por el señor Andrés Torres Russo, bajo mi orientación y guía.

Ing. Juan José León
Profesor guía
C.C.: 170750676-0

## Declaración de Autoría del Estudiante

Declaro que este trabajo es original, de mi autoría, que se han citados las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes.

## Agradecimientos

Éste trabajo no hubiera sido posible sin la gran ayuda y apoyo de muchas personas, gracias a todos aquellos que han estado conmigo en la realización de esta documentación, en especial a Gustavo Baquero, mi gran amigo, Juan Rodríguez, mi mentor.

## Dedicatoria

Dedico el presente trabajo a mis padres, que han sido y son los impulsores de las mejores cosas de mi vida.

## Resumen

Uno de los trabajos que más tiempo requiere ser desarrollado al momento de administrar las tablas de la base de datos de un proyecto, cualquiera que este sea, es crear interfaces o pantallas para la gestión de administración de las mismas, generando de esta forma formularios *HTML* para la manipulación de la información.

Esto ha sido un gran problema para un desarrollador Web ya que se deben generar muchos componentes para que las acciones antes mencionadas ocurran; el tiempo involucrado en la creación de dichas soluciones es exponencial y suele ser una tarea muy laboriosa; además de que a medida que la tecnología avanza en la Web se requiere de más conocimientos al momento de implantar una solución de software para Internet. Se requiere de herramientas avanzadas que permitan optimizar los tiempos de desarrollo y que eviten al máximo el crear aplicaciones desde cero.

Es por eso el nacimiento de **DbForm**, que es un marco de trabajo orientado a objetos diseñado con el objetivo de facilitar al desarrollador Web, donde la creación de formularios *HTML* generados a partir de esquemas de bases de datos, que cumplen con estándares y las tecnologías propuestas por la *Web 2.0*, reduciendo ampliamente el tiempo que toma el desarrollo e implementación de una interfaz Web para administración de registros de una base de datos cualquier número de tablas; además se integra a una arquitectura Modelo-Vista-Controlador provista por el *Zend Framework*, marco de trabajo orientado a objetos desarrollado y mantenido por *Zend*.

**Tabla de Contenidos**

## Tabla de Figuras

## Tablas

# 1. Introducción

En el desarrollo de una aplicación Web, sea éste un portal o un sitio Corporativo siempre se debe considerar la implementación de una aplicación alimentadora de la base de datos que debe acompañar como parte de la estructura fundamental de esta aplicación.

Estas aplicaciones son formularios generados en *HTML*[1], generalmente bajo el mismo diseño gráfico de la aplicación o con un diseño personalizado y bajo un esquema de seguridad de control de acceso, ya que no cualquiera puede modificar los datos de una aplicación arbitrariamente; y se encargan de recoger de una manera amigable los datos de cada tabla de la base de datos y aplicar las acciones de: creación y modificación de información para cada uno de los registros de las tablas.

Esto quiere decir que se deben desarrollar las pantallas de administración, tanto para añadir un registro nuevo como para modificar al mismo, para cada una de las tablas que el proyecto pueda disponer; se debería adecuar el diseño gráfico propuesto para cada una de estas pantallas e integrar éste desarrollo bajo un marco de trabajo que permita su mantenibilidad y extensibilidad, haciendo que toda la plataforma esté en relación con la tecnología actual; hacer uso de funcionalidades extras como *AJAX*[2] para la gestión de datos haciendo más amigable la interacción de los formularios con los futuros usuarios o alimentadores del sistema.

Todos estos requerimientos deben establecerse bajo parámetros y estándares para su implementación haciendo que el desarrollo cada vez se vea más complicado y los tiempos de desarrollo sean cada vez más extensos, además se requiere que los desarrolladores de software conozcan de muchas más herramientas y tecnologías para la elaboración de aplicaciones Web modernas

---

1 HTML: hyper text markup language, metalenguaje para desarrollo de páginas Web.
2 AJAX: Asynchronous Javascript and XML, peticiones asincrónicas a un servidor Web

que cumplan con los lineamientos que propone la *Web 2.0*[3].

## *1.1 Formulación del problema a resolver*

Debido al problema planteado anteriormente, el presente trabajo de investigación pretende desarrollar un *framework* que permita a un desarrollador Web generar de manera óptima y rápida formularios *HTML* de administración de registros de una base de datos en plataforma *Web* y en un ambiente *Web 2.0*.

Reduciendo la carga de trabajo asignada a la ejecución de las tareas antes mencionadas, proveyendo al desarrollador de una herramienta útil de trabajo que agilitará el desarrollo de aplicaciones entregando resultados correctos de una manera dinámica y amigable.

Este proyecto y la investigación necesaria se apoyarán en el uso de tecnologías modernas de desarrollo de aplicaciones Web, estándares, uso de las mejores prácticas en el desarrollo de software y patrones de diseño. Se utilizarán herramientas *Open-Source* para su implementación; y finalmente se pondrá a disposición de la comunidad *Open-Source* el aplicativo desarrollado con el fin de generar una contribución a la misma y ganar reconocimientos por el aporte. Cabe mencionar que el estándar de programación será *PHPDoc*[4] para la codificación correcta de la aplicación, esto se explicará más adelante, sin embargo es válido mencionar que gracias a este estándar generar la documentación del *framework* será muy sencilla.

## *1.2 Objetivo general*

Implementar un sistema de software que se convertirá en un *framework* que permita generar formularios *HTML* en un ambiente *Web 2.0*, a partir de

---

3 Web2.0: Uso de un grupo de tecnologías aplicadas a la Web, discutido por O'Reilly Media, Inc. y MediaLive International

4 PHPDoc: Estándar de documentación de código fuente PHP

esquemas de bases de datos *MySQL 5.0*.

## *1.3 Objetivos específicos*

Los objetivos específicos a alcanzar con este desarrollo son los siguientes:

- Investigar y definir el concepto de *Web 2.0*, con el fin de comprender a cabalidad sus características y estándares.
- Diseñar un *framework* en un ambiente *Web 2.0*, para la generación de los formularios *HTML* en base a una arquitectura de desarrollo bajo el patrón de diseño *MVC* (Modelo – Vista – Controlador)[5], arquitectura implementada por *Zend Framework*[6], marco de trabajo *MVC* para *PHP 5*.
- Implementar el *Framework* utilizando un modelo orientado a objetos bajo el lenguaje de programación *PHP* versión 5.
- Generar aplicaciones Web en base al *framework* propuesto y demostrar la fácil administración, mantenimiento y escalabilidad en el desarrollo de estas aplicaciones.
- Aportar este Framework y su documentación a la comunidad *Open-Source*, principalmente la comunidad de *Zend*; para así contribuir con una solución que agilite considerablemente la generación de formularios partiendo de una base de datos *MySQL*.

Es decir que el propósito de esta investigación es generar la primera versión de un *framework* que se integre con el *Zend Framework*, del cual se hablará ,más adelante, con la finalidad de aprovechar la arquitectura *MVC* de éste como también ciertas de sus librerías para la generación de plantillas *HTML*

5 MVC: Modelo Vista Controlador, patrón de diseño de software
6 Zend Framework: Marco de Trabajo para desarrollo de aplicaciones con PHP 5 orientado a objetos

reusables y optimizar el rendimiento de la aplicación y su escalabilidad; que permitirá al desarrollador de software generar rápidamente formularios *HTML* de administración de los registros de las bases de datos, integrándose así con todo el resto del ambiente de la solución general.

Esto permitirá una reducción considerable de tiempos en la creación de pantallas con los formatos y la programación de la lógica de negocio para los registros de las tablas de la base de datos de diferentes tipos de proyectos.
Hay que dejar claro que el sistema a desarrollarse es un conjunto de funciones de apoyo para construir partes de una solución de software; definiéndolo así como un *framework*. Todos estos temas se tratarán más a fondo durante el desarrollo de éste proyecto.

## *1.4 Definición de Framework*

Un *framework*, o Marco de Trabajo, es una solución de software que brinda ciertas facilidades para que otra solución de software sea creada a partir de ella, facilitando la implementación de funcionalidad y reduciendo notablemente los tiempos de desarrollo ya que el desarrollador se puede enfocar de mejor manera en los requerimientos específicos del negocio mismo del sistema y disponer de mayor tiempo para realizarlos.

## *1.5 A quien va dirigido éste desarrollo*

El siguiente proyecto está dirigido a profesionales que tengan conocimientos en la generación de soluciones Web, experiencia en programación orientada a objetos con *PHP* y que manejen el *Zend Framework* para el desarrollo de sus aplicaciones Web, que conozcan de implementación de soluciones de software en arquitectura *MVC*, conocimientos en manejo de motores de base de datos y estándares *SQL* y adicionalmente debe estar bien relacionado con las tecnologías propuestas por la *Web 2.0* y sus estándares, para más referencia en la terminología planteada en este documento referirse al Anexo I (Glosario

del Negocio).

## *1.6 ¿Por qué utilizar Zend Framework?*

Siendo el *Zend Framework* un conjunto de librerías orientadas a objetos construido en *PHP* 5, para el desarrollo de aplicaciones de gran y pequeña escala en el mismo lenguaje de programación; una de sus características principales es que permite armar un ambiente *MVC*, patrón de diseño para crear soluciones de software, que facilita la extensibilidad, mantenibilidad, transportabilidad y reusabilidad de una solución de software.

Adicionalmente, el *Zend Framework* brinda la facilidad al desarrollador para construir ágilmente servicios Web con un extenso conjunto de librerías propias y aportadas por la comunidad de desarrolladores *PHP* así como funcionalidades para la conectividad a diferentes *RDBMS*[7].

7 RDBMS: Relational Data Base Management System, sistema de manejo de base de datos relacionales.

# 2 Metodología para el desarrollo del *framework*

Una metodología de software significa realizar el desarrollo de un sistema siguiendo una serie de pasos, generando documentación y cumpliendo con una serie de condiciones o metas. Esto se realiza con el fin de garantizar la correcta construcción de los aplicativos y que a su vez se disponga con documentación necesaria para que se pueda comprender lo realizado o para entender como generar nuevas funcionalidades a la herramienta.

En la actualidad se pueden encontrar un sin número de metodologías, y aunque cada una de ellas resuelve al desarrollo de un sistema de distinta manera, todas comparten el propósito de elaborar de manera correcta y estandarizada los proyectos de software.

Siendo el propósito de este proyecto de tesis crear una primera versión de un *framework* para la generación de formularios Web, se ha decidido implementar la metodología ágil llamada *Programación Extrema*, más conocida como *XP*; para desarrollar los métodos encargados de recibir parámetros específicos y que devuelven respuestas de la misma manera que formarán parte del *framework*.

Se determino la utilización de *XP* puesto que el producto final de éste proyecto de titulación es un grupo de funcionalidades codificadas en *PHP 5* que serán el primer prototipo estable de una solución que evolucionará a nuevas versiones, característica fundamental de *XP.* El proyecto requiere ser codificado en el menor tiempo posible, deberá ser susceptible a recurrentes cambios y debe permitir una integración cada vez más profunda con el *Zend Framework*. La documentación generada con *XP* es puntual y concisa, gracias a que propone continuidad en el acercamiento con el usuario final, por ende ayuda a que los prototipos sean aprobados en un menor tiempo posible y asegura la estandarización y el rendimiento de la solución.

## *2.1 Programación extrema*

Programación Extrema o en inglés *eXtreme Programming (XP)*, es una

metodología de desarrollo de software que forma parte de las metodologías ágiles y rápidas para el desarrollo de aplicaciones de software.

Estas metodologías se caracterizan por proporcionar suficiente pero limitada documentación, liberando de esta manera la carga de trabajo a los participantes del desarrollo proveyéndoles de mayor tiempo para que puedan generar las soluciones requeridas. También se caracterizan por gestionar un gran número de reuniones con los clientes, que permiten comprobar periódicamente los avances de la aplicación. Otra característica importante es que el tiempo asignado suele ser muy reducido y poco flexible.

*Kent Beck* en el año 1999 en su libro "*Extreme Programming Explained: Embrace Change"* propone una metodología que está basada en una serie de prácticas y valores de simplicidad, comunicación, retro alimentación y valentía, persiguiendo el objetivo de mejorar la productividad al momento de desarrollar soluciones de software.

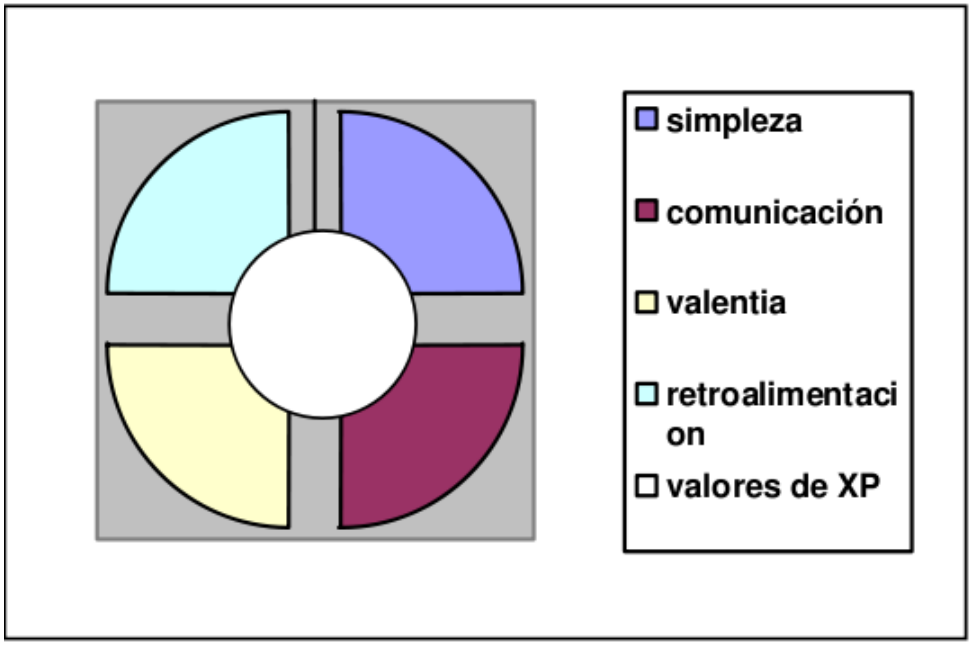


**Figura 1 Valores XP**

### 2.1.1 Características

Las principales características de esta metodología se organizan en 12 principios básicos explicados en 4 categorías:

- **Retroalimentación a escala fina**
  1. **El principio de pruebas**: pretende establecer un periodo de pruebas de la aplicación para definir las entradas al sistema y las salidas del mismo.
  2. **Proceso de planificación**: se requerirá que el usuario aporte con el llenado de documentos llamado *Historias de Usuario* o *User Stories* en inglés, en donde se recogerán una lista detallada de los requerimientos específicos del sistema; son pequeños documentos de un máximo de tres líneas en donde se describe que debe hacer el sistema, para mayor referencia dirigirse al Anexo A (Historia de Usuario). La primera versión del *framework* utilizará la métrica de Puntos de Función de Mark II, que además proveerá con la información de los costos del desarrollo de la iteración I, para mayor información referirse al Anexo B (Estimación de Costos y Recursos)

     En conjunto, las historias de usuarios y los costos, son la información necesaria para elaborar el *Plan de Entregas*, que define de manera clara y especifica los tiempos de entrega de la aplicación para posteriormente recibir retroalimentación por parte del usuario, mayor información referirse al Anexo C (Plan de Entregas).

