Universidad de las Ciencias Informáticas

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Framework de Mapeo Objeto-Relacional para PHP

Autores: Lisandra Díaz Romero

Yosbel Marín Sardiñas

Tutor: Ana Marys Serafín Linares

Ciudad de la Habana

Junio, 2010

Declaración de autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) a que haga el uso que estimen pertinente con el mismo. Para que así conste firmo la presente a los ____ días del mes de junio del 2010.

Autores:

Yosbel Marín Sardiñas

_____

Lisandra Díaz Romero

_____

Tutor:

Ing. Ana Marys Serafín Linares

_____

Dedicatoria

A mis padres y a mi hermana querida, con toda la fuerza de mi corazón va dedicado todo mi esfuerzo.

Gracias por existir…

Lisandra

A mis padres y hermanos

Yosbel

Agradecimientos

A Dios por guiar mi vida por el buen camino.

A Gladys, mi madre, lo es todo, cualquier adjetivo es insuficiente para describirla.

A Fernanado, padre de crianza, por ponerme en las manos, cuando tenía 9 años, un libro de Turbo Pascal y un manual de MSDOS.

Lorenza, mi abuela, dice que soy su futuro, y me prepara como tal. La amo.

Leonel, padre biológico, a pesar de la distancia no abandona el rol de padre.

Maylin, por ser tan atenta y preocupada.

A Todos mis hermanos.

A mi tía Mayda, que aunque está lejos no se olvidó de mí y me escribió, para preguntarme que me hacia falta y darme su apoyo.

A mi tía Anaivis, que me regaló la memoria flash que ha jugado un papel protagónico en este trabajo.

A mis amigos y compañeros y a la comunidad Gamer, somos una familia.

A los que de una manera u otra pusieron su grano de arena para que esta tesis fuera realidad.

Estos son:

Los profesores: Ana Marys, Clarissa, Yareisys, Febe

Y estudiantes: Miguel y Jorge Fernando, por prestarme su PC. Fredy, Michael.

Estos también me animaron a seguir adelante, en momentos claves en que todo parecía fallar. Siempre conté con sus opiniones positiva desde el principio. Su apoyo es lo que me ha animado a seguir.

También les agradezco a todos aquellos a los que tuvieron que aguantar mis pesadeces, compañeros de cuarto, tanto hembras como varones.

Y en especial a mi novia Lisandra, por todo el amor que me ha dado, por los momentos malos y buenos que hemos pasado, en fin todos los sinónimos de cariño y amor que puedan existir, ella me ha mostrado cuáles son.

Yosbel

Agradecimientos

A las tres personas más importantes de mi vida, sin las cuales no podría estar aquí siendo la persona más feliz del universo, sin ustedes mi vida no tendría sentido, porque ustedes forman la luz que ilumina mi camino. A mi madre y a mi padre por estar presentes cuando los necesito en los buenos y malos momentos, por saber perdonar mis errores y ayudarme a levantar. Nunca será suficiente para agradecerles todo lo que han hecho por mí, enseñarme a vivir ha sido su enseñanza eterna, pero más grande ha sido darme fuerzas para vivir.

A mi hermana querida, gracias por todo el amor que me das, por estar siempre a mi lado dando lo mejor de ti y por saber soportarme.

A mi pequeña hermana Yuly, que aunque esta muy lejos siempre me dio muy buenos consejos, me apoyó sin límites y dio fuerzas suficientes para terminar y poder ser una profesional, gracias por todo el amor que me has dado.

A mis abuelos Luz Marina, China y Reinaldo, gracias por cuidar de mí, brindarme su ayuda incondicional y ser los abuelos más lindos del mundo, los adoro…

A todos mis tíos en especial a Rosa, Romilio, Cucho y aquellos que ya no están con nosotros pero que siempre los llevo en el corazón.

A Gladys, Lucia, abue las quiero como a una madre, a Darmalinita, Dachelis, Taimir e Ismael, gracias por haberme ayudado a sentir feliz todo el tiempo que duró mi carrera y estar siempre presentes .

A mis amistades, Naivis, Jehysi, Marisleidis y a todos aquellos que aunque ya no se encuentran a mi lado de una forma u otra han marcado la diferencia, contribuyendo a mi desarrollo como profesional y sobre todo como ser humano.

A mi compañero Yosbel, que por ser el ultimo no es menos importante, pero si el mas especial, gracias por darme todo tu apoyo, amor y comprensión, Kiss.

Lisandra

Resumen

Los framework de mapeo objeto-relacional surgieron para mejorar el intercambio de información entre una aplicación orientada a objetos y una base de datos relacional. Algunas de sus características presentan desventajas que frenan el desarrollo exitoso del proceso de intercambio de información. El presente trabajo consiste en desarrollar un framework de mapeo objeto-relacional libre y multiplataforma, el mismo tiene como propósito mejorar su usabilidad y aprendizaje, facilitando al programador su proceso de desarrollo. El documento recoge los resultados del trabajo realizado para desarrollar el framework, identificando en un primer momento la situación problemática y efectuando un estudio del arte. Luego se realiza un análisis de las distintas tendencias y tecnologías actuales que pueden ser utilizadas para el modelamiento y desarrollo del framework. Se identifican las funcionalidades que debe brindar el sistema y se analiza para determinar la forma en que se va a desarrollar su diseño e implementación realizando la modelación de este proceso.

Palabras claves:
Base de datos relacional
Framework
Mapeo objeto-relacional
PHP

Índice

INTRODUCCIÓN---------- 1
CAPÍTULO 1 ACTUALIDAD DE LAS HERRAMIENTAS DE MAPEO OBJETO-RELACIONAL---------- 5
1.1 CONCEPTOS ASOCIADOS AL DOMINIO DEL PROBLEMA 5
1.2 ACTUALIDAD DEL DOMINIO DEL PROBLEMA 7
1.3 ANÁLISIS DE SOLUCIONES EXISTENTES 10
1.4 TENDENCIAS ACTUALES 12
1.4.1 Patrones 13
1.4.1.1 Patrones para el mapeo a base de datos relacionales---------- 13
1.4.1.1.2 Patrón de mapeo a base de datos relacional utilizado ---------- 14
1.4.1.2 Patrones de diseño ---------- 14
1.4.1.2.1 Patrones de diseño utilizados---------- 15
1.5 ENTORNO DE DESARROLLO INTEGRADO 16
1.6 LENGUAJE DE IMPLEMENTACIÓN 17
1.7 METODOLOGÍAS DE DESARROLLO 17
1.7.1 Proceso unificado de desarrollo de software 17
1.7.2 Programación Extrema 18
1.8 LENGUAJE DE MODELADO 19
1.9 HERRAMIENTA CASE 20
1.9.1 Visual Paradigm for UML 20
1.10 ESTÁNDAR SQL 20
1.10.1 ISO/IEC 9075:1992 20
1.11 FRAMEWORK DE PRUEBAS 21
1.11.1 Lime 21
1.12 CONCLUSIONES 22
CAPÍTULO 2 PRESENTACIÓN DE LA SOLUCIÓN PROPUESTA 23
2.1 DESCRIPCIÓN DEL ENTORNO DE DESARROLLO 23
2.1.1 Características del framework ORM a desarrollar 23
2.1.2 Personas relacionadas con el sistema 24
2.2 REQUERIMIENTOS 24
2.2.1 Requerimientos funcionales que debe cumplir el sistema 24
2.2.2 Requerimientos no funcionales que debe cumplir el sistema 26
2.3 PLANIFICACIÓN 29
2.3.1 Historias del Usuario 29
2.3.2 Iteraciones 32
2.3.2.1 Plan de Iteraciones ---------- 32