     Dichas reuniones son muy importantes y deben ser realizadas periódicamente involucrando a todos los miembros del equipo.
  3. **El cliente en el sitio**: que se refiere a que el usuario final sea partícipe en el proceso de desarrollo de la aplicación, ayudando con aclaraciones en los requerimientos, estableciendo prioridades y tiempos de elaboración, ésta práctica disminuye el proceso largo de comunicación cliente-desarrollador y la cantidad de documentación a generarse más sus costos, idealmente el cliente estará junto al equipo de desarrollo durante todo el proceso de elaboración del proyecto.
  4. **Programación en parejas**: si bien no es un requisito se recomienda que la programación de la aplicación sea en parejas, puesto que con

ésta práctica los programadores escriben mejor código, generando aplicaciones óptimas. En el caso de ser una sola persona las reglas de *XP* no cambian, sin embargo hay que tomar en cuenta que con diferentes roles se puede distribuir tareas de mejor manera.

- **Proceso continuo en vez por lotes**
  1. **Integración continua**: se refiere a cada modificación que se vaya realizando en el sistema es preferible incluirla a menudo en la aplicación general y no esperar a incluir cúmulos de modificaciones que podrían derivar en conflictos.
  2. **Refactorización**: se refiere a que a medida que el desarrollo se va dando, los programadores pueden re codificar partes de la aplicación ya implementadas, mejorando continuamente el código de la misma.
  3. **Entregas pequeñas**: realizar funcionalidades o prototipos que ya sean visibles para el usuario y que se vayan modificando con el proceso de desarrollo. Ésta será la primera versión del *framework* que pasará a pruebas.

- **Entendimiento compartido**
  1. **Diseño simple**: hacer lo minino necesario en el menor tiempo posible, es decir mantener un código sencillo.
  2. **Metáfora**: se busca catalogar a las acciones del sistema con frases descriptivas en lugar de por ejemplo: los tradicionales diagramas descritos por *UML*, o los distintos artefactos de *RUP*: que son metodologías características por generar una extensa documentación.

     *"La metáfora expresa la visión evolutiva del proyecto que define el alcance y propósito del sistema."*[8]

     Las metáforas generalmente se documentan utilizando *Tarjetas CRC*

8 http://eisc.univalle.edu.co/materias/WWW/material/lecturas/xp.pdf

*(Clase, Responsabilidad y Colaboración)*[9] que ayudan a los programadores a definir las actividades a realizarse. Cada tarjeta significa una clase en programación orientada a objetos. Para la primera versión del *framework* se propone la utilización de *PHPDoc*, herramienta que permite la generación de la documentación de los métodos y las propiedades del código de las clases de un sistema, estas son ayudas para el programador, en el Anexo F (Documentación del *Framework*) se puede observar claramente el resultado generado.

3. **Propiedad colectiva del código**: cualquier miembro del equipo debe conocer y debe tener habilidad de modificar parte o todo el código en cualquier momento en el proceso de desarrollo del sistema.
4. **Estándar de codificación**: antes de empezar con la codificación, se debe establecer un estándar para la codificación y escritura del código para el sistema, éste deberá ser conocido por todo el equipo de desarrolladores, *"de forma que parezca que ha sido realizado por una única persona"*[10].

   Para la primera versión del *framework* y sus futuras versiones se deberá utilizar el estándar establecido por el *Zend Framework,* para mayor información referirse a la bibliografía del presente proyecto (sección final del documento de Trabajo de Titulación).

- **Bienestar del programador**
  1. **La semana de 40 horas**: optimización del tiempo, se refiere a que se debe trabajar duro en cumplir los objetivos en plazos cortos de tiempo, así que no se debe dar lugar a tiempos muertos causando que se generen cargas de trabajo incrementando las horas de trabajo en otros días. Se debe mantener fresca la mente del programador.

9 CRC: Clase responsabilidad y colaboración, documentador de tareas para desarrolladores.
10 http://www.chuidiang.com/ood/metodologia/extrema.php

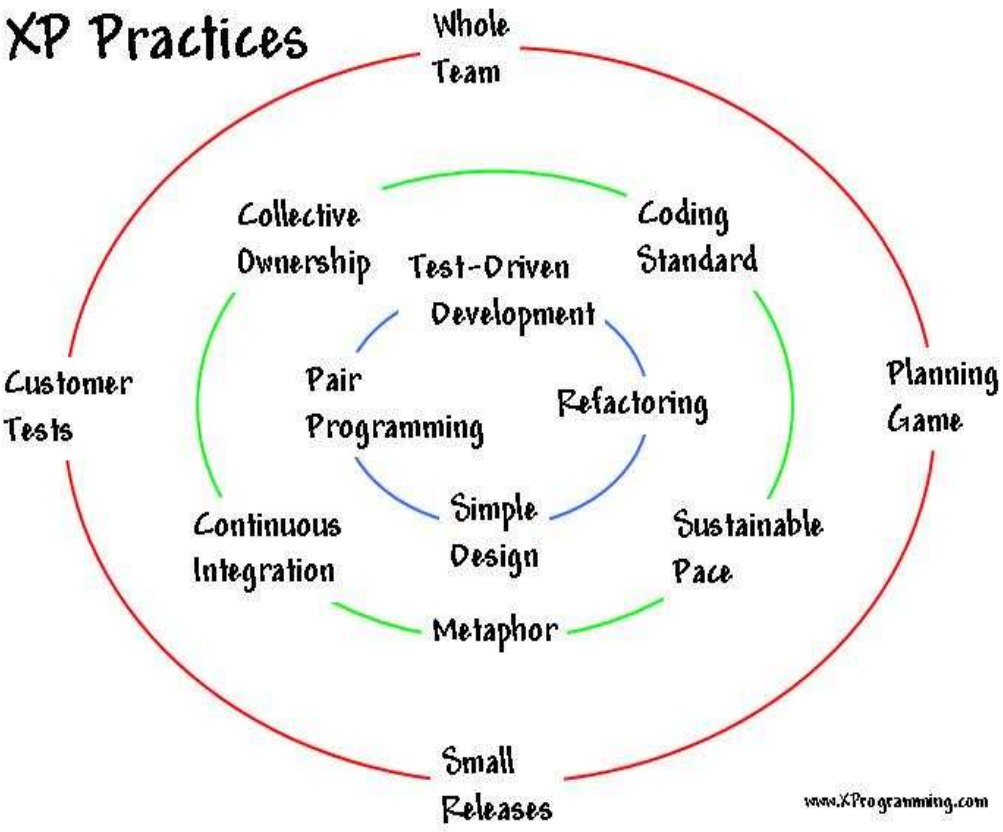


**Figura 2 Prácticas XP**

### 2.1.2 Ciclo de vida *eXtreme Programming*

*Extreme Programming* se resume a seis fases de desarrollo y se concentra en una serie de iteraciones en donde se incrementarán las funcionalidades y el tamaño del sistema. El proyecto planteado implicará la realización de la primera iteración del *framework*; esta es la razón por la cual las fases de Mantenimiento y Muerte del Proyecto, mencionadas más adelante, no aplican para el mismo, sin embargo desde la segunda iteración, ya serán aplicables.

I. **Exploración**: En esta fase se inicia el sistema debido a la necesidad presentada por el cliente. Se realizan una serie de reuniones con él para poder elaborar el documento de Historia de Usuario; este archivo contiene

los requerimientos del usuario, representado en cuadros simples donde se colocará la información puntual de la manera que el sistema debe funcionar. Para mayor información referirse al Anexo A (Historia de Usuario).

| HISTORIA DE USUARIO | | | | |
|---|---|---|---|---|
| Versión: | **1** | Nombre: | **Conectar con Base de Datos** | |
| Usuario: | **Andrés Torres, Gustavo Baquero** | | | |
| Modificación de Historia Número: | | **1** | Iteración Asignada: | **1** |
| Prioridad en Negocio: | | **1** | Tiempo Estimado: | **2 (días)** |
| Descripción:<br>- **Crear un API que permita una conexión a una base de datos MySql, que provea de funciones básicas para ejecutar consultas SQL; devolver resultados correctos de las consultas ejecutadas; incluir seguridades para garantizar óptimos resultados.** | | | | |
| Observaciones:<br>- **Debe implementar un patrón de diseño de software.**<br>**- Utilización de metodología de desarrollo.** | | | | |

**Tabla 1: Historia de Usuario - Conectar con base de datos**

En el proyecto presentado se realizará el documento de Estimación de Costos y Recursos, utilizando la métrica Mark II. En el mismo se exponen las fases del sistema y el recurso humano utilizado en cada una de ellas, esto más otros factores proporcionará el costo total del desarrollo del *framework*. Para mayor información referirse al Anexo B (Estimación de Costos y Recursos).

II. **Planificación de la Entrega (*Release Planning*)**: En esta fase se plantean una serie de elementos que apoyarán al cálculo correcto de tiempos de entrega, que son realizados en la fase anterior. El documento de Plan de Entregas es un cronograma de actividades del desarrollo de la aplicación. Hay que estipular las fases del ciclo de vida de *XP* y se incluirán los historiales de usuario como sub fases de la fase de Iteración. De esta manera se tendrá un tiempo de todos los elementos del proyecto. Para mayor información referirse al Anexo C (Plan de Entregas).

Posterior a esto se realizará el Plan de Iteraciones del sistema, donde se pretende presentar al cliente un listado de los puntos a cumplir en las diferentes iteraciones. Para el proyecto presentado se estipula una sola iteración debido a que se construirá un *framework*; una vez terminado el desarrollo y se realizado un análisis de funcionalidades se podrá empezar con las siguientes iteraciones, mismas que contemplarán todas las fases del ciclo de vida de *XP*. Para mayor información referirse al Anexo D (Plan de Iteraciones).

Adicionalmente cabe recalcar que se tendrán varias reuniones con el cliente para aprobar los documentos y para aclarar cualquier duda al equipo de desarrollo.

III. **Iteraciones**: En esta fase se realizan las iteraciones que el sistema contemple. Comúnmente involucran al cliente en la terminación de cada funcionalidad y hay una continua retroalimentación de información por la manera que se la lleva acabo, En cada iteración se realizan los puntos planteados en el Anexo D (Plan de Iteraciones) y se debe ajustar con los tiempos expuestos en el Anexo C (Plan de Entregas).

Las tareas involucradas son principalmente la codificación de las historias de usuarios, por lo que en el presente proyecto se realizarán las siguientes tareas dentro de esta fase:

a. Conectar con Base de Datos
b. Cargar el Menú de Administración.
c. Crear Formulario.
d. Cargar menú de Despliegue de Datos.
e. Editar un Formulario.
f. Integrar con *Zend Framework*.
g. Desarrollo de los Casos de Pruebas del sistema (documento para comprobar el funcionamiento de la aplicación).

Con el finalizar de cada iteración se tendrá el sistema listo para producción.

IV. **Producción**: Esta fase consiste en comprobar el correcto funcionamiento del sistema y de sus diferentes partes con el cliente. El documento de Casos de Pruebas es la guía a seguir para poder realizar esta sección. Para mayor información referirse al Anexo E (Casos de Pruebas).

    De igual manera se generará la documentación del *framework* que contendrá una explicación de cómo implementar el sistema y como funciona la aplicación en sus diferentes partes; en el presente documento se estipula la utilización del estándar *PHPDoc* que facilitará la generación de la documentación. Para mayor información respecto a la documentación del *framework* referirse al Anexo F (Documentación del Framework).

    Esta fase culmina con el sistema presentado en este proyecto.

V. **Mantenimiento**: En esta fase se estipula el soporte al cliente y el mantenimiento progresivo para el sistema. Comúnmente requiere de recursos adicionales para poder cumplir con todas las tareas planteadas. Se recalca que esta fase no aplica para el presente proyecto.

VI. **Muerte del Proyecto**: Es cuando el cliente deja de proveer más requerimientos y se puede dar por concluido la documentación y el trabajo sobre el proyecto en curso. De igual manera se puede presentar esto debido a la insatisfacción del cliente por los resultados expuestos en las fases anteriores. Cabe recalcar que esta fase no aplica para el presente proyecto.

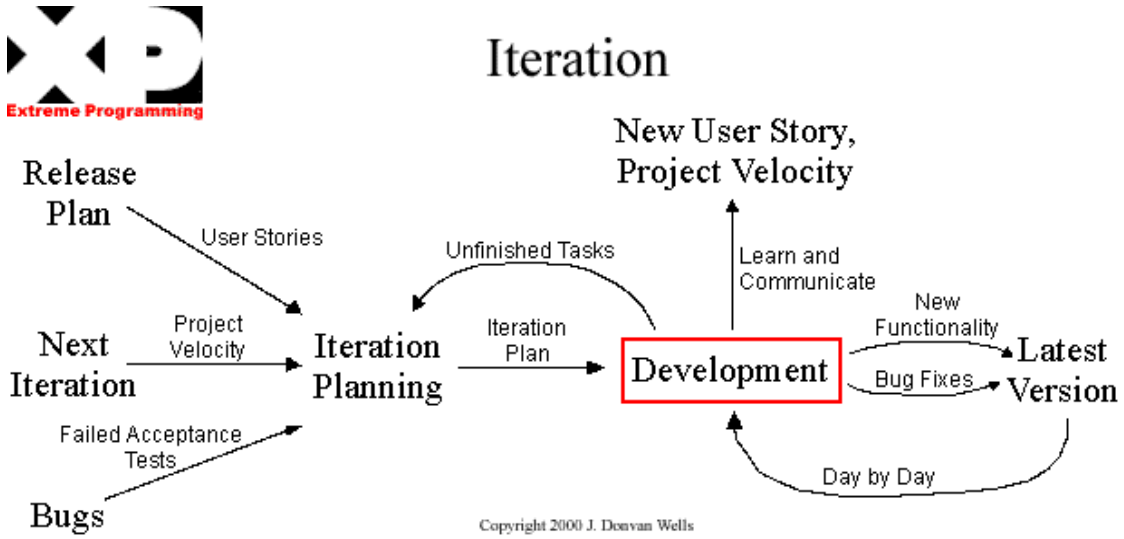


**Figura 3 Iteraciones XP**

### 2.1.3 Ventajas y desventajas de *XP*

*XP* es una metodología que ha generado muchas opiniones dentro del área de desarrollo de software a pesar de tener pocos años en el mercado, las opiniones positivas que *XP* tiene son:

- Los programadores novatos aprenden de los programadores más experimentados.
- El estilo de la programación se vuelve único.
- El código es conocido por todos, la propagación del código de las funcionalidades hacia todos los miembros del equipo ayuda a solucionar problemas más rápido.
- El cliente tiene el control sobre las prioridades y tiempos de los requerimientos.
- Las pruebas en las funcionalidades son continuamente realizadas durante todo el proyecto.
- Es una metodología que se adapta a cambios repentinos y rápidos en la aplicación.
- *"La calidad de los sistemas basados en XP tiende a ser un poco mejor,*

*en particular si se utilizan patrones de diseño."*[11]

- Es por medio de estos puntos que la programación en parejas propuesta por *XP* permite que el código fuente generado sea de mejor calidad, 4 ojos ven más que dos.

Las negativas:

- *XP* no garantiza el trabajo de 40 horas a la semana, puede ser que el tiempo que una funcionalidad se alargue y vaya fuera del estimado que fue propuesto.
- Debido a la constante refactorización, la documentación suele quedar desatendida.
- *XP* necesita de programadores muy buenos, que estén en la capacidad de hacer un buen diseño de software, rápido y sencillo.

Estos puntos exponen la eficacia de *XP* a pesar de sus críticas. Una pregunta común es ¿qué tan grande puede llegar a ser un proyecto utilizando con *XP*? Fácilmente se puede llegar a tener un equipo de 12 personas; sin embargo un desarrollo que requiera el doble de personal podría generar problemas de comunicación entre los desarrolladores, si esto llega pasar es recomendado optar por metodologías que apliquen, o crear varios grupos de desarrollo con *XP*, asignando tareas específicas a cada uno y así poder escalar considerablemente el número de participantes.

11 http://www.hackerdude.com/2002/10/18/programacion-extrema-mini-introduccion/

# 3 Web 2.0

En la actualidad es muy común en el ambiente de desarrollo Web encontrarse con el concepto de *Web 2.0*.

Inicialmente muchos desarrolladores creyeron que la *Web 2.0* era una tecnología nueva de desarrollo de aplicaciones en plataforma Web, otros creían que la *Web 2.0* era una metodología de desarrollo de aplicaciones para la Web. En ambos casos estaban equivocados.

*La Web 2.0* va más allá de ser una tecnología o una metodología para la creación de aplicaciones en plataforma Web, es una filosofía nueva para impulsar el uso de muchas tecnologías y metodologías ya existentes para construir diferentes aplicaciones en la Web que sean independientes de plataforma en la que son desarrolladas para ser ejecutadas y que permitan la interacción del usuario final con el contenido del sitio Web entre otras cosas.

## *3.1 Nacimiento de la "filosofía" Web 2.0*

El origen de *Web 2.0* nace en de una lluvia de ideas en una reunión que tuvieron dos grandes compañías de la Web, *O'Reilly Media, Inc.* y *MediaLive International*., el objetivo de esta reunión fue desarrollar ideas para una conferencia.

Una vez organizadas las ideas lanzadas en esta reunión se concluyó que la Web estaba en una revolución, con reglas que cambian y modelos de negocio que evolucionan rápidamente; comparando aplicaciones como *DoubleClick* (equivalente a la *Web 1.0)* y *Google AdSense* (equivalente a *Web 2.0*). Se lanzó la primera conferencia sobre la *Web 2.0* en Octubre del 2004. La segunda conferencia tuvo lugar en octubre de 2005.


**Figura 4 Mapa visual Web 2.0**

En esta sesión se determinaron los varios aspectos que las aplicaciones Web que se están generando en los últimos años aportando con sus contenidos y funcionalidades diferente a los sitios Web construidos bajo el concepto de la *Web 1.0* en donde los contenidos eran totalmente estáticos y la interacción con el usuario era escasa.

Ejemplos:

| **Web 1.0** | **Web 2.0** |
|---|---|
| Double Click | Google AdSense |
| Britannica Online | Wikipedia |
| Personal Web Sites | Blogging |
| Publishing | Participation |
| Ofoto | Flickr |

**Tabla 2: Ejemplos Web 2.0**

Uno de los ejemplos más representativos en la actualidad de la *Web 2.0* es ***facebook.com***, que es una comunidad virtual con más de 200 millones de usuarios registrados hasta el momento a nivel mundial.

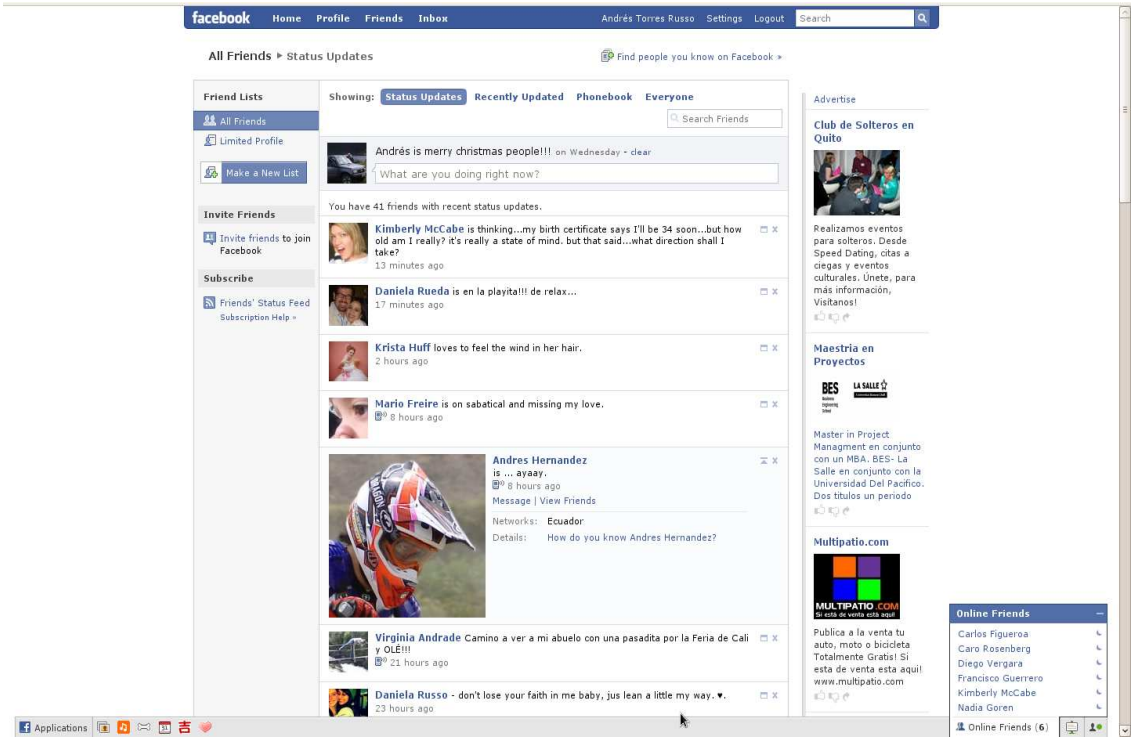


**Figura 5 Página de amigos de facebook.com**

En esta página se muestran principales elementos de una comunidad virtual, las relaciones existentes entre usuarios o "amigos", sus fotos, comentarios, blogs, acciones, entre otras cosas. Adicionalmente permite comunicación instantánea mediante el uso de un chat y otras opciones bastante amigables al usuario, que antes solo estaban disponibles en aplicaciones de escritorio.

## *3.2 Concepto de Web 2.0*

Engloba el uso de tecnologías de desarrollo de aplicaciones Web y metodologías de desarrollo de software, en donde el objetivo es generar aplicaciones que:

- Remplacen a las aplicaciones de escritorio, es decir que puedan ser capaces de sustituir las comunes aplicaciones como el chat, Word,

Excel, etc. y se centralicen independientes de un sistema operativo.

- La plataforma es la Web, es decir que las aplicaciones deberán ejecutarse vía un explorador Web o aplicativos que permitan una conexión con el Internet.
- El usuario es creador y consumidor de los contenidos del portal Web, es decir el usuario es capaz de generar contenido en un sitio Web, como por ejemplo hacer comentarios en noticias publicadas.
- Los contenidos se manejan con una herramienta generadora de contenidos o CMS (*Content Management System*), herramientas que ayudan al manejo ordenado de los contenidos de un sitio Web.

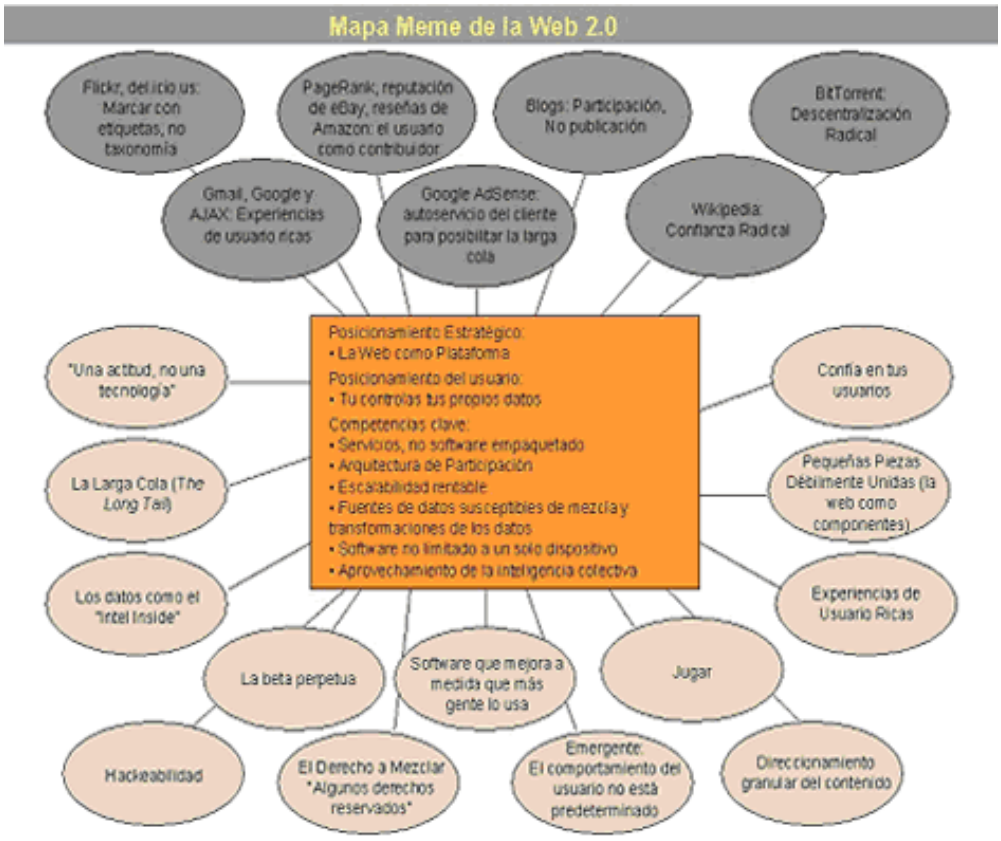


**Figura 6 Web 2.0 como plataforma**

## *3.3 Tecnologías*

La siguiente lista muestra las principales tecnologías y estándares utilizados

para el desarrollo de aplicaciones *Web 2.0*:

- Respetar los estándares establecidos por la *W3C*[12], como el *XHTML*[13].
- Separación de contenido del diseño con uso de hojas de estilo *CSS*[14] (*Cascade Style Sheets*).
- *Sindicación*[15] de contenidos.
- *AJAX* (Javascript asincrónico y XML).
- Implementación de herramientas de animación o multimedia como *Flash*, o *Flex.*
- Uso de lenguajes de programación como *Ruby on Rails*, *PHP*, entre otros.
- Utilización de comunidades virtuales.
- Dar control total a los usuarios en el manejo de su información.
- Proveer o consumir Servicios Web (*XML*).
- Facilitar el posicionamiento en buscadores de Internet (*google*, *Yahoo!*, entre otros), con *URL*[16] amigables o *friendly url's* en inglés.

Éstas tecnologías han existido durante mucho tiempo, es por eso que la *Web 2.0* no es una tecnología, sino la combinación de todas las antes mencionadas más las implementaciones de estándares y patrones para la construcción de software de calidad.

12 W3C: World Wide Web Consortium
13 XHTML: eXtensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto.
14 CSS: Cascade Style Sheets; hojas de estilo en cascada.
15 Sindicación: proceso de obtención de datos de otros servidores.
16 URL: Uniform Resource Locator, localizador uniforme de recursos; es una dirección que permite acceder a un archivo o recurso, tal como páginas html, php,asp, entre otros.

# 4 PHP

*PHP*, cuyo acrónimo significa *Personal Home Pages* o *Hypertex Pre-processor*, es un lenguaje de programación creado por *Rasmus Lerdof* en el año 1994.

Originalmente *PHP* fue diseñado para la creación de páginas Web dinámicas y a diferencia de muchos otros lenguajes de programación *PHP* es un lenguaje interpretado, es decir, cuando un cliente hace una petición a un servidor Web, éste toma como entrada código *PHP*, ejecuta el interprete *PHP* y como respuesta genera páginas dinámicas en formato *HTML* (por ejemplo obteniendo información de una base de datos y generando formularios). Puede ser instalado en los más populares servidores Web actuales, tales como *Apache*[17] (*Open-Source*) e *Internet Information Services*[18] (Software Licenciado), etc. y también tiene la capacidad de ejecutarse en la mayoría de sistemas operativos tales como *Linux, Windows, Unix* etc.

PHP, actualmente es mantenido por la compañía *Zend* [19] y es distribuido bajo la licencia *PHP License*[20] la cual especifica a *PHP* como software de libre distribución, utilización y producción.

17 Apache: Servidor Open Source de aplicativos Web.
18 IIS: Internet Information Services, servidor de aplicativos Web de Microsoft.
19 Zend: Organización que mantiene el lenguaje PHP.
20 PHP License: Licenciamiento de PHP.

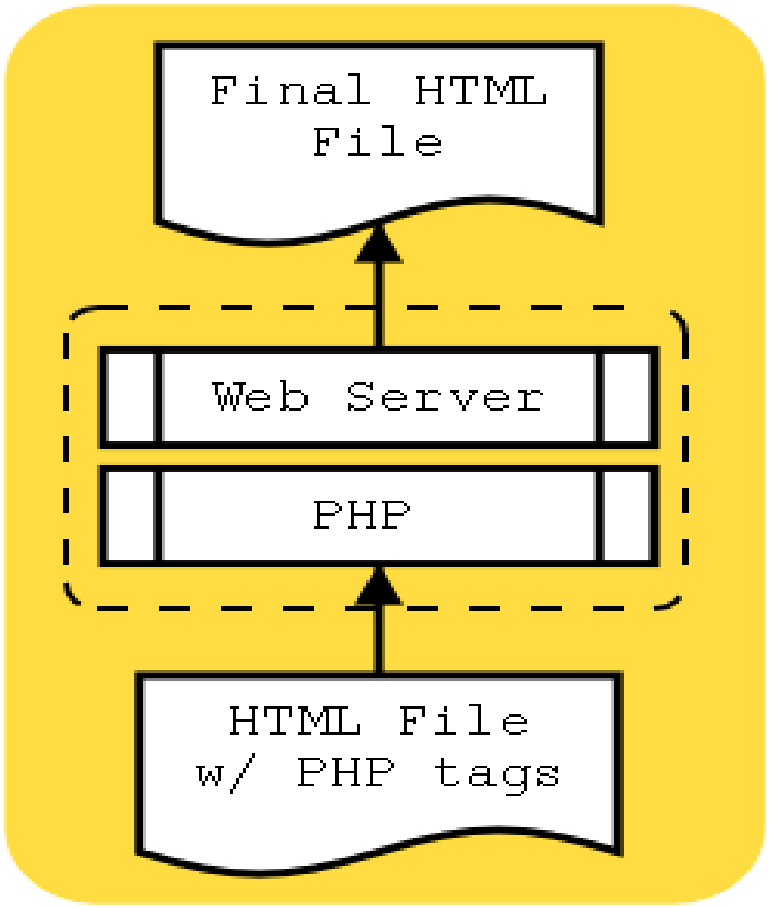


**Figura 7 PHP interpretado**

## *4.1 PHP 5*

La versión 5 de *PHP* fue lanzada en el año 2004 con un gran cúmulo de mejoras y nuevo desarrollos, mejor soporte para *XML*[21] y *Web Services*[22] pero sobre todo está el soporte nativo de un modelo orientado a objetos ya que *PHP* en sus orígenes no disponía de un modelo de objetos real. Ésta increíble mejora posiciona al lenguaje ya no solo como un lenguaje de programación de páginas dinámicas sino en un potente lenguaje de programación capaz de crear aplicaciones empresariales potentes y robustas tal como *Java* de *Sun*

21 XML: extensible markup language
22 Web Services:servicios Web en castellano, publicación de información de un sitio Web mediante XML

*Microsystems* o *C#* de *Microsoft.* Un ejemplo claro de esto es que grandes empresa como *Google*, *Yahoo!* o *Drupal*, utilizan *PHP* para sus sitios Web; muchos procesos recursivos son realizados por *PHP* y la lógica es manejada por el mismo lenguaje.