Índice

2.3.3 Plan de entrega 33
2.3.4 Conclusiones 34
CAPÍTULO 3 DISEÑO E IMPLEMENTACIÓN 35
3.1 DISEÑO DEL SISTEMA 35
3.1.1 Estrategias de Programación 35
3.1.2 Patrones de diseño utilizados 35
3.1.3 Convenciones 37
3.1.4 Paquetes y organización 38
3.1.5 Diagrama de clases del sistema 40
3.1.6 ¿Qué es un componente? 41
3.1.6.2 Diagrama de componentes ---------- 42
3.2 IMPLEMENTACIÓN 45
3.2.1 Iteración 1 45
3.2.2 Iteración 2 48
3.2.3 Iteración 3 50
3.2.4 Iteración 4 52
3.3 CONCLUSIONES 54
CAPÍTULO 4 PRUEBAS 55
4.1 PRUEBAS 55
4.1.1 Pruebas de Aceptación 55
4.2 TEST DE USABILIDAD 60
4.3 CONCLUSIONES 61
CONCLUSIONES ---------- 62
RECOMENDACIONES ---------- 63
REFERENCIA BIBLIOGRÁFICA ---------- 64
BIBLIOGRAFÍA ---------- 67
ANEXO 1 ---------- 71
ANEXO 2 ---------- 72

INTRODUCCIÓN

Con la aparición del software libre han surgido nuevas metas para Cuba, este garantiza la soberanía tecnológica, fundamentalmente la de las naciones subdesarrolladas, que a diferencia del programa propietario profundiza, fortalece la transculturación globalizante y conlleva a la pérdida de autonomía. La migración hacia el software libre constituye una necesidad táctica y estratégica para el país, proporcionando una seguridad en términos del soporte informático. Las aplicaciones informáticas libres brindan todas las posibilidades para llegar a convertirse en una fortaleza mundial en esta esfera y eliminar los vínculos comerciales con las empresas privadas de la industria del software.

La Universidad de las Ciencias Informáticas (UCI) es uno de los principales pilares en la informatización de la sociedad cubana. Fue fundada con el principal objetivo de formar un capital humano especializado en ciencias informáticas, producción de software y servicios informáticos. Convirtiéndose con esta responsabilidad en cómplice del logro de la emancipación de la tecnología cubana.

Cuba es un país que pretende ser una potencia de software, para alcanzar esta meta es necesario lograr una verdadera independencia tecnológica. Se hace imperante construir herramientas propias de trabajo. Las que se disponen en la actualidad son en su gran mayoría extranjeras, por lo que pueden en cualquier momento cambiar sus términos de licencia o dejar de recibir soporte por sus desarrolladores.

Los frameworks de mapeo objeto-relacional (ORM) adquieren protagonismo en situaciones que se hace necesario agilizar el proceso de desarrollo de aplicaciones Web. Los existentes poseen características que dificultan su aprendizaje y sacrifican usabilidad. Es común notar en estos un alto grado de configurabilidad, lo que trae consigo la imposibilidad de dominarlos totalmente si no se cuenta con experiencia en el trabajo con ellos.

Para desarrollar las tareas básicas y la interacción con la mayoría de estos frameworks ORM, se ofrece al usuario una interfaz de línea de comandos, haciéndolo enfrentarse a una consola, lo cual hace al usuario elegir entre ser dependiente de la documentación o memorizar una serie de comandos e instrucciones, no cumpliendo así con el objetivo final de su utilización, que es agilizar el proceso de creación de una aplicación Web, obstaculizando a los programadores inexpertos en el trabajo con estas herramientas.

Para realizar consultas a la base de datos implementan diferentes estrategias, con la máxima tendencia a evitar las sentencias SQL (Lenguaje de Consultas Estructurado). Algunos hacen uso de Objetos Criterio (Criterion Objects), no son más que objetos configurables por criterios de selección, para sustituir las instrucciones SQL, cada criterio de selección SQL es tratado como un objeto, lo cual hace las consultas en extremo verbosas, y más aún si se necesita hacer una consulta compleja; otros usan un lenguaje propio conocido como OQL (Leguaje de Consultas Objetos), que propone notables mejoras en cuanto a sencillez y legibilidad del lenguaje, pero la implementación de los ORMs actuales no han logrado aún deshacerse totalmente del SQL, y la facilidad de uso de sus interfaces de consultas es mejorable.

Las bibliotecas extensas, por su gran tamaño tienen la particularidad de ocupar gran cantidad de memoria, tanto física como RAM (Memoria de Acceso Aleatorio), lo que provoca que al conectarse una gran cantidad de usuarios a una aplicación concurrente construida sobre un framework con estas características, el punto de desbordamiento de la RAM sea menor, el servidor debe recurrir a la Memoria Virtual (porción física de memoria asignada para auxiliar la RAM) con más urgencia, disminuyendo progresivamente el desempeño de la aplicación.

Los frameworks ORM brindan la posibilidad de omitir detalles durante el proceso de desarrollo de una aplicación, los cuales se encarga de resolver a través de estrategias predefinidas por dicha herramienta. A veces no son lo suficientemente específicas como para ajustarse a necesidades tan definidas y en el intento de ahorrar trabajo llevan al desarrollador por un camino indeseado. Muchos de los existentes requieren para su funcionamiento hacer alteraciones en la base de datos, agregando columnas y tablas (comúnmente llamada información oculta), que necesitan para su labor. Si se construye una base de datos utilizando algunos de estos ORMs que necesitan información oculta, sólo se logrará su uso óptimo a través del framework ORM utilizado para crearla, pues la información oculta para este, es información visible para otras herramientas.

Esta **situación problemática** ocasiona que en el momento del despliegue del software, después de un gasto considerable de tiempo en el desarrollo de la aplicación, se vea un atraso a causa del planteamiento de nuevas estrategias, a partir de la incompatibilidad de los recursos requeridos por el software y de los disponibles por el hardware. La opción de reconstruir o mejorar el software, no es compatible con el objetivo principal de un proyecto productivo de este tipo que es tener una aplicación lista lo antes posible que cumpla con los requerimientos necesarios para ser usada por el cliente.

De la situación anterior se identifica el siguiente **problema a resolver**: ¿Cómo agilizar el proceso de desarrollo de aplicaciones Web con frameworks de mapeo objeto-relacional enfocados en su usabilidad?

Definiendo como **objeto de estudio** el proceso de desarrollo de los frameworks de mapeo objeto-relacional y como **campo de acción** el proceso de desarrollo de los frameworks de mapeo objeto-relacional para aplicaciones Web en PHP.

Se plantea como **objetivo general** de esta investigación: desarrollar un framework de mapeo objeto-relacional para PHP.

Para cumplir con este objetivo se proponen las siguientes **tareas investigativa**:

1. Describir el estado del arte.
2. Evaluar soluciones y herramientas encontradas.
3. Seleccionar la Metodología de Desarrollo de Software.
4. Identificar las funcionalidades que debe brindar el sistema.
5. Diseñar el sistema a implementar.
6. Implementar el sistema.
7. Realizar pruebas de aceptación.
8. Documentar el sistema implementado.
9. Elaborar un manual de usuario para la utilización de la herramienta implementada.

Para guiar el desarrollo de este proceso se plantea la siguiente **idea a defender**:
Si se crea un framework de mapeo objeto-relacional fácil de aprender y dominar, posibilitará un incremento de la productividad y ahorro considerable del tiempo de trabajo.

Como **posible resultado** se espera obtener un framework de mapeo objeto-relacional con requerimientos mínimos de configuración e interfaz gráfica para realizar las tareas básicas.

Para tributar al cumplimiento del objetivo general de la presente investigación se propone el uso de los siguientes métodos científicos.

Métodos teóricos

Analítico-sintético: para analizar la situación problemática, determinar posibles variantes de solución al consultar las bibliografías y extraer los elementos más importantes relacionados con el objeto de estudio.

Inductivo-deductivo: partiendo de herramientas y hechos concretos, se determinan las características comunes, que llevan hacia la formulación de la idea a defender, deduciendo de esta los posibles resultados a obtener con el desarrollo de la investigación.

Modelación: utilizado para modelar cierta abstracción de la realidad con el objetivo de documentar el producto de la investigación y para ofrecer argumentos explicativos de manera que sea comprensible.

Métodos empíricos

Observación: para analizar la naturaleza del problema con el fin de extraer mediante la observación las principales características y realizar la formulación de la situación problemática. También es utilizado en la fase de prueba para la comprobación del cumplimiento de la idea a defender.

Entrevista: La opinión de expertos y usuarios de los frameworks ORM, conducen a obtener información específica de los problemas que estos presentan. La información relacionada con el tema durante la investigación permite acumular una serie de problemáticas que posteriormente serán objetos de estudio para descubrir soluciones.

Capítulo 1 Actualidad de las herramientas de Mapeo Objeto-Relacional

En el siguiente capítulo se llevará a cabo la fundamentación teórica. Se expondrá el estado del arte analizando las soluciones existentes. Después se abordarán las tendencias y tecnologías utilizadas y se especificará la metodología de desarrollo a utilizar.

1.1 Conceptos asociados al dominio del problema

SQL (Lenguaje de Consultas Estructurado, en inglés Structured Query Lenguaje): es un estándar en el lenguaje de acceso a bases de datos. Originalmente, era un lenguaje de acceso al sistema de gestión de bases de datos denominado DB2 en plataformas 390 de IBM. En la actualidad está adoptado por ISO. (MasterMagazine, 2004)

Lenguaje estandarizado por el Instituto Nacional Estadounidense de Estándares (ANSI, por sus siglas en inglés, American National Standards Institute) en 1986 y ratificado por ISO en 1987 como el lenguaje de consulta de todos los manejadores de base de datos.

ORM (Mapeo Objeto-Relacional, en inglés Object-Relational Mapping): son diseñados para integrar lenguajes de programación orientados a objetos capaces de manejar base de datos relacionales. (K. Barry Douglas, 2010)

Es una técnica de programación para manejar datos entre sistemas que utilizan lenguaje de programación orientado a objetos y base de datos relacionales. Básicamente es manejar una base de datos relacional a través de objetos nativos en un lenguaje específico.

Base de Datos (BD): es una colección de archivos interrelacionados, son creados con un sistema gestor de base de datos. El contenido de una base de datos engloba a la información concerniente (almacenadas en archivos) de una organización, de tal manera que los datos estén disponibles para los usuarios, una finalidad de la base de datos es eliminar la redundancia o al menos minimizarla. (Departamento de Sistemas y Computación del Instituto Tecnológico de La Paz)
Es un conjunto de datos que pertenecen a un mismo contexto, almacenados con carácter transitorio o permanente.

POO (Programación Orientada a Objetos): es un paradigma de programación que se basa en objetos y sus interacciones, que se llevan a cabo a través del envío de mensajes, para invocar sus funcionalidades que se encuentran encapsuladas dentro de los mismos objetos. (Katrib Miguel)

Código Abierto (en inglés Open Source): el término 'open source' fue acuñado por Christine Peterson, y se registró para actuar como marca registrada para los productos de software libre. La intención de su definición, es establecer que esos criterios contengan la esencia de lo que quieren los programadores que signifique: que aseguren que los programas distribuidos con 'licencia open source' estarán disponibles para su continua revisión y mejora para que alcancen niveles de fiabilidad que no pueda conseguir ningún programa comercial 'cerrado'. (Rojo Iñaki I., 1999)
Es el término con el que se conoce al software distribuido y desarrollado libremente, se enfoca mayormente en los beneficios prácticos de compartir el código.

Framework: término usado en programación orientada a objetos para definir un conjunto de clases que definen un diseño abstracto para solucionar un conjunto de problemas relacionados. (Universidad de Alicante, 2009)
Marco de trabajo, del inglés "frame" que significa marco y "work" que significa trabajo. Son herramientas implementadas tanto con objetivos específicos como generales que reúnen un conjunto de características y funcionalidades para facilitar la implementación de las aplicaciones a las cuales va dirigida. Es un patrón o esqueleto de una aplicación con un conjunto de estándares y organización, el cual permite la construcción de una aplicación organizada.