## *4.2 PHP y la Web 2.0*

En la era del la *Web 2.0*, el proceso para desarrollar software cambia para el desarrollador, puesto que en ésta, las aplicaciones son codificadas independientes de plataforma en la que residen, es decir que son codificadas para funcionar en cualquier sistema operativo, que no sea necesaria de instalar, simplemente que el usuario pueda acceder de la forma más simple y sencilla a su aplicación, desde cualquier computador o dispositivo que posea capacidades de conexión al Internet.

Existen más de 20 millones de sitios Web desarrollados en *PHP* en la actualidad, entre estos se destacan: *Yahoo!*, *Flickr.com*, *Facebook*, *Tagged.com*, que son sitos completamente desarrollados en *PHP* y los más nombrados como aplicaciones líderes de la *Web 2.0*, demostrando así que *PHP* es el lenguaje preferido para desarrollar aplicaciones *Web 2.0* gracias también a que es fácil de usar, fácil de aprender, es liviano y extendible, además cuenta con innumerables *API's*[23] y librerías que le permiten conectarse con tecnologías de la *Web 2.0* como librerías *AJAX* para el desarrollo de aplicaciones basadas en esta tecnología.

## *4.3 PHP como herramienta de desarrollo del framework*

Basado en el punto anterior, que expone a *PHP* como un potente candidato como lenguaje de programación para aplicaciones *Web 2.0* y tomando en

23 API: Access provider interface, interfaz que provee accesos remotos para consumir datos de una aplicación.

cuenta que el *Zend Framework* ofrece todas las capacidades para crear aplicaciones *Web 2.0* y está escrito en éste lenguaje, se escoge a *PHP* como lenguaje de desarrollo de la primera versión del *framework*.

# 5 Patrones de diseño de software

Los patrones de diseño de software son soluciones probadas a problemas recurrentes (patrones) que ocurren el momento de desarrollar sistemas, tratando de simplificar y mejorar continuamente el proceso de elaboración de aplicaciones o funcionalidades.

Éstas son soluciones que ya han sido probadas y que resuelven problemas comunes en el diseño orientado a objetos de cualquier aplicación.

El concepto de patrones de diseño viene de un trabajo realizado por 4 personas (*Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides*) que se publicó en el año 1995 en un libro con el título *"Patrones de diseño: Elementos de software orientado a objetos reutilizables"*[24].

Los patrones de diseño se clasifican en función del propósito para el cual han sido definidos:

- Creacionales: solucionan problemas de creación de instancias. Nos ayudan a encapsular y abstraer dicha creación.
- Estructurales: solucionan problemas de composición (agregación) de clases y objetos.
- De Comportamiento: soluciones respecto a la interacción y responsabilidades entre clases y objetos.

24 http://es.kioskea.net/contents/genie-logiciel/design-patterns.php3

| | | Propósito | | |
|---|---|---|---|---|
| | | *Creación* | *Estructural* | *Comportamiento* |
| **Ámbito** | *Clase* | Factory Method | Adapter | Interpreter<br>Template Method |
| | *Objeto* | Abstract Factory<br>Builder<br>Prototype<br>Singleton | Adapter<br>Bridge<br>Composite<br>Decorator<br>Facade<br>Flyweight<br>Proxy | Chain of Responsability<br>Command<br>Iterator<br>Mediator<br>Memento<br>Observer<br>State<br>Strategy<br>Visitor |

**Figura 8 Clasificación Patrones de Diseño, los más comunes patrones de diseño**

## *5.1 MVC*

Es un patrón de diseño, que busca separar la lógica de negocios de la lógica de presentación de cualquier solución de software que se esté desarrollando bajo este patrón.

Fue introducido en el año 1979 por *Smalltalk* para el desarrollo de sus aplicaciones que inicialmente se llamo *"Programación de Aplicaciones en Smalltalk-80(TM): Como utilizar Modelo Vista Controlador"*[25].

El problema específico que intenta solucionar este patrón es el siguiente:

> "*Es muy frecuente que se solicite cambio a interfaz. Los cambios a interfaz deberían ser fáciles y efectuados en tiempo de ejecución. El cambio de interfaz no debería de tener consecuencias para el núcleo del código de la aplicación.*"[26]

25 Como usar un Modelo-Vista-Controlador: http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html

26 http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/patrones.html

La solución a este problema establece que el sistema a desarrollar debe ser dividido en tres componentes, que son:

- Modelo (procesamiento de datos)
- Vista (salida de datos)
- Controlador (entrada de datos)

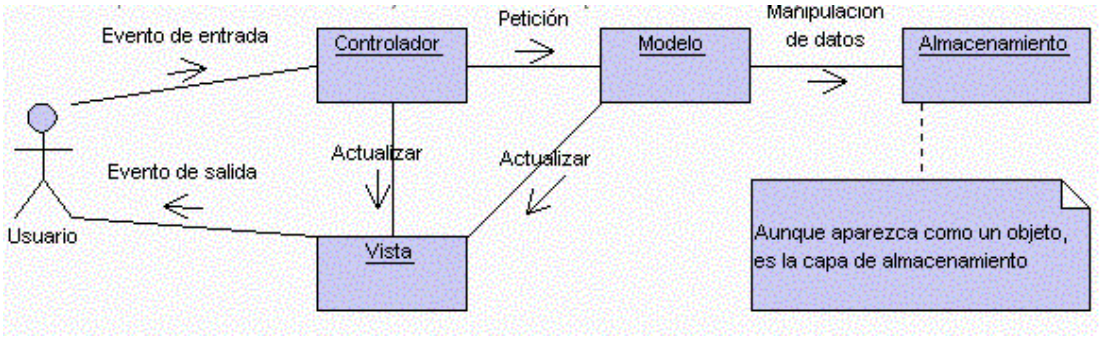


**Figura 9 MVC Diagrama**

### 5.1.1 Modelo

El modelo en una arquitectura *MVC* es el encargado de alterar directamente la estructura de datos de una aplicación, es decir, será el encargado de modificar los registros de una tabla de un motor de base de datos.

### 5.1.2 Vista

La vista es la representación del modelo ante el usuario denominado Interfaz de usuario, es la encargada de presentar de forma visual los datos en un formato adecuado y poder recoger de él información para pasarla luego, vía un controlador, al modelo.

### 5.1.3 Controlador

Un controlador es el encargado de procesar toda la lógica de negocios con los datos recibidos por la vista para luego pasarlos al modelo.

### 5.1.4 *Zend Framework* como marco de trabajo *MVC*

El *Zend Framework* es un marco especializado de trabajo construido para desarrollar aplicaciones Web orientadas a objetos implementado en *PHP* 5 y estructurado con las mejores prácticas orientadas a objetos. Uno de los principales componentes de éste marco de trabajo es el componente "*Layout*" el cual nos permite implementar un ambiente *MVC* para el desarrollo de cualquier aplicación.

En la actualidad, *Zend Framework* es considerada una de las mejores herramientas para consumir y publicar servicios Web de nuestras aplicaciones y su enfoque principal está en ayudar a generar aplicaciones *Web 2.0* modernas y seguras.

Está distribuido bajo la licencia *New BSD License* que permite su uso comercial sin necesidad de un pago por licenciamiento y al igual que *PHP*, es mantenido por la misma compañía, *Zend*.

Dentro del presente proyecto, el *framework* generado se integrará con *Zend Framework*, para ser parte de las librerías del mismo y es también por esta razón que la utilización de *MVC* como patrón de diseño es necesaria en el desarrollo del sistema.

### 5.1.5 Otros *frameworks* MVC

Existen varias implementaciones de *frameworks MVC* con *PHP* que son gratuitas y que pueden ser descargar desde el Internet, que si bien no son tan completos y su documentación no es la adecuada, permiten crear aplicaciones profesionales con *PHP 5* bajo una arquitectura *MVC*, tales como:

- CakePHP.
- Symfony.
- Solar PHP.

## *5.2 Singleton*

*Singleton,* en español, Instancia Única, es un patrón de diseño que está

encargado de crear una sola instancia de un objeto haciendo que la instanciación de un objeto se realice una sola vez y que se garantice el acceso global a esa instancia.

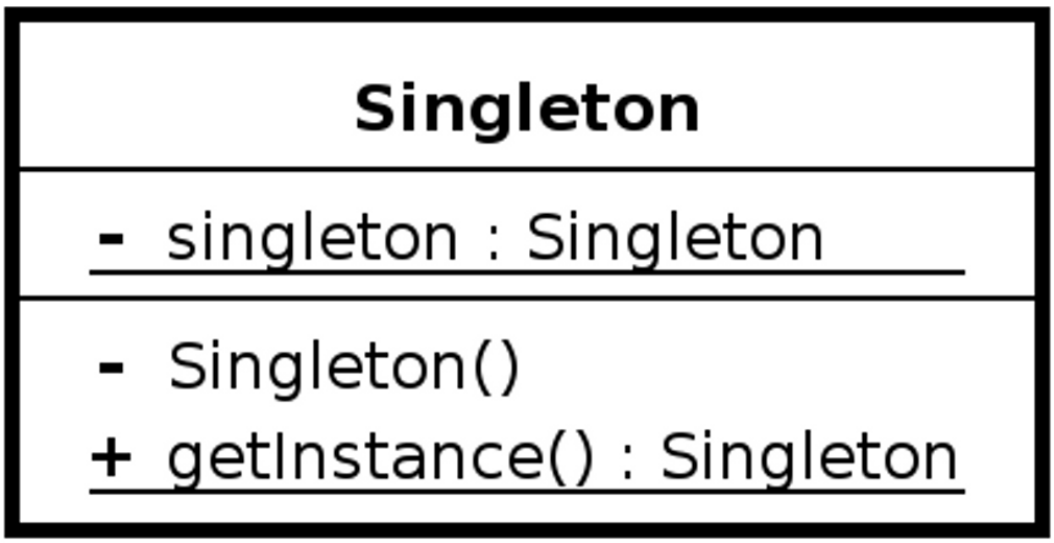


**Figura 10 Diagrama de clases de Singleton**

La primera versión del *framework* utilizará éste patrón para la construcción de una clase de conexión a la base de datos, esto permitirá tener una sola instanciación del objeto de conexión que será global para todos los métodos del *framework*, además que proveerá de métodos adicionales para proveer de funcionalidades para las consultas a realizarse así como elementos de seguridades básicas.

En las futuras iteraciones se mantendrá esta funcionalidad y el uso de este patrón. De esta manera se deja desde la iteración I una conexión funcional y fácil de integrar a nuevos procesos que el *framework* necesite.

# 6 Esquema de Información (*INFORMATION SCHEMA*)

El estándar *ANSI/ISO SQL:2003 parte 11* establece la normativa para el uso de un Esquema de Información y un Esquema de Definición que describen la estructura, la integridad de las relaciones existentes, especificación de seguridades y autorizaciones de una implementación *SQL*[27].

Siendo éste un estándar, muchos de los actuales *RDBMS* lo tienen implementado como parte de sus estructura *SQL* el concepto de *SCHEMATA* o *INFORMATION_SCHEMA*, lugar en donde se guardarán todas las especificaciones antes mencionadas de las estructuras *SQL* presentes en el *RDMBS*, es decir de las bases de datos y sus respectivas tablas.

Para el desarrollo de la iteración 1 del *framework* se ha establecido la utilización del motor de base de datos *MySQL* en su versión 5 utilizando la información almacenada en la base de datos *INFORMATION_SCHEMA* concerniente a relaciones, tipos de datos, tamaño de campos y comentarios de las tablas de una base de datos ejemplo para la generación de los formularios *HTML* de administración.

Estas características hacen que el desarrollo del *framework* para futuras versiones sea extendible para general información para alimentar los formularios desde otros repositorios de datos más allá de *MySQL*.

27 SQL: leguaje de estructurado de consultas

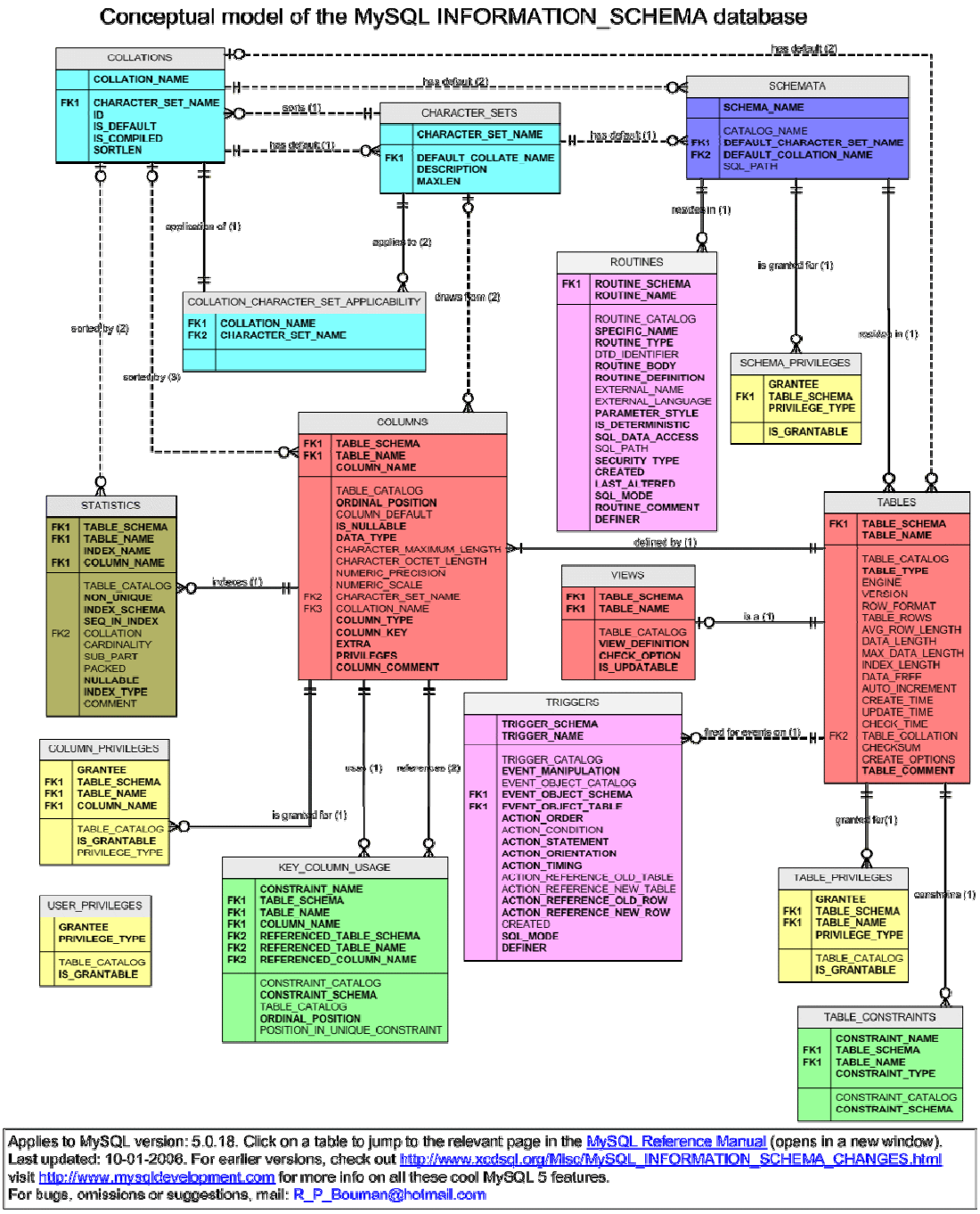


**Figura 11 Information Schema de MySql**

## *6.1 MySQL*

MySQL es un sistema gestor de base de datos relacional Open-Source bajo la licencia *GPL*, fue creado por *David Axmark, Allan Larsson, y Michael Widenius*,

quienes desarrollaron *MySQL* con la idea de que sea un motor muy rápido de consulta de datos y que cumpliese con los estándares propuestos por *SQL* para las bases de datos relacionales.

*MySQL*, en su versión 5, que es la versión que se utilizará para el *framework*, implementa las definiciones del estándar *ANSI/ISO SQL: 2003 parte 11* incluyendo como una base de datos propia de *MySQL* o base del sistema a la base *INFORMATION_SCHEMA* de la cual se trató en el tema anterior.

Las principales características de *MySQL* son:

- Está escrita en *C y C++*.
- Puede funcionar en varias plataformas, es decir es multiplataforma (*Linux, Unix, Windows, Mac OS, Solaris*).
- Posee varios motores de almacenamiento tales como *MyISAM, Memory/heap, MySQL Cluster, Federated, Archive, CSV, Blackhole*e e *InnoDB*, mismo que será utilizado como estándar para la utilización de la primera versión del *framework*.
- *INFORMATION_SCHEMA*.
- Soporte para Procedimientos almacenados.
- Soporte para conexiones seguras bajo *SSL*.

## *6.2 Motor InnoDB*

El motor de almacenamiento *InnoDB* de *MySQL* posee como principal característica el soporte de transacciones de tipo *ACID*[28], soporte para procedimientos almacenados así como integridad referencial entre tablas, una de las características principales que se toma como estándar para el *framework*, puesto que se necesita de tablas que manejen integridad referencial para reflejar las funcionalidades esperadas.

Se distribuye bajo el mismo licenciamiento que *MySQL*, es decir bajo la licencia *GNU GPL*, convirtiendo al motor *InnoDB* de libre distribución además de que es

28 ACID: Atomicidad, consistencia, aislamiento, durabilidad

el motor que se instala por defecto al momento de usar un servidor de *MySQL*.

# 7 DbForm Framework

## *7.1 Análisis de requerimientos*

Se requiere la implementación de un conjunto de métodos que permitan a un programador de software simplificar el proceso de ingreso y manipulación de la información de los registros de una tabla de una base de datos.

En una visión general la solución a ser implementada deberá solventar los siguientes:

- Ser implementado bajo una arquitectura *MVC* propia del *Zend Framework*, ya que es con éste *framework* con el que se desarrolla las aplicaciones empresariales *PHP* dentro de la organización.
- Generar un formulario básico *HTML* que cumple con estándares *XHTML* y que se integrará con el entorno visual conocido también como *"look and feel"* definido en las plantillas de cualquier proyecto, es decir, se integrara con las cabeceras, tamaños de bloques y pies de pagina, así como también con las hojas de estilo *CSS* del mismo.
- Ser lo más eficiente al momento de la generación de los formularios, recibirá la menor cantidad de parámetros y proveerá de varios métodos que devolverán cierto tipo de información de la tabla que se quiera administrar.
- De existir integridad referencial entre la tabla a ser generada con alguna otra tabla, la solución será capaz de identificar esta relación y hacer parte del *HTML* del formulario generado ésta relación existente.
- Proveer de métodos de inserción de los datos ingresados por el usuario con una mínima de validaciones.
- Proveer de métodos de edición de los datos ingresados por el usuario con una mínima de validaciones.
- Identificar automáticamente los tipos de datos *HTML* en función de los tipos de datos almacenados en las tablas de la base de datos, así como también, utilizar los comentarios de las tablas y de los registros como los

textos a ser presentados como nombres en los formularios *HTML*.

- Debe existir una implementación para la encriptación de los datos que serán enviados entre el cliente y el servidor al momento de la ejecución de la acción de enviado de datos o *Submit* del formulario así como para encriptar valores que vayan a ser pasados por *URL*.
- Debe existir la documentación necesaria de la codificación de toda la solución, todo esto bajo algún estándar de codificación de *PHP* y mejores prácticas de programación.

Es de esta forma que se puede construir correctamente el documento de Historia de Usuario así como el documento de Plan de Iteraciones, esta información se ve reflejada resumida pero concisa en las diferentes historias. Las reuniones con el cliente apoyan el desarrollo de este documento. Para mayor información de la historia de usuario del sistema y las historias que lo conforman referirse al Anexo A (Historia de Usuario) y Anexo D (Plan de Iteraciones).

Adicionalmente hay que dejar claro que otros documentos fueron generados, todos ellos en base al de Historia de Usuarios. Se pretende de esta manera dejar claro los presupuestos y tiempos que implica el desarrollo de la solución. De esta manera se determinó que el costo total de la aplicación será de 2,655.77 dólares, y que el tiempo para entregar el sistema será de 1.12 meses o 4.29 semanas. Para mayor información referirse al Anexo B (Estimación de Costos y Recursos).

## *7.2 Riesgos y limitaciones*

El proyecto presenta los siguientes riesgos:

- Cambios en la estructura de las tablas de *INFORMATION_SCHEMA*: en el caso que se produzca un cambio en la estructura de la información de *INFORMATION_SCHEMA*, se deberá modificar y adaptar el presente

desarrollo a estos nuevos cambios, teniendo en cuenta que tantos los costos del desarrollo y los tiempos pueden variar.

- Cambios en la estructura *MVC* del *Zend Framework*: estos cambios implicarían que se deban modificar las salidas de los datos para que puedan ser visualizadas correctamente en las interfaces de usuario o pantallas. Si un cambio de este tipo ocurre se deberá modificar las platillas de salida de datos. Lo que incurrirá en cambios de tiempos y costos en el desarrollo.
- Que *Zend Framework* implemente en alguna de sus actualizaciones una funcionalidad similar a la propuesta con **DbForm**: esto afectaría al sistema haciéndolo menos competitivo en relación a la solución propuesta por *Zend Framework*; sin embargo éste proyecto sigue siendo un aporte a la comunidad y por ende es factible terminar su desarrollo y publicarlo al Internet para mejoramientos continuos donde se deberá estudiar otros beneficios y aplicaciones que se puedan brindar al desarrollador Web.

El sistema pretende cumplir a cabalidad los requerimientos de usuario expuestos en el Anexo A (Historias de Usuario), donde la solución final permitirá realizar las tareas antes expuestas. A pesar de ello se debe tomar en cuenta que proporcionar una completa y optimizada funcionalidad conlleva a un mayor tiempo de desarrollo y a la generación de varias versiones del prototipo inicial; es por esto que el sistema propuesto en el presente trabajo de titulación tiene las siguientes limitantes:

- El sistema permitirá una reducida creación de tipos de campos *HTML*, que cumplan con las siguientes características:
    - Fecha,
    - Cajas de área de texto.
    - Campos de contraseña.
    - Listas desplegables.
    - Campo de texto.
- No permitirá el acceso a otros repositorios de datos diferentes a *MySQL*.
- No se puede conectar a ningún otro motor de *MySQL* que no sea *InnoDB*.
- Las tablas que no posean correctos comentarios tanto para sus registros como para sus nombres no serán correctamente desplegados.

## *7.3 Diseño*

En función de los requerimientos propuestos, la solución intenta solventar todo lo expuesto implementándolo esta primera iteración del *framework* generador de formularios *HTML Web 2.0*, se establece que el marco de trabajo orientado a objetos debe ser codificado en *PHP* 5, cuya labor será proveer a un programador Web de clases y métodos para la generación rápida y ágil de formularios Web de las tablas *MySQL* de un proyecto realizado bajo una arquitectura *MVC* manejada con el *Zend Framework*, marco de trabajo igualmente implementado en *PHP 5* y orientado a objetos.

Para esta primera versión se utilizara la base de datos *INFORMATION_SCHEMA* de *MySQL*; ésta base de datos es donde se encuentran todos lo meta-datos que *MySQL* posee de sus bases activas. Es de éste repositorio de datos y por medio del *framework*, denominado **DbForm**, que se tomará los datos para él la generación de los métodos que describen los elementos de un formulario Web. Una de las ventajas de usar

*INFORMATION_SCHEMA* es que al ser esto un estándar *SQL* está presente en muchos y más populares *RDMBS*, haciendo que **DbForm**, sea extendible hacia otros motores de base de datos más allá de *MySQL*, otra notable ventaja es que el usuario de conexión que se debe usar en un motor de base de datos para acceder a la información provista es de *SELECT*, esto quiere decir que no se necesitan permisos especiales para ningún usuario de la base de datos para obtener información y generar los formularios. La generación de los formularios de las tablas propuestas contará en un inicio de tres clases definidas en la primera iteración de **DbForm** establecidas en el Anexo G (Diagrama de Clases).

Existirá una clase de conexión a *MySQL* implementado el patrón de diseño *Singleton*, esta clase proveerá un objeto global de conexión para todos los métodos de consulta del **DbForm**. Además proveerá de métodos para inserción y manipulación de arreglos enviados por métodos *GET* o *POST*[29], que serán los datos ingresados por el usuario en los formularios generados.

Se proveerá de una clase de encriptación y des encriptación usando la librería *mcrypt*[30] de *PHP*, potente librería de encriptación que permite que se puedan elegir varios algoritmos de encriptación tales como *DES*, *TripleDES*, etc.

Se integrará fácilmente con el *Zend Framework*, trabajando como un modelo del mismo, haciendo que todos los métodos framework estén siempre disponibles.

Se implementará una clase con la capacidad de apoyarse en la anteriores descritas y que genere dinámicamente la estructura *HTML* de la tabla que se éste analizando.

En la siguiente figura se muestra la interacción de las clases del *framework* **DbForm:**

[29] GET & POST: métodos de envío de datos por la Web.
[30] Mcrypt: librería de encriptación de PHP.

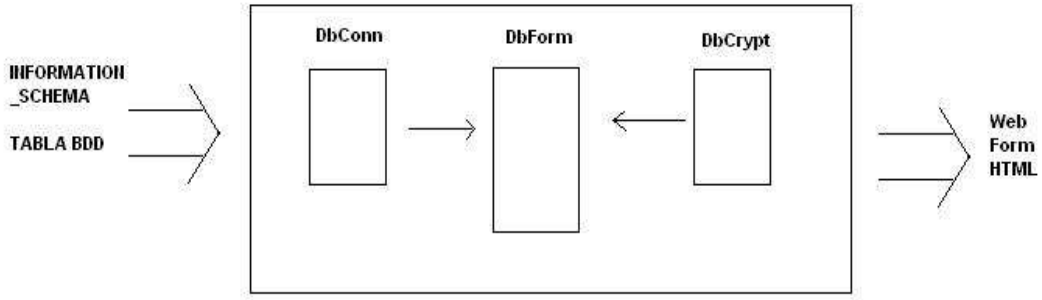


**Figura 12 Flujo entrada y salida DbForm**

En conjunto, la figura anterior presenta la interacción existente entre las clases del *framework* y como es el flujo de entrada y salida de los datos existentes en *INFORMATION_SCHEMA* y en la cualquier tabla de una base de datos, es decir, la clase DbForm toma los datos necesarios de *INFORMATION_SCHEMA* y de los datos de la tabla que haya sido enviada como parámetro, para, conjuntamente con DbConnection y DbCrypt, procesar ésta información y devolver un formulario *HTML* que muestre de manera correcta toda la estructura de la tabla analizada.

El siguiente diagrama muestra lo métodos y propiedades de cada una de las clases propuestas para ésta primera iteración:

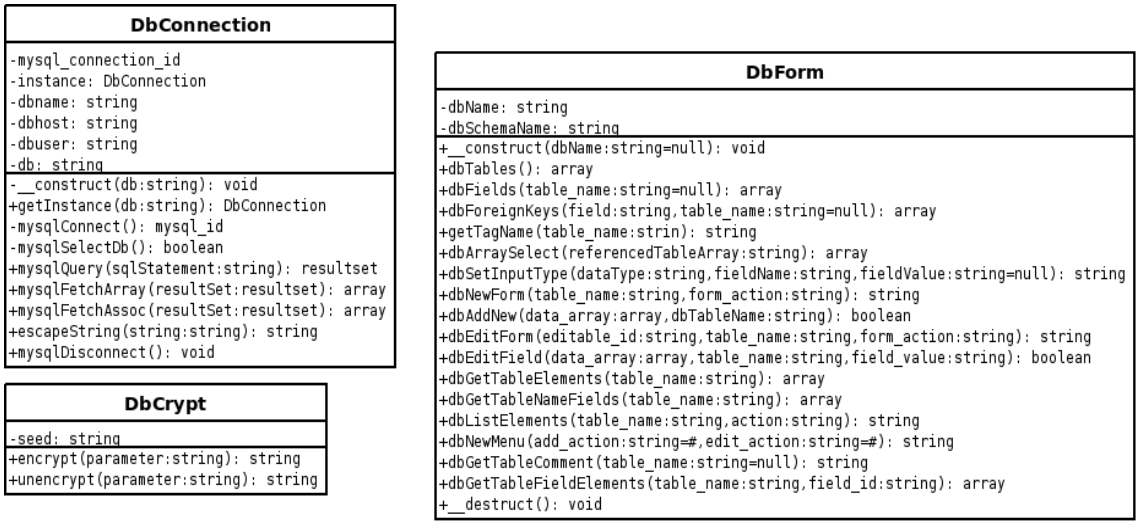


**Figura 13 Métodos y propiedades DbForm**

La documentación del *framework* y las librerías mostradas en la figura anterior se encuentra mejor explicada en el Anexo F (Documentación del Framework); documentación que será realizada con *PHPDocumentator* también conocido como *PHPDoc*, y las mejores prácticas de programación a ser implementadas están dictadas por *Zend Framework*.