Carga Perezosa (en inglés Lazy Load): patrón utilizado en el campo de modelos conceptuales, permite retardar la carga de un determinado objeto o colección de objetos (carga que suele ser costosa) justamente hasta el momento en el que el objeto o colección de objetos son necesarios. Este patrón contribuye a mejorar la eficiencia en las operaciones de programación habituales. (Zorrilla Unai, Hernández Octavio, Quintás Eduardo, 2008)
Es un patrón de diseño que consiste en no cargar o demandar un recurso u objeto hasta el punto de ser necesitado.

Carga Ansiosa (en inglés Eager Load): es todo lo contrario de Lazy Load, consiste en cargar recursos u objetos antes de ser solicitados.

Lenguaje Unificado de Modelado (**UML,** en inglés Unified Modeling Language): es un lenguaje de modelado que ayuda a especificar, visualizar y documentar, modelos de sistema de software, incluyendo su estructura y diseño, de manera que cumpla con todos sus requisitos. Es utilizable para modelar negocios y también sistemas que no sean específicamente de software. (Object Management Group, 2010)

1.2 Actualidad del dominio del problema

Las bases de datos relacionales han tenido un mayor éxito que las bases de datos orientadas a objetos, son las mayormente usadas y más populares para todo tipo de aplicaciones que requieran de un sistema de base de datos. La POO es utilizada para los proyectos de gran escala, por las oportunidades que ofrece principalmente la de una ventajosa reusabilidad.

Entre el modelo de datos orientado a objetos (el utilizado en la capa de aplicaciones) y el relacional (utilizado en la capa de almacenamiento de los datos) existe una incoherencia llamada Diferencia por Impedancia Objeto-Relacional, en inglés O/R Impedance Mismatch. Esto proviene de los orígenes de estos dos paradigmas ya que el paradigma orientado a objetos está basado en los principios de la ingeniería, mientras que el paradigma relacional nace sobre principios matemáticos.

Este sencillo ejemplo que ilustra una de las deficiencias, pretende hacer entendible la diferencia por impedancia:

Se tienen las siguientes entidades de negocio.

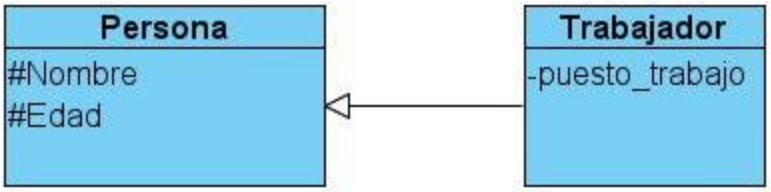

Figura 1 Diagrama de clases Persona-Trabajador

Una instancia de la clase **Trabajador** quedaría.

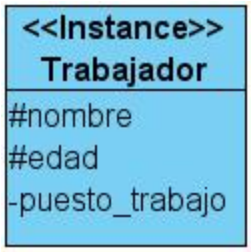

Figura 2 Instancia de la clase Trabajador

Es decir desde un objeto de tipo **Trabajador** se accede a su propiedad **puesto_trabajo**, también a las heredadas de la clase **Persona** (**nombre** y **edad**).

A continuación el diagrama entidad-relación correspondiente a este modelo.

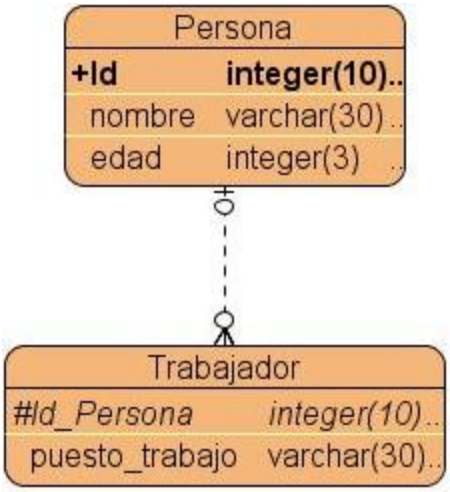

Figura 3 Diagrama entidad relación Persona-Trabajador

A simple vista se ve la imposibilidad de representar herencia en un modelo relacional, para poder seleccionar un trabajador completo se realizaría a través de una sentencia SQL parecida a lo que se muestra a continuación:

SELECT * **FROM** *Persona*, *Trabajador*
WHERE *Trabajador.Id_Persona = Persona.Id* **AND** *otros criterios de selección*

La entidad **Trabajador** en el modelo relacional ofrece solamente el atributo **puesto_trabajo** y una referencia a la entidad relacionada **Persona**. Otro problema es, si se restringe el acceso a la clase **Persona** declarándola como abstracta, solamente se pueden crear objetos de tipo **Trabajador**, lo que quiere decir que las únicas instancias que existen de la clase **Persona** son las propias instancias de la clase **Trabajador**, en cambio en el modelo relacional no se tienen opciones para esto, es decir se puede tener un registro de la tabla **Persona** que no esté relacionado con ningún registro de la tabla **Trabajador**. En conclusión no es posible representar objetos y sus relaciones, tales y como son sobre un modelo de objetos relacional.
La técnica ORM apareció para solucionar esta diferencia y facilitar el manejo de datos relacionales desde la capa de aplicación usando POO. Otro objetivo es eliminar las sentencias SQL y proveer homogeneidad

al código de la capa de aplicaciones.

En la actualidad existen varios frameworks ORM para el lenguaje PHP. Los más potentes presentan un alto grado de configuración, lo que parece ofrecer un amplio espectro de personalización, se convierte en tiempo perdido de desarrollo, mientras más configurable más necesidad de capacitación y experiencia requieren los usuarios de la biblioteca en uso. Algunos poseen una inclinada curva de aprendizaje, lo que hace a los desarrolladores más dependientes de la documentación.

1.3 Análisis de soluciones existentes

Doctrine

Es un potente ORM libre para PHP 5.2.3 y versiones mayores, desarrollado inicialmente por Konsta Vesterinen. Una de sus características principales es la de opcionalmente escribir consultas a la base de datos relacional al estilo POO dialecto-SQL llamado DQL (Lenguaje de consultas Doctrine, en español) inspirado por Hibernate. Una de las principales inconformidades de sus usuarios es el proceso de liberar la memoria manualmente porque los objetos relacionados devueltos como resultado de una consulta poseen referencias circulares, función que el recolector de basuras del actual PHP no soporta, además posee métodos mágicos lo que aporta limitaciones en el completamiento de código. Según François Zaninotto el autor principal de “The Definitive Guide to Symfony” es relativamente más lento que Propel.