La facilidad que permite *PHPDoc* el momento de generar la documentación de un sistema es la razón por la cual se usa este estándar y se lo recomienda para aplicaciones desarrolladas en *PHP*. La utilidad provista es tan amplia que permite generar diferentes tipos de documentación desde una interfaz Web o desde una herramienta de desarrollo que esté integrada correctamente con este estándar. El documento que se obtiene es la Documentación del *Framework*, que funcionan para este proyecto como un manual de usuarios y de implementación para que pueda ser usado por la comunidad *Open-Source* o quien esté interesado en el mismo.

## *7.4 Implementación*

El *framework* **DbForm** en su primera iteración consta de tres librerías

encargadas de cumplir con los requerimientos y las Historias de Usuarios previamente mencionadas en este trabajo de titulación y en el Anexo A (Historias de Usuarios):

- DbForm: librería encargada de realizar las gestiones administrativas de la información que se almacena en las diferentes tablas de una base de datos *MySQL*, que cumple con los requerimientos del sistema **DbForm** y basándose en los estándares estipulados en el Anexo H (Requisitos de Configuración y Limitaciones). La característica principal de esta librería es que provee métodos fáciles de acceder, que devuelven una estructura *HTML* válida y que reflejan totalmente todos los campos de la tabla que se ha decido administrar.
- DbCrytp: librería encargada de proporcionar seguridad a los datos por medio de algoritmos de encriptación, propios de *PHP,* que garantizan la confiabilidad de la información para que esta pueda ser manipulada. Es el apoyo para que la librería DbForm encripte los datos que deben ser asegurados previo a enviarlos al servidor Web para que sean procesados.
- DbConnection: librería que implementa el patrón *Singleton*, esta librería provee todos los métodos para la conexión y manipulación de una base de datos *MySQL*, apoyo para la librería DbForm al momento de generar las consultas necesarias al motor de base de datos.

Las clases descritas en los puntos anteriores claramente muestran que se ha cumplido con los requisitos establecidos para en el Anexo D (Plan de Iteraciones), requerimientos planteados para la primera iteración de éste trabajo de titulación.

Clases descritas individualmente en los siguientes diagramas:

### 7.4.1 DbForm

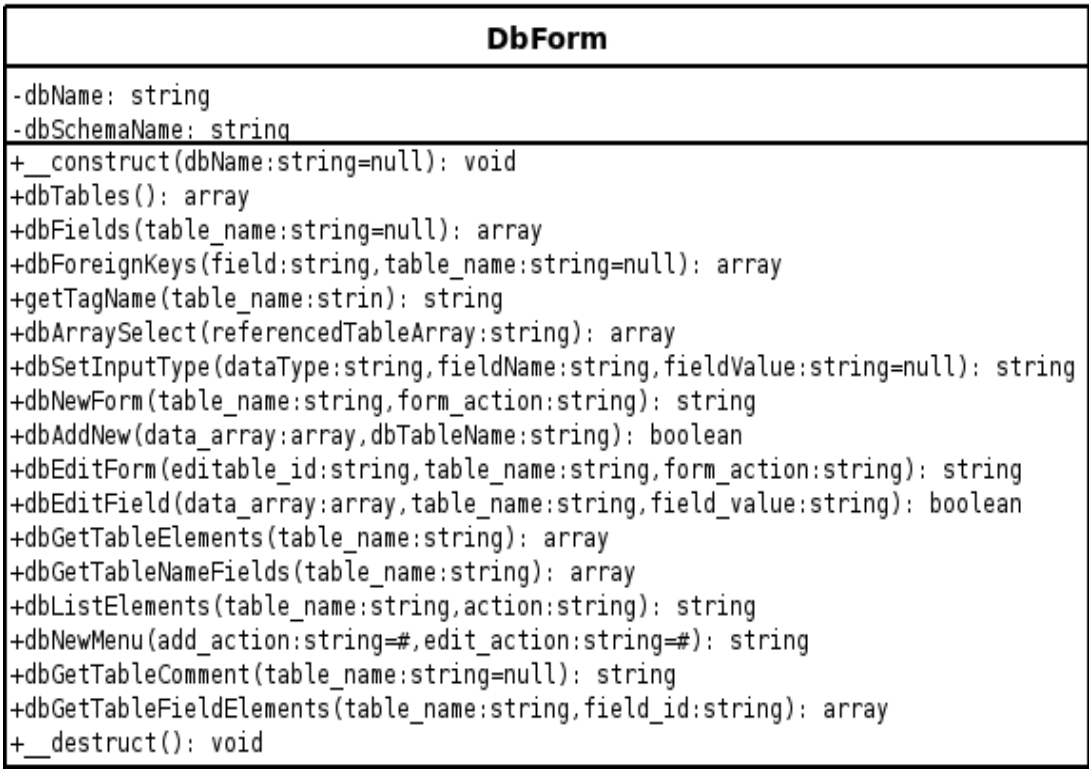


**Figura 14: Diagrama de clases DbForm**

Los métodos de esta clase proveerán las siguientes funcionalidades:

- __construct: constructor de la clase, encargada de inicializar las variables.

- dbTables(): método encargado de devolver un arreglo asociativo con la información de todas las tablas existentes en una base de datos. Esta información es obtenida extrayendo los datos desde tabla *TABLES* de *INFORMATION_SCHEMA.*

- dbFields(): método encargado de devolver un arreglo asociativo con la información de todos los campos de una tabla. Estos datos son obtenidos de *INFORMATION_SCHEMA.* Esta información es obtenida extrayendo los datos desde tabla *COLUMNS* de *INFORMATION_SCHEMA.*

- dbForeignKeys(): método encargado de devolver un arreglo asociativo indicando la tabla referenciada y el campo referenciado para un campo y una tabla dados, es decir devolverá el nombre y la clave primaria de una tabla que esté en integridad referencial con otra. Esta información es obtenida extrayendo los datos desde las tablas *REFERENCED_TABLE_NAME*, *REFERENCED_COLUMN_NAME* de *INFORMATION_SCHEMA.*

- getTagName(): método encargado de identificar cual es el campo descriptivo en una tabla de una base de datos, basados en la marca del comentario y devuelve el nombre del campo.

- dbArraySelect(): método encargado de devolver la representación *HTML* para las relaciones existentes entre dos tablas.

- dbSetInputType (): método encargado de procesar un campo de una tabla, en función del tipo de dato del campo en la base y devolver el código *HTML* del mismo. La lógica de la representación está incluida dentro del mismo método.

- dbNewForm(): método que genera un formulario *HTML* para una tabla dada y una acción para el formulario, identifica automáticamente tipos de datos y relaciones existentes para la tabla haciendo uso del los métodos descritos en éste *framework*.

- dbAddNew(): método encargado de añadir un nuevo registro a la base de datos, recibe como parámetros un arreglo que proviene de un formulario Web y el nombre de la tabla donde se realizará la inserción.

- dbEditForm(): método encargado de generar un formulario *HTML*

de una tabla dada y un arreglo de sus campos , es decir crea la interfaz de edición de estos datos.

- dbEditField(): método encargado de editar todos los elementos de un registro de una tabla dada junto con el arreglo de datos correspondiente.

- dbGetTableElements(): método que devuelve un arreglo asociativo con todos los campos de una tabla dada.

- dbGetTableNameFields(): método público que devuelve un arreglo asociativo con los nombres de los campos de una tabla dada. Estos datos son obtenidos de *INFORMATION_SCHEMA* de la tabla *COLUMNS* de la misma.

- dbListElements(): método encargado de generar una interfaz *HTML* donde se listan todos los registros de una tabla dada, previa a su edición.

- dbGetTableFieldElements(): método encargado de devolver un arreglo asociativo con todos los elementos de un registro para una tabla dada y un identificador del mismo.

- dbNewMenu(): método encargado de crear una interfaz *HTML* listando todas las tablas de una base de datos, esta interfaz posee las acciones de creado y editado de los datos de las mismas tablas.

- dbGetTableComment(): método encargado de devolver el comentario de una tabla dada. Estos datos son obtenidos de *INFORMATION_SCHEMA* de la tabla *TABLE_COMMENT.*

- __destruct(): destructor de la clase.

### 7.4.2 DbConnection

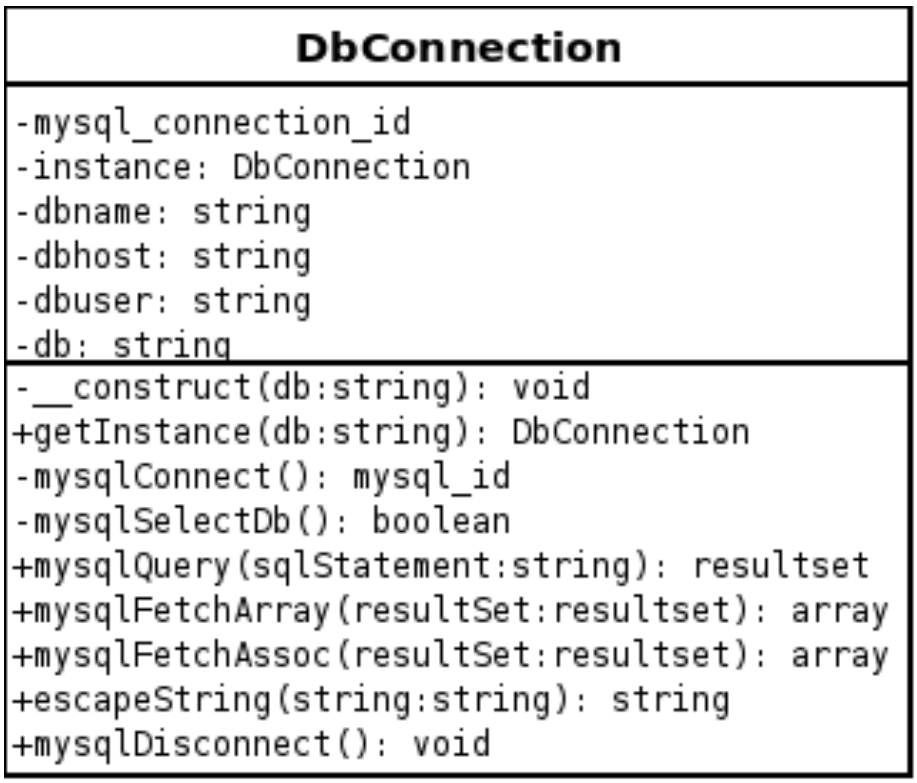


**Figura 15: Diagrama de clases DbConnection**

Los métodos de esta clase proveerán las siguientes funcionalidades:

- __construct(): constructor privado de la clase, inicializa las variables de la clase.

- getInstance(): instancia estática de la clase provista por el patrón *Singleton*.

- mysqlConnect(): método privado que provee una conexión a un

servidor *MySQL*.

- mysqlSelectDb(): método privado que selecciona una base de datos en *MySQL.*

- mysqlQuery(): método público encargado de ejecutar una consulta *SQL* contra un servidor *MySQL* y devolver un resultado para esa consulta.

- mysqlFetchArray(): método encargado de devolver un arreglo numérico de una consulta *SQL*.

- mysqlFetchAssoc(): método encargado de devolver un arreglo asociativo del resultado de una consulta dada.

- escapeString(): método encargado de validar una cadena de caracteres con el método de *MySQL mysql_escape_string().*

- mysqlDisconnect(): método encargado de desconectar una conexión activa de *MySQL.*

### *7.4.3 DbCrypt*

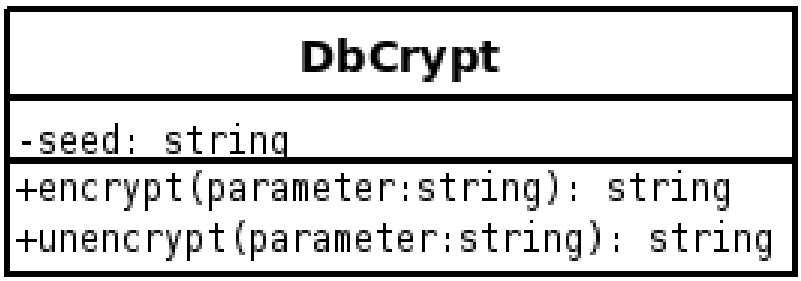


***Figura* 16: Diagrama de clases DbCrypt**

Los métodos de esta clase proveerán las siguientes funcionalidades:

- encrypt(): método que se encarga de encriptar una cadena de caracteres con el algoritmo *TripleDES*[31], haciendo uso de la librería *Mcrypt* de *PHP* antes descrita.

- unencrypt(): método encargado de desencriptar una cadena de caracteres para el algoritmo *TripleDES*, haciendo uso de la librería *Mcrypt* de *PHP* antes descrita.

### 7.4.4 Integración de "DbForm" con el *Zend Framework*

Para la integración del **DbForm** con el *Zend Framework* se propone el siguiente ejemplo para la demostración de la solución:

Requerimientos para el ejemplo:

- Una base de datos de ejemplo, para esta demostración, se creará una pequeña base que consta de 3 tablas todas ellas como *InnoDB* en un servidor de base de datos *MySQL*. Las tablas a ser administradas son: "Ciudad", "País" y "Estado", estas tablas pretenden simular las tablas de una base de datos de un sistema real:
    - Ciudad
        - En la figura mostrada a continuación se describe una estructura ejemplo para la tabla denominada "Ciudad", mostrando todos sus atributos como tipo de campos, longitudes de los mismos, etc., así como los cometarios tanto para la tabla como para

[31] http://www.alegsa.com.ar/Dic/triple%20des.php

cada unos de los registros, importantes datos para la utilización de **DbForm.**

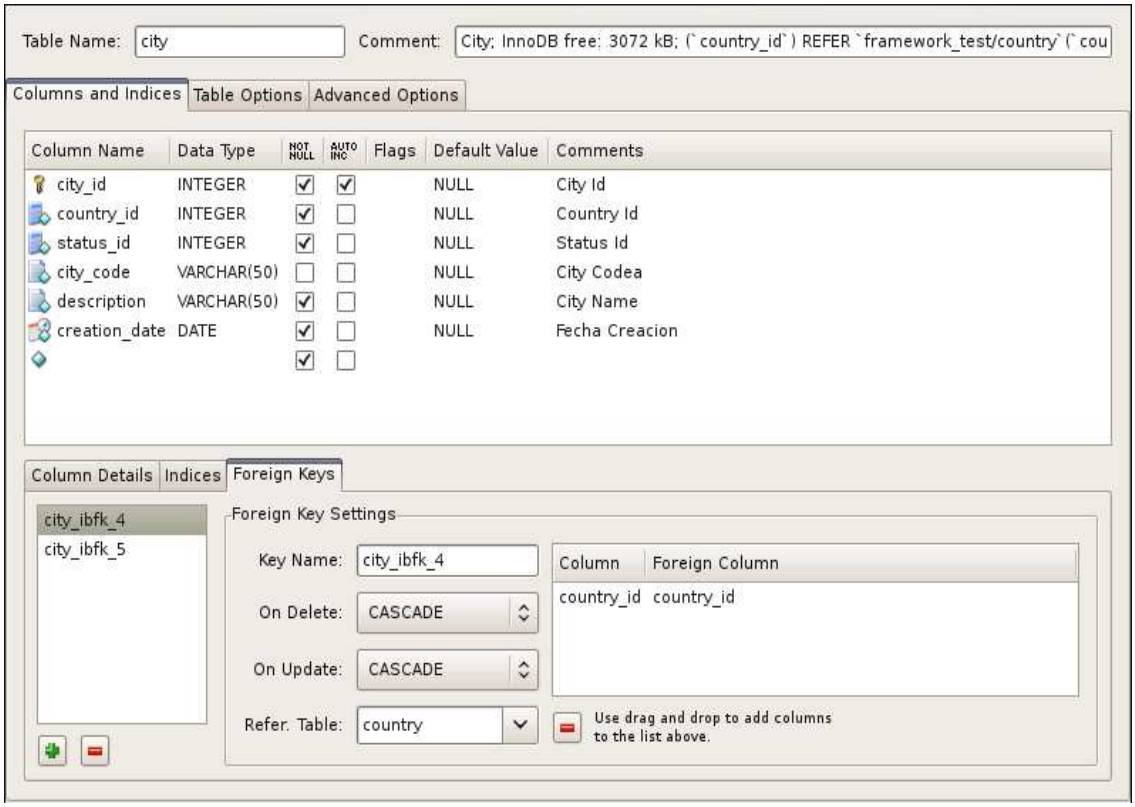


**Figura 17: Tabla de ejemplo "Ciudad"**

- País
  - En la figura mostrada a continuación se describe una estructura ejemplo para la tabla denominada “País”, mostrando todos sus atributos como tipo de campos, longitudes de los mismos, etc., así como los cometarios tanto para la tabla como para cada unos de los registros, importantes datos para la utilización de **DbForm.**