Propel

Es una biblioteca ORM de código abierto escrita en PHP 5. Permite el acceso a base de datos relacionales usando objetos, proveyendo una simple interface de funciones para almacenar y recuperar datos. Propel permite a los desarrolladores trabajar con las base de datos relacionales de la misma forma que se trabaja con clases y objetos en PHP. Según Fabien Potencier el creador del framework Symfony, entre los ORM que usa es el más verboso y no es lo suficientemente eficiente en el manejo de operaciones complejas.
Es de los más antiguos y mejor documentados, además de incluir validaciones. (Leech, 2007)

ORM de CakePHP

CakePHP es un framework de desarrollo en el lenguaje PHP que tiene incorporado su propio ORM. Su principal éxito se encuentra en que es fácil de aprender, acompañado de las prestaciones del framework. Sus principales críticas han sido por el uso de arreglos en vez de verdaderos objetos y la ausencia de ”Lazy Load”, causando la pérdida de usuarios importantes, por ejemplo, el sitio Mozilla Addons lo desechó por la sobrecarga de memoria que generalmente devuelve enormes arreglos anidados. (XMundo, 2009)

Rocks

Es una biblioteca de código abierto que mapea base de datos relacionales similar al patrón Active Record facilitando el acceso y el manejo de datos. El mapeo es dinámico lo que le resta un poco en velocidad. La biblioteca no requiere configuración, todo lo que necesita saber es si los parámetros de conexión son válidos y ella por si sola se configura.

Qcodo

Es un framework de código abierto escrito en PHP 5 que implementa un ORM con las funciones básicas CRUD (acrónimo de Crear Leer Actualizar Borrar, en inglés Create Read Update Delete) con interfaz de usuario y vínculos usando la tecnología AJAX (JavaScript Asíncrono y XML, en inglés Asynchronous JavaScript And XML) para modelo de datos existentes. En adición incluye una fuerte integración con herramientas HTML (Lenguaje de Marcado de Hipertexto, en inglés HyperText Markup Language) y JavaScript para interactuar directamente con las entidades generadas. Además es una herramienta compacta por lo que posibilita su portabilidad.

ADOdb_Active_Record

Las tablas y tuplas de la base de datos es abstraída a objetos nativos de PHP. Esto permite que los programadores se enfoquen más en la manipulación de los datos y menos en la escritura de consultas SQL. Esta herramienta permite conectarse a múltiples base de datos. ADOdb_Active_Record está diseñado bajo los principios del patrón de diseño Active Record e intenta representar lo más fielmente posible las tablas de la base de datos usando objetos PHP. Para este ORM cada instancia de una clase

mapa representa un tupla en la tabla. Las relaciones entre tablas también pueden ser definidas permitiendo que los objetos de ADOdb_Active_Record se aniden, manejando así las relaciones en la base de datos como relaciones entre objetos. (Lim John, 2009)

EZPDO

Es una biblioteca muy completa, su peculiaridad se esconde en el estilo de configuración e instrucciones a través de líneas de comentarios. Requiere para su uso un mínimo de conocimientos SQL, tiene un pequeño núcleo para garantizar la rapidez y maneja relaciones uno a mucho y mucho a mucho automáticamente. Implementa un propio sistema de consultas llamado EZOQL y genera las tablas de la base de datos automáticamente. (ezpdo4php, 2007)

1.4 Tendencias actuales

En la actualidad hay una gran demanda de aplicaciones empresariales. Por su gran tamaño y el manejo requerido de gran cantidad de información se hace necesario el uso de sistema de base de datos. Los desarrolladores actuales usan tecnología Web por sus prestaciones, facilidades sobre la arquitectura cliente-servidor y por su orientación a documentos.

Los frameworks de abstracción de datos al igual que los ORM han llegado a la cúspide, debido a las facilidades que ofrecen para el tiempo de desarrollo. Existe mucha competencia entre las empresas desarrolladoras de software y optimizar el tiempo para la entrega de un producto final se ha convertido en un requerimiento primordial.

Por otra parte la industria de software va camino al software libre que en estos tiempos ha alcanzado la cima. Esto ha permitido que los programadores de todo el mundo compartan el conocimiento y contribuyan a la desmonopolización del software.

En estos tiempos la arquitectura y el diseño de software están en plena maduración, el surgimiento y uso de patrones de diseño se ha hecho muy popular. En cuanto al diseño de las capas de acceso a datos hay

dos patrones relevantes uno es el Data Mapper y el otro es el Active Record, este último prevalece, se acerca más a la POO.

1.4.1 Patrones

Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, entonces describe el núcleo de la solución a ese problema, de manera que puedes usar esa solución un millón de veces, sin hacerlo dos veces de la misma manera (Fowler Martin, 2001).

1.4.1.1 Patrones para el mapeo a base de datos relacionales

El autor Martin Fowler recomienda en su libro “Patterns of Enterprise Application Architecture” cuatro patrones para el mapeo a base de datos relacionales.

Active Record

“Un objeto que envuelve una tupla en una tabla de una base de datos o en una vista, encapsula el acceso a la base de datos y agrega lógica de dominio en ese dato.” Un objeto se encarga de los datos y el comportamiento. Muchos de estos datos son persistentes y necesitan ser almacenados en una base de datos. Active Record usa el más obvio acercamiento, poniendo la lógica de acceso a datos en el dominio del objeto. Así todas las clases conocen como manipular sus datos. (Fowler Martin, 2001).

Data Mapper

“Una capa de mapeo que mueve datos entre objetos y una base de datos mientras se mantienen independientes uno de otro del mapeo en sí mismo.” Data Mapper es una capa de software que separa los objetos cargados en memoria de la base de datos. Su responsabilidad es transferir datos entre la base de datos y los objetos, también es el encargado de aislar uno del otro. Con el Data Mapper los objetos no necesitan conocer la base de datos ni interface de código SQL. (Fowler Martin, 2001).

Table Data Gateway

"Un objeto que actúa como puente a una tabla de la base de datos" .Consiste en crear una clase por cada tabla, cada una de estas clases se encargan de realizar todas las consultas o acciones hechas a la tabla mapeada. Un simple Table Data Gateway encapsula el código SQL para el acceso a una simple tabla (Fowler Martin, 2001).

Row Data Gateway

"Un objeto que actúa como puente a una simple tupla en una tabla de una base de datos". Un Row Data Gateway ofrece objetos que se asemejan exactamente a una tupla en una estructura de datos relacional, pero se puede acceder a este con mecanismos regulares de programación. Todos los detalles de acceso permanecen ocultos detrás de esta interface (Fowler Martin, 2001).

1.4.1.1.2 Patrón de mapeo a base de datos relacional utilizado

El patrón escogido inicialmente fue Active Record, para darle la facilidad al usuario de colocar la lógica de negocio con los mecanismos de persistencia de datos. Pero después de la primera iteración se tomaron en cuenta los riesgos de memoria y rendimiento que acarrea esta estrategia. Fue tomada como segunda y final alternativa el uso de Row Data Gateway. De esta manera el usuario desde la lógica de negocio podrá usar objetos que garantizan la persistencia de sus datos.

1.4.1.2 Patrones de diseño

Representan la solución a un problema de diseño y para que esta sea considerada un patrón, debe haber comprobado su efectividad dando respuesta a los problemas similares dados en ocasiones anteriores; tiene que ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

1.4.1.2.1 Patrones de diseño utilizados

Registro (en inglés Registry)

Es un medio simple y eficiente de compartir datos y objetos en nuestra aplicación sin tener que preocuparse de mantener numerosos parámetros o hacer uso de variables globales. (Armenteras Comellas Màrius, 2007)

Un objeto conocido que otros objetos pueden utilizar para encontrar objetos y servicios comunes. En esencia parece un objeto global, el cual es accedido directamente por otros.

Método de Fabricación (en inglés Factory Method)

Define una interface para crear un objeto, dejando a las subclases decidir el tipo específico. Permite delegar la responsabilidad de instanciación a subclases. (Garzás Javier, 2000)

Centraliza en una clase constructora la creación de objetos de un subtipo, de un tipo determinado, ocultando al usuario la casuística para elegir el subtipo que crear. (Jaramillo Daniel E., 2008)

Solitario o Instancia única (en inglés Singleton)

Asegura una sola instancia de una clase y provee un punto de acceso global al objeto instanciado. (Garzás Javier, 2000)

Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. (Jaramillo Daniel E., 2008)

Mapa de Identidad (en inglés Identity Map)

Evita tener en memoria dos representaciones distintas del mismo objeto en una transacción de negocio; funciona como un caché de objetos de negocio; como corolario, minimiza las lecturas a la base de datos (el ciclo de vida de un Mapa de Identidad es una transacción de negocio). (Paredes Adrián M., 2008)

Mapeo de Llave Foránea (en inglés Foreign Key Mapping)

Es usado para mapear relaciones de uno a muchos; una relación de este tipo, que en memoria consiste en un objeto A con una colección de objetos B dentro, se mapea con una clave foránea del lado de la entidad contenida; la relación se invierte en la base de datos. (Paredes Adrián M., 2008)

Estado (en inglés State)

Permite a un objeto alterar su comportamiento cuando su estado interno cambia. El objeto parecerá haber cambiado de clase. Cuando un objeto cambia de estado puede responder de maneras diferentes a los mismos mensajes. La forma tradicional es poner sentencias condicionales. (Garzás Javier, 2000)

1.5 Entorno de desarrollo integrado (IDE por sus siglas en inglés)

PHP Designer

Es más que un potente, ligero y rápido entorno de desarrollo integrado y editor para PHP, está lleno de funcionalidades para HTML, Hojas de Estilo en Cascada (en inglés Cascade Style Sheets, CSS) y JavaScript, que facilitan el proceso de desarrollo de un software, tanto para principiantes como para profesionales.

- Altamente configurable con soporte para resaltado de sintaxis inteligentes.
- Análisis instantáneo de sintaxis.
- Soporta código orientado a objetos.
- Soporta auto completamiento y plantillas de código.
- Soporta gestión de proyectos, frameworks.

Se utilizará el IDE **PHP Designer** por sus características que ayudan a una programación rápida, es extremadamente ligero y a la misma vez brinda una completa gama de funcionalidades. Entre ellas configurabilidad y especificidad para el lenguaje PHP, HTML y JavaScript.

1.6 Lenguaje de implementación

PHP acrónimo de "PHP (Personal Home Page): Hypertext Preprocessor"

Es un lenguaje multiparadigma, inicialmente fue creado con la idea de un lenguaje del lado del servidor con simples funciones de procesamiento de hipertexto. Fue desarrollándose soportando tanto la programación estructurada como la orientada a objetos, esta última la más idónea para el desarrollo de grandes software por sus características que permiten la reusabilidad y nivel de abstracción. Desde la versión 5 ha aumentado sus grandes prestaciones para el trabajo con objetos, además de su estabilidad y compatibilidad con los principales sistemas operativos.

Es un lenguaje "Open Source" interpretado de alto nivel, se encuentra en la preferencia de muchos programadores y empresas. Incluso en los estudios estadísticos hecho por Langpop (empresa que ofrece estadísticas en distintos ámbitos sobre la popularidad de los lenguajes de programación) aparece entre los lenguajes más solicitados entre los principales buscadores y en las ofertas de trabajo, por lo que desarrollar para PHP no es tiempo perdido.

1.7 Metodologías de desarrollo

1.7.1 Proceso unificado de desarrollo de software (RUP por sus siglas en inglés Rational Unified Process)

Es una metodología muy robusta está entre las metodologías llamadas pesadas por su rigurosidad y la construcción de una serie de artefactos y documentación detallada muy necesaria para el desarrollo del proyecto.

- **Iterativo e Incremental**: se hacen varias iteraciones en las cuales se produce por cada una, una nueva versión del software.
- **Centrado en la arquitectura**: el desarrollo de todo el sistema gira en torno a la arquitectura del

mismo.

- **Guiado por casos de uso**: los casos de uso describen el comportamiento del sistema. También a partir de estos se realizan los casos de pruebas para comprobar las funcionalidades que el cliente espera del producto.

1.7.2 Programación Extrema (XP por sus siglas en inglés: eXtreme Programming)

Fue creada en el año 1999 su inicio lo dio la publicación del libro Extreme Programming Explained: Embrace Change por Kent Beck. Es una metodología que sigue los principios del manifiesto ágil, a continuación se mencionan.

- La principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de software de valor.
- Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.
- Entregar con frecuencia software que funcione, en períodos de un par de semanas hasta un par de meses, con preferencia en los períodos breves.
- Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.
- Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea.
- La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara.
- El software que funciona es la principal medida del progreso.
- Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica enaltece la agilidad.
- Las mejores arquitecturas, requisitos y diseños emergen de equipos que se auto-organizan.

- En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia.

Es un proceso de desarrollo iterativo e incremental, hace uso de constante pruebas unitarias incluyendo pruebas de regresión, integra al equipo de trabajo con el cliente. Además de la refactorización del código para garantizar su manutención y legibilidad, así como la supresión de todos los errores para la realización de entregas frecuentes.