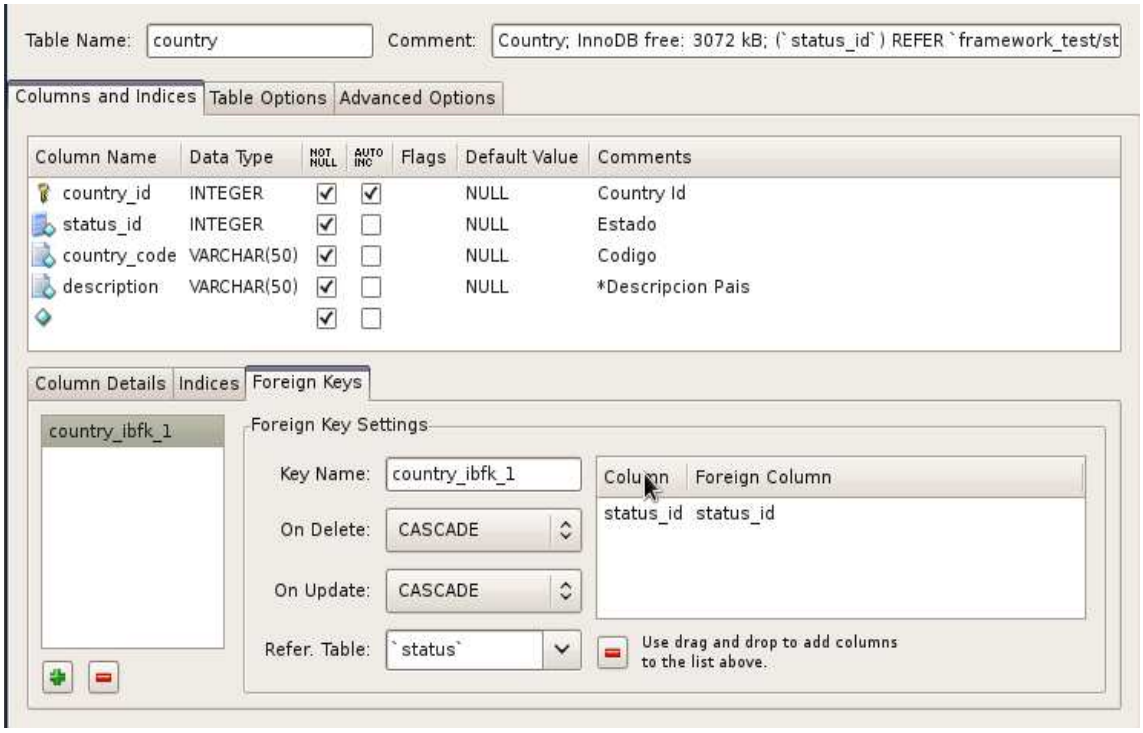


**Figura 18: Tabla de ejemplo "País"**

- Estado
  - En la figura mostrada a continuación se describe una estructura ejemplo para la tabla denominada “Estado”, mostrando todos sus atributos como tipo de campos, longitudes de los mismos, etc., así como los cometarios tanto para la tabla como para cada unos de los registros, importantes datos para la utilización de **DbForm.**

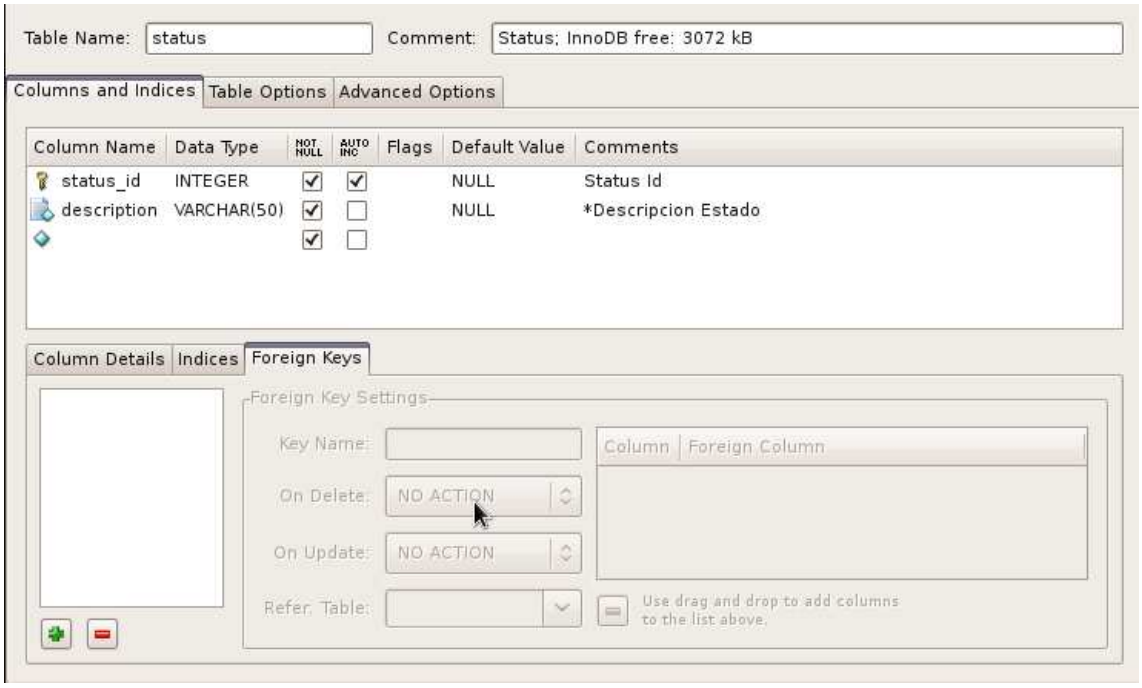


**Figura 19: Tabla de ejemplo "Estado"**

- El *Zend Framework*, para esta versión se utilizará la versión 1.5 del mismo.
- Crear un ambiente *MVC* con el *Zend Framework* junto con la estructura de archivos, para este punto se seguirá la guía propuesta por el manual de implementación *MVC* del *Zend Framework*,[32] para mayor información sobre la implementación de un ambiente *MVC* con éste *framework*, referirse a la bibliografía del presente trabajo, sección *Zend Framework* además de que la estructura propuesta para su correcta integración se muestra claramente en la figura 20.
- Esta guía ayudará en la implementación de la estructura de archivos, la arquitectura *MVC* y para crear las vistas de las interfaces de usuario donde se mostrarán los formularios generados.

[32] http://www.zendframework.com/docs/quickstart/set-up-the-project-structure

Estructura de archivos y arquitectura *MVC*:

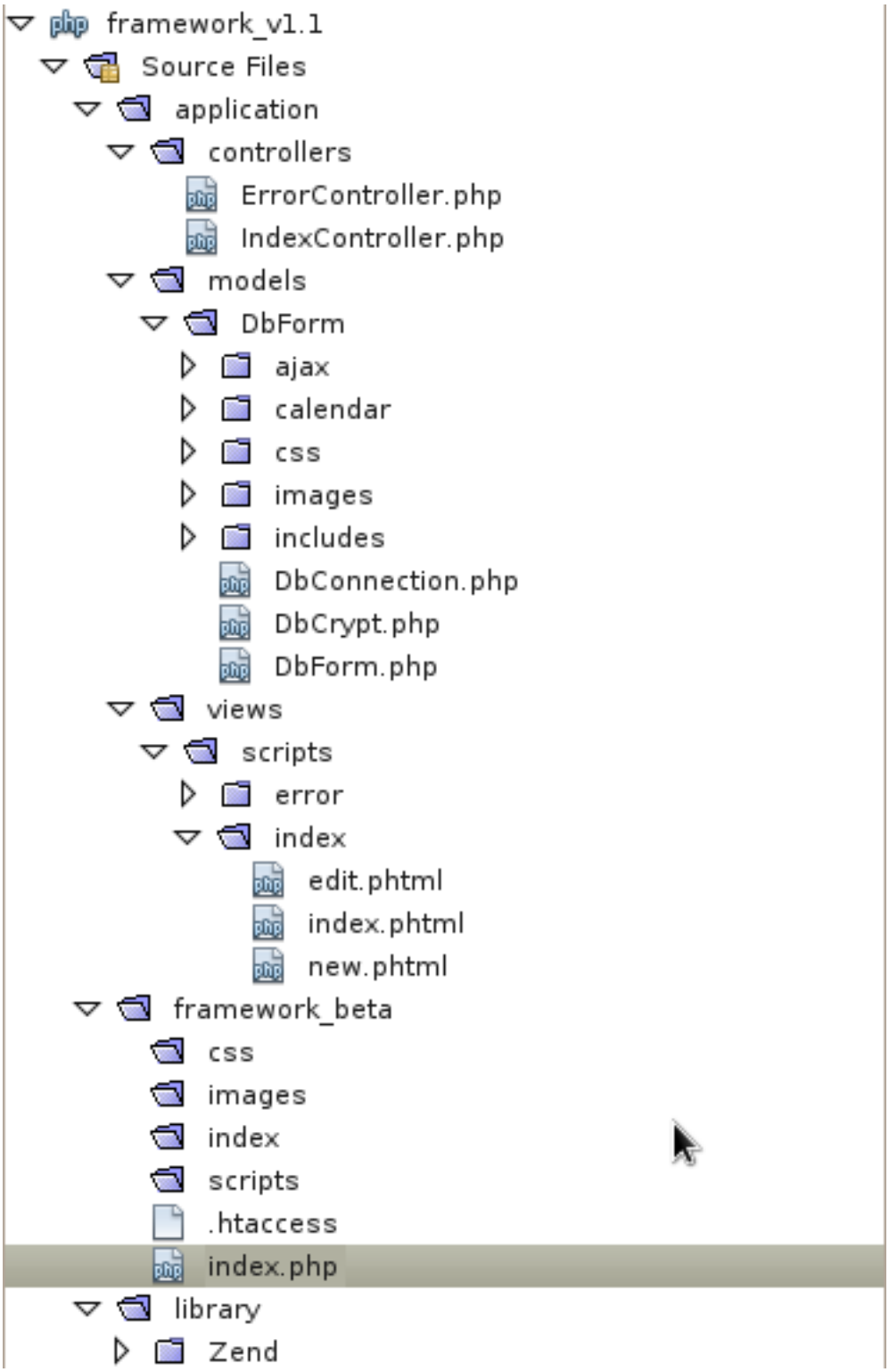


**Figura 20: Estructura de archivos MVC**

- Código de Ejemplo: Se va a dividir el código las tres acciones que un controlador del *Zend Framework* va a manejar, estas acciones son: indexAction, newAction, editAction que un programador deberá implantar al momento de instanciar un objeto de tipo **DbForm**:
    - indexAction

```
20    public function indexAction()
21    {
22
23        //todo
24        $add_action  = $this->_request->getBaseUrl() . "/index/new"; // path for the add action.
25        $edit_action = $this->_request->getBaseUrl() . "/index/edit"; // path for the edit action.
26
27        $form = new DbForm(); // DbForm Object instance
28
29        $menu  = $form->dbNewMenu($add_action, $edit_action); // HTML menu list of all tables in specified database.
30
31        $this->view->form  = $menu; //load HTML menu into view
32    }
```

**Figura 21: Controlador indexAction()**

Aquí se invoca al método dbNewMenu y el resultado obtenido en *HTML* en la vista es el siguiente:

| No. | Table | Action |
|---|---|---|
| 1 | City | Add Edit |
| 2 | Country | Add Edit |
| 3 | Status | Add Edit |

**Figura 22: Resultado HTML de indexAction()**

- newAction

```
34      public function newAction ()
35      {
36          //todo
37          $form = new DbForm();
38          $postAction = $this->_request->getParam('New', 0); //action sent by the form
39          $table_name =  $this->_request->getParam('value', 0); //table name
40          if (empty($postAction)){ // if no form action
41
42              $action = ""; // html form action
43              $arr  = $form->dbNewForm($table_name, $action); // create new form
44              $this->view->arr = $arr; // send html form to view
45
46          } else { //if action: insert values into table
47              $post = $_POST; // form data; MUST BE VALIDATED!!!!!!
48              $table_name =  $this->_request->getParam('value', 0); //table name
                if($message = $form->dbAddNew($post, $table_name)){
50                  $this->view->message = "Field inserted....!!";
51              } else {
52                  $this->view->message = "Field not inserted....!!";
53              }
54          }
55      }
```

**Figura 23: Controlador newAction()**

Aquí se invoca al método dbNewForm que creará el formulario *HTML* de la tabla que se recibe como parámetro en la vista de

edición y el resultado es el siguiente:

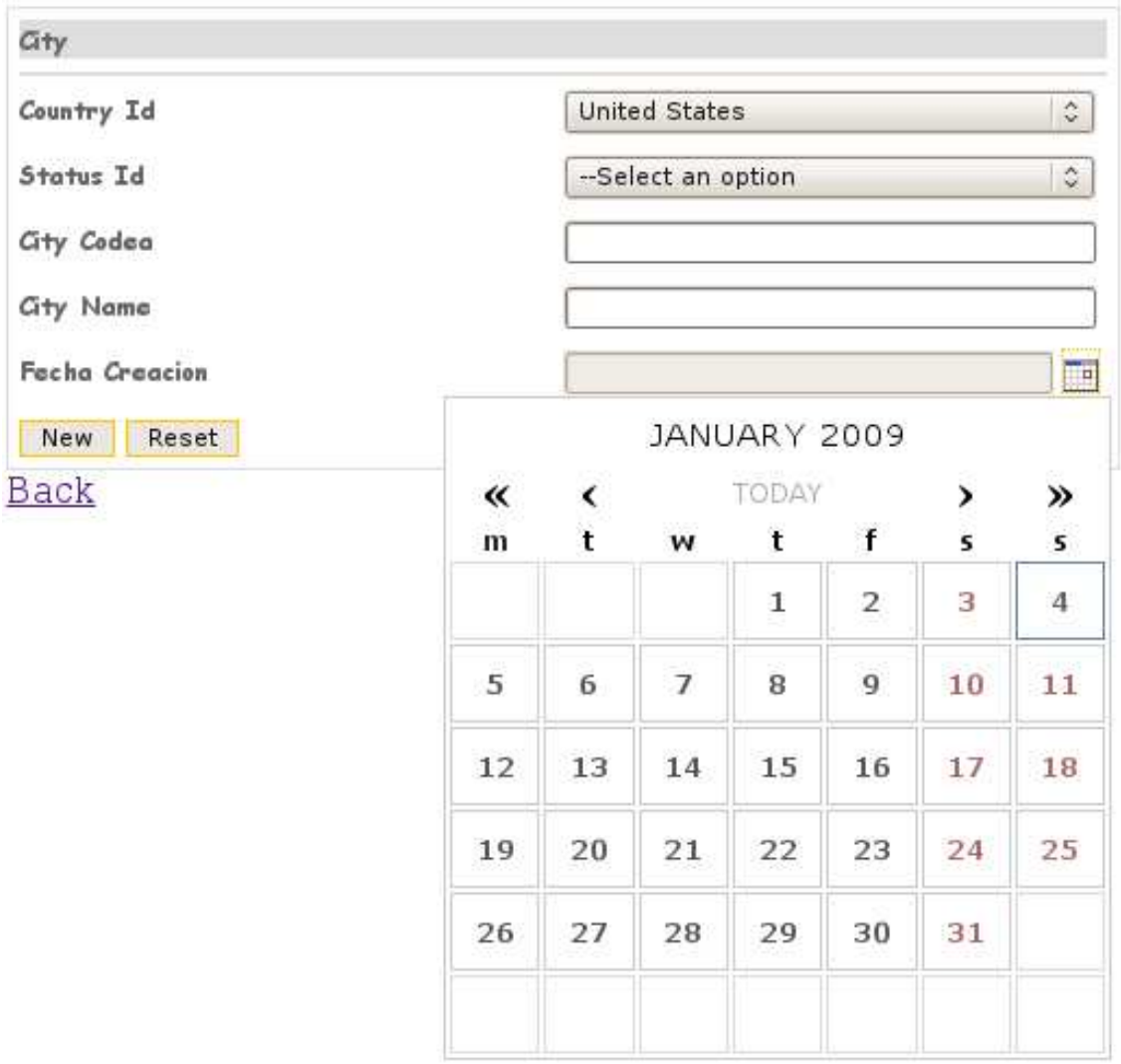


**Figura 24: Resultado HTML newAction()**

- editAction

```
57      public function editAction()
58      {
59          $table_name = $this->_request->getParam('value', 0);
60          $form = new DbForm();
61          $this->view->title = $form->dbGetTableComment( $table_name );
62          $postAction = $this->_request->getParam('Edit', 0); //action sent by the form
63          if ( $postAction ){
64              $post = $_POST; // form data; MUST BE VALIDATED!!!!!
65              if($message = $form->dbEditField($post, $table_name)){
66                  $this->view->message = "Field edited....!!";
67              } else {
68                  $this->view->message = "Field not edited....!!";
69              }
70          }else{
71              if( $table_name ){
72                  $grid = $form->dbListElements( $table_name ); //load grid
73                  $this->view->arr = $grid;
74              }
75              $table_id = $this->_request->getParam('id', 0);
76              if ( $table_id && $table_name){
77                  $action = "#";
78                  $arr = $form->dbEditForm($table_id, $table_name, $action);
79                  $this->view->arr = $arr;
```

**Figura 25: Controlador editAction()**

En estas líneas de código se utiliza el método dbEditForm para crear el formulario de edición de un registro, el resultado es la siguiente figura:

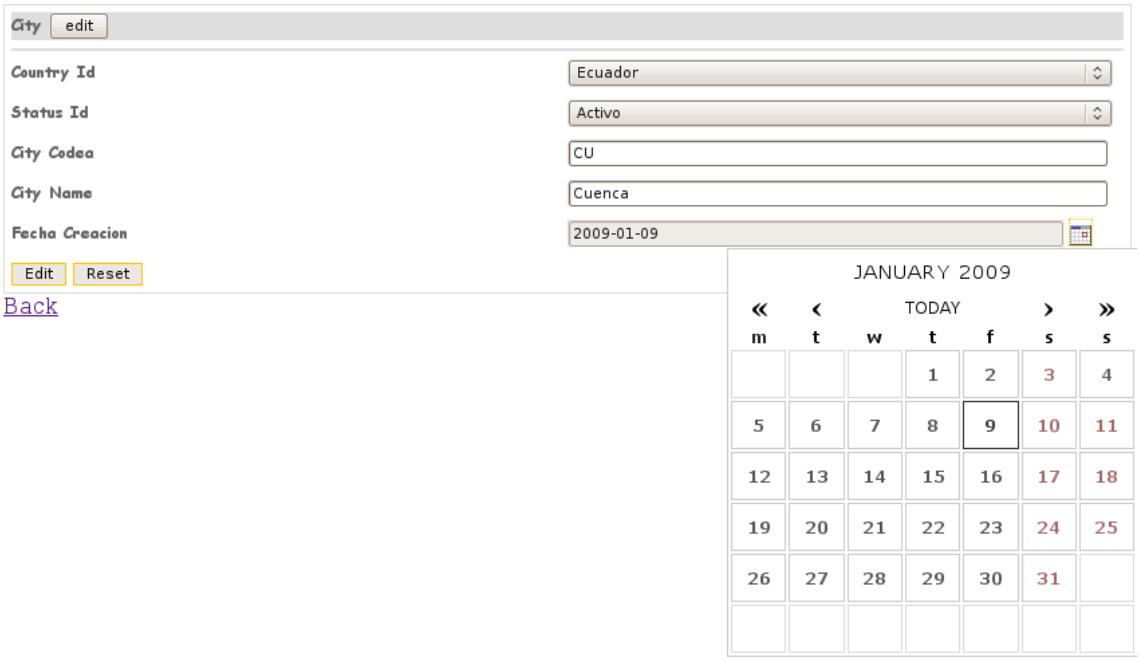


**Figura 26: Resultado HTML editAction()**

# 8 Conclusiones y recomendaciones

## *8.1 Conclusiones*

- La definición de *Web 2.0* es de suma importancia para la construcción de la solución final; es gracias al tener claro el concepto y las tecnologías que se utilizan que se pudo definir los estándares a seguir; y de esta manera las soluciones provistas por la utilización del mismo son óptimas y respetan los criterios de la *Web 2.0*.
- La utilización del patrón de diseño *MVC* es la razón por la cual **DbForm** se integró de manera correcta con el *Zend Framework*, con el fin que las arquitecturas sean compatibles, que acoplarlas no tome mucho tiempo y se lo pueda realizar correctamente. El resultado final es una funcionalidad adicional al *Zend Framework* que permite agilizar procesos muy comunes entre desarrolladores.
- La solución generada en el presente proyecto, fue desarrollada bajo un lenguaje de programación *Open-Source*, *PHP 5*; mismo que permitió la creación de una herramienta basada en un modelo orientado a objetos que, entre otras cosas, es de gran apoyo para escalar la solución a futuras versiones, reutilizar lógica y código fuente existes y también permite la integración con el *Zend Framework*.
- Las pruebas realizadas demuestran que la generación de formularios extraídos de bases de datos relacionales *MySQL* que se rijan a los parámetros requeridos por **DbForm**, son óptimos, confiables y reducen exponencialmente el trabajo que normalmente se tiene que realizar para generar soluciones similares.
- La solución generada es de gran apoyo para la comunidad de *Zend*, su aporte ha permitido que un grupo de desarrolles optimice su trabajo y reduzca horas de trabajo con el correcto uso de la misma; se espera que

este **DbForm** se distribuya a un número más alto de personas dentro de esta y otras comunidades similares a medida que se difundan sus resultados por las mismas.

- Se puede concluir que la utilización de una metodología ágil de desarrollo de software puede optimizar la construcción de herramientas de apoyo para programadores, ayudando a la disminución de tiempos al momento de la implementación de partes de una solución completa de software y distribuyendo el trabajo para que se documente menos y desarrolle más, haciendo que la documentación generada sea de calidad y simple.

## *8.2 Recomendaciones*

- Se recomienda que se realice un estudio de tecnologías y conceptos de *Web 2.0* para realizar un desarrollo que se acople a la misma, esto es importante tomar en cuenta ya que son un grupo extendido de técnicas y soluciones que han parametrizado la *Web 2.0* para una correcta evolución de la misma.
- Es recomendable usar un patrón de diseño que permita realizar de mejor manera la resolución un proyecto de software, esto se refleja en el presente proyecto ya que al usar *MVC* se pudo acoplar con *Zend Framework* y ser parte de una solución más grande. De igual manera al uso del patrón *Singleton* para el apoyo en la conexión a la base de datos demuestra ser de gran ayuda tanto para el presente como para futuros desarrollos debido a la transportabilidad que permite.
- Es recomendable que el momento de crear una nueva funcionalidad para un *Framework* ya existente se utilice el mismo lenguaje de programación con el que éste fue desarrollado, el mismo criterio hay que tenerlo en cuenta para los patrones y modelo de gestión que se provea.
- Las pruebas de un sistema y la correcta documentación de las mismas son necesarias en un sistema para que tenga un reconocimiento de la

validez de los resultados que brinda. La implantación e interacción del *framework* desarrollado demuestran su efectividad para resolver el problema de la generación de formularios dinámicos en base a bases de datos *MySQL* y la reducción considerable del tiempo invertido en esta tarea.

- Se recomienda ser partícipe de una comunidad de software y contribuir con la misma, con el fin de generar soluciones que apoyen el trabajo de otros desarrolladores, obtener un reconocimiento por ello y finalmente retribuir al mundo de *Open-Source* ideas, funcionalidades o aplicaciones, así como se extrajo de ella lo mismo inicialmente.
- La utilización de una metodología rápida para el desarrollo de software es recomendable gracias a que con ella se puede tener un apoyo y guía a seguir para desarrollar soluciones de software más rápido y eficientemente. En el presente proyecto, *XP*, demostró ser la metodología indicada para en un corto tiempo generar un *framework* extensible a uno más grande de manera correcta y óptima.
- Se recomienda, para las siguientes versiones del *framework*, la utilización de más funcionalidades provistas por *Zend Framework*, con el fin de ser cada vez más parte del mismo y su integración sea más sencilla y correcta. Un ejemplo de ello sería eliminar la clase *singleton* para la conexión a la base de datos y utilizar la clase *Zend_Db*, que es propia del *Zend Framework*; ésta clase ya implementa un patrón singleton para su instanciación y además proporciona los adaptadores necesarios para la conexión a otros *RDBMS* diferentes a *MySQL*.
- Se recomienda que se aumente funcionalidades tipo *AJAX* con el fin de ser de mayor agrado a los usuarios finales, también con ello se podría aportar con una mayor ayuda y personalización de mensajes en pantalla de una mejor manera. Esto también equivale a seguir siendo parte de la *Web 2.0* y crecer al mismo tiempo que ésta lo hace.
- En las futuras versiones se recomienda crear un archivo de

configuración preferiblemente en *XML* que sirva como parametrización para crear los elementos que formaran parte del formulario *HTML*, así como las validaciones adicionales para cada tipo de campo que no ha sido considerado para el desarrollo de la primera versión.

# 9 Bibliografía

**A**LEXANDER, Christopher; **I**SHIKAWA, Sara; **S**ILVERSTEIN, Murray; **J**ACOBSON, Max; **F**IKSDAHL-KING, Ingrid; **A**NGEL, Shlomo, A pattern language: towns, buildings, constructions, 1977.

**C**RANFORD Jason; Peachpit Press, CSS, DHTML & Ajax; cuarta edición, 2007.

**B**ECK, Kent, Una explicación de la Programación extrema: aceptar el cambio, 2002. Addison-Wesley Iberoamericana España, S.A

**N**EWKIRK, James; **M**ARTIN, La Programación Extrema en la práctica, 2002. Addison-Wesley Iberoamericana España, S.A.

**A**CID:
http://databases.about.com/od/specificproducts/a/acid.htm
**A**pache:
http://www.maestrosdelWeb.com/editorial/sindicando/
**C**akePHP:
http://cakephp.org/
**C**RC:
http://lsi.ugr.es/~ig1/isoo/crc.pdf
**I**IS:
http://www.microsoft.com/windowsserver2003/iis/default.mspx
**I**NFORMATION SCHEMA:
http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=34917
**I**nnoDB
http://dev.mysql.com/doc/refman/5.1/en/innodb-overview.html
**M**etodología XP:

http://www.monografias.com/trabajos-pdf/prototipos-rad-programacion/prototipos-rad-programacion.pdf

http://www.xprogramming.com/xpmag/whatisxp.htm

http://www.xprogramming.com

**M**VC:

http://www.proactiva-calidad.com/java/patrones/mvc.html

**M**ediaLive International:

http://www.medialiveinternational.com/

**M**ySQL

http://dev.mysql.com/doc/refman/5.0/es/features.html

**O**'Reilly Media, Inc.:

http://oreilly.com/

**O**pen Source

http://www.eweek.com/article2/0,1895,1983364,00.asp

http://www.eweek.com/c/a/Application-Development/eWEEK-Labs-Bakeoff-Open-Source-Versus-Net-Stacks/

**P**HP:

http://talks.php.net/show/fossin_Web2/1

http://www.php.net/usage.php

http://www.php.net/manual/en/intro.mcrypt.php

http://www.Webexperto.com/articulos/art/217/historia-de-php/http://www.scribd.com/doc/247520/PHP-Leads-Web2-0

**P**HP License:

http://www.php.net/license/

**P**HPDoc:

http://www.phpdoc.org/

**P**atrones de diseño

http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/patrones.html

**S**indicación:

http://www.maestrosdelWeb.com/editorial/sindicando/

**S**ingleton:

http://upload.wikimedia.org/wikipedia/commons/f/fb/Singleton_UML_class_diagram.svg

http://blogs.oshyn.com/jtapia/entry/5_quick_approches_to_singleton

**S**olar PHP:

http://www.solarphp.com/

**S**QL:

http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.admin.doc/doc/c0004100.htm

**S**ymfony:

http://www.symfony-project.org/

**W**eb 2.0:

http://internality.com/Web20/

http://www.pangea.org/peremarques/Web20.htm

http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-Web-20.html

**W**3C:

http://www.w3c.org/

**X**HTML:

http://www.maestrosdelWeb.com/editorial/haciaxhtml/

**X**P

https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Metodolog%C3%ADas%20%C3%81giles%20-%20Extreme%20Programming.ppt

http://www.xprogramming.com/xpmag/whatisxp.htmhttp://es.tldp.org/Presentaciones/200211hispalinux/ferrer/robles-ferrer-ponencia-hispalinux-2002.html#id2754777s

http://www.asturlinux.org/archivos/jornadas2006/ponencias/ProgExtrema_Berrueta/ponencia-sl-y-xp.pdf

http://www.chuidiang.com/ood/metodologia/extrema.php

http://eisc.univalle.edu.co/materias/WWW/material/lecturas/xp.pdf

**Z**end:

http://www.zend.com

**Z**end Framework:

http://www.zendframework.com

http://zendframework.com/manual/en/coding-standard.html

http://framework.zend.com/docs/quickstart

# Anexos