Selección de la metodología de desarrollo

Se escogió **XP** para el desarrollo del producto por sus características, sus propiedades flexibles que aumentan la productividad y la resistencia al cambio de requerimientos. Adaptándolo a las necesidades reales del desarrollo en proceso se toma como el rol de cliente al mismo equipo de desarrollo ya que los usuarios finales de la herramienta son los propios desarrolladores de aplicaciones Web escritas en PHP; a las necesidades que dan forma a sus exigencias son incorporadas opiniones de desarrolladores externos al equipo de trabajo. Con el uso de esta metodología se podrán tener continuas versiones del producto y la agregación frecuente de nuevas funcionalidades.

1.8 Lenguaje de Modelado

Para la especificación y documentación del producto el lenguaje de modelado es esencial, constituye un pilar para el mantenimiento del software, su posterior actualización y soporte. El lenguaje de modelado utilizado es UML, posee características suficientes para la modelación y la documentación de un software; incluso va más lejos, aun soportando modelación de negocio.

UML prescribe una notación estándar y semánticas esenciales para el modelado de un sistema orientado a objetos. Previamente, un diseño orientado a objetos podría haber sido modelado con cualquiera de la docena de metodologías populares, causando a los revisores tener que aprender las semánticas y notaciones de la metodología empleada antes que intentar entender el diseño en sí. Ahora con UML,

diseñadores diferentes modelando sistemas diferentes pueden sobradamente entender cada uno los diseños de los otros. (Serafín Ana M., Quesada Alexander, 2007)

1.9 Herramienta CASE

1.9.1 Visual Paradigm for UML

Es una herramienta de modelado muy fácil de usar e intuitiva. Soporta UML 2.1, ingeniería inversa, generación de código, importación y exportación de Lenguaje de Marcado Extensible (en inglés Extensible Markup Language, XML), generador de informes.

Visual Paradigm for UML soporta todos los diagramas UML, generación de código Java, diagramas Lenguaje de Modelado de Sistemas (en inglés Systems Modeling Language, SysML) y entidad relación. Provee características de modelación extensiva para caso de uso incluyendo funciones completas para diagramas de caso de uso, editor de eventos de flujo y generación de diagrama de actividades. También produce documentación del sistema en los formatos HTML, MS Word y Formato de Documento Portable (PDF, en inglés Portable Document Format). Los desarrolladores pueden diseñar documentación del sistema a través de plantillas. El análisis del sistema puede estimar las consecuencias a cambios con diagramas de análisis de impacto, como matrices y diagramas de análisis.

1.10 Estándar SQL

1.10.1 ISO/IEC 9075:1992

Estándar del lenguaje de consultas SQL, definido en 1992. Fue diseñado con el objetivo de obtener un lenguaje de base de datos con integridad, mejoras y como reemplazo para el estándar internacional ISO/IEC 9075:1989. Contiene las características del estándar ISO/IEC 9075:1989, el SQL que conforma ISO/IEC 9075:1989 también conforma a este estándar. Pero este agrega significantes nuevas particularidades y capacidades para especificación.

Los cambios técnicos incluyen mejoras a características existentes en su antecesor del año 89, así como sus definiciones. Cambios significativos incluyen una mejor definición de invocaciones directas del lenguaje SQL, capacidades de diagnostico mejoradas, área de diagnostico y soporte para declaraciones. (Digital Equipment Corporation, Julio 1992)

1.11 Framework de pruebas

1.11.1 Lime

Existen muchos frameworks de pruebas unitarias en PHP, bien conocidos como PhpUnit y SimpleTest. Symfony tiene el suyo propio, llamado Lime. Es basado en la biblioteca de pruebas Test::More escrita en Perl. El resultado de las pruebas es mostrado como especifica TAP (Cualquier Protocolo de Pruebas del inglés Test Anything Protocol), diseñado para una mejor lectura de la salida de pruebas.

- Lime provee soporte para pruebas unitarias. Es más ligero que otros framework de prueba escritos en PHP.
- Lanza los ficheros de pruebas individualmente, para evitar efectos extraños entre cada corrida de pruebas. No todos los frameworks de pruebas garantizan un ambiente limpio para cada prueba.
- Las pruebas de Lime son muy legibles como también los son la salida mostrada. En sistemas compatibles Lime usa salida de color para distinguir información importante.
- Symfony usa Lime, así que se pueden encontrar muchos ejemplos en su código fuente.
- El código de Lime es validado por pruebas unitarias.
- Está escrito en PHP, es rápido y está bien codificado y se encuentra contenido en un solo fichero llamado 'lime.php' sin alguna otra dependencia.

(Fabien Potencier)

1.12 Conclusiones

Después de haber desarrollado el presente capítulo se ha concluido que actualmente en los frameworks ORM implementados para PHP existe una tendencia generalizada a ofrecer funcionalidades extras (quizás por objetivos comerciales) a la responsabilidad de *capa de acceso a datos* que es el objetivo principal de estas bibliotecas y en ciertos puntos dificultando su función principal.

La competencia de los ORM para el leguaje PHP es bastante fuerte, dentro de la comunidad de programación Web es el de mayor puntuación después de Java. Hay muchas dificultades en las bibliotecas de acceso a datos actuales que deben ser superadas. Algunas veces por economizar el período de desarrollo se sacrifica la eficiencia del producto final y el tiempo que aparentemente se economiza con anterioridad, es el utilizado posteriormente dándole soporte a la aplicación.

El framework propuesto presenta características que ayudan al trabajo de los programadores, este muestra una interfaz de programación que facilita su uso, así como la compatibilidad con distintos sistemas de base de datos relacionales desde una misma aplicación. El objetivo principal de este capítulo es describir las funcionalidades que debe brindar el sistema a implementar.

2.1 Descripción del entorno de desarrollo

Los primeros requerimientos y los más generales son concebidos a partir de la experiencia del cliente en su trabajo con bibliotecas similares, deficiencias encontradas mediante entrevistas realizadas a desarrolladores con experiencia, e información recopilada de programadores participantes en foros de Internet.

La metodología XP propone el uso de tormenta de ideas y pequeñas reuniones en la fase de inicio, desde donde se proponen los requerimientos preliminares para darle paso a las primeras historias del usuario, que servirán de base para empezar la implementación.

2.1.1 Características del framework ORM a desarrollar

En la introducción del presente documento fueron mencionadas varias características de los frameworks ORM actuales que pueden ser mejoradas. La herramienta en desarrollo propone soluciones para varias de ellas.

Bajo nivel de configuración: el nivel de configuración del framework propuesto, tiende a cero, se examina por todos los medios la posibilidad de minimizar su configurabilidad.

Interfaz Gráfica: para realizar las tareas básicas, la herramienta propone una interfaz gráfica de usuario, instalable. No se necesita de interfaz de línea de comando, cada tarea es realizable a través de la interfaz de administración.

Lenguaje compacto para consultas: Para que los usuarios realicen consultas a la base de datos relacional, se propone un dialecto orientado a objetos y compacto.

Tamaño de la biblioteca de clases: Pasan por alto funcionalidades adicionales, que aunque puedan tributar al desarrollo de una aplicación determinada, no están definidas dentro de una capa de acceso a datos. Se limita sólo a las funciones necesarias, para contribuir a la simplificación del código.

Independencia: El framework propuesto está diseñado teniendo en cuenta la separación entre las funcionalidades soportadas y la estructura de la base de datos relacional con la cual trabaja el usuario. No se requiere de tablas, información oculta en la base de datos relacional o alteraciones a la misma.

2.1.2 Personas relacionadas con el sistema

Las personas relacionadas con el sistema son aquellas que se benefician del mismo. Como el producto propuesto es un framework ORM funcional, su uso está enmarcado en necesidades concretas relacionadas con el desarrollo de aplicaciones programadas en PHP, que requieren el uso de base de datos relacionales.

Usuario: Todo programador con necesidad de facilitar su proceso de desarrollo de aplicaciones en PHP utilizando información contenida en una base de datos relacional.

2.2 Requerimientos

Los requerimientos son una descripción de las necesidades o deseos bajo los cuales se tienen las primeras ideas de una solución.

2.2.1 Requerimientos funcionales que debe cumplir el sistema

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Estos están encaminados a que es lo que el sistema debe hacer. (Rational Rose)

RF 1: Realizar funciones básicas de gestión.

RF 1.1: Crear objetos contenedores de información.

RF 1.2: Extraer de la base de datos relacional información en forma de objetos.

RF 1.3: Actualizar datos en la base de datos.

RF 1.4: Borrar datos de la base de datos.

RF 2: Mapear la base de datos con la cual desea trabajar.

RF 2.1: Especificar parámetros de conexión de una base de datos relacional existente.

RF 2.2: Conectarse a la base de datos relacional.

RF 2.3: Extraer metadatos sobre la base de datos relacional.

RF 2.4: Crear un esquema detallado de la base de datos relacional.

RF 2.5: Generar el código PHP correspondiente a la base de datos relacional a utilizar.

El usuario puede querer migrar a otro sistema de base de datos relacionales diferente al utilizado o simplemente escoger otro para desplegar su aplicación.

RF 3: Brindar una interfaz común para conectarse con diferentes sistemas de base de datos relacionales.

RF 3.1: Permitir cambio de manejador de base de datos relacional.

El usuario puede necesitar trabajar con varias bases de datos relacional a la vez y requerir de varias conexiones diferentes, ya que la información a utilizar puede que se encuentre distribuida.

RF 4: Trabajar con diferentes modelos mapeados en una misma aplicación.

RF 4.1: Manejar varias conexiones a la vez.

RF 4.2: Permitir el cambio de conexión en tiempo de ejecución.

RF 4.3: Cambiar de modelo en tiempo de ejecución.

RF 5: Mapear las relaciones automáticamente.

RF 5.1: Identificar las relaciones existentes entre las tablas de la base de datos relacional.

RF 5.2: Mapear las relaciones existentes.

El usuario requiere que el framework sea compatible con distintos sistemas de base de datos relacionales.

RF 6: Ser extensible a través de manejadores de base de datos relacionales.

RF 6.1: Soportar sistema de reconocimiento de manejadores.

RF 6.2: Implementar clase interfaz que contiene las restricciones y funcionalidades de un manejador de base de datos relacional.

Debido a la gran popularidad y uso del formato JSON (Notación de Objetos Simples de Java) hay una gran probabilidad de uso de este formato en aplicaciones Web. El presente framework está integrado con este formato.

RF 7: Implementar integración con JSON.

RF 7.1: Exportar la información obtenida en forma de objetos a JSON

RF 7.2: Importar información a persistir en formato JSON.

El mapeo objeto-relacional se encarga de evitar el lenguaje SQL en su nivel de abstracción.

RF 8: Consultas a través de instrucciones completamente orientadas a objetos.

RF 8.1: Realizar consultas con una interfaz propia.

RF 8.2: Almacenar información en la base de datos relacional con instrucciones orientadas a objetos.

RF 8.3: Recuperar información de la base de datos relacional con instrucciones orientadas a objetos.

2.2.2 Requerimientos no funcionales que debe cumplir el sistema

Los requerimientos no funcionales son propiedades o cualidades que el sistema debe tener. (Rational Rose). Son las características que hacen al producto atractivo, usable, rápido o confiable.

Apariencia

El sistema provee a los usuarios de una interfaz común para conectarse con diferentes sistemas de base de datos relacional. Presenta un diseño sencillo y elementos visibles que representan cada una de sus acciones, permitiendo realizar las tareas básicas al trabajar con el framework y lograr que los usuarios tengan un buen entendimiento de la aplicación.

Usabilidad

El sistema puede ser utilizado por cualquier persona que tenga conocimientos de programación en PHP. Generalmente para realizar las tareas sobre los framework actuales estos proveen de una interfaz de línea de comando. En el presente producto se ha tenido en cuenta esto como una limitación de las herramientas actuales ya que influyen negativamente en el proceso de aprendizaje y dominio de la herramienta. Provee al usuario de una interfaz gráfica para realizar las tareas básicas para trabajar con el framework, evita la dependencia de interfaz de línea de comandos.
En la solución propuesta se le brinda la comodidad al usuario de evitar en todo momento las sentencias SQL. Para esto las funciones CRUD se realizan completamente orientada a objetos, algo similar al patrón Active Record. Se brinda una interfaz de programación para recuperar información de la base de datos relacional, de fácil uso y aprendizaje.

Rendimiento

La información traída de la base de datos relacional si es reutilizada posteriormente, no tendrá que ser recargada desde la base de datos relacional, para esto es implementado un sistema de caché que se ocupa de guardar temporalmente los objetos y recuperarlos cuando haya una nueva petición.

El lenguaje PHP es un lenguaje interpretado, por lo que es necesario que las funcionalidades se encuentren menos distribuidas, pero a la par es imperante evitar cargar ficheros de código que no vayan a ser utilizados por el usuario, para garantizar la eficiencia de memoria. Los manejadores de base de datos relacional son portadores de funcionalidades de extracción de metadatos de la base de datos relacional,

que son utilizadas en el momento del mapeo, por lo que se han implementado tácticas para la separación de estas, las necesarias para las conexiones y la realización de consultas a la base de datos relacional. El beneficio principal de esta estrategia es minimizar el tamaño de los ficheros cargados y optimizar el uso de memoria del servidor.

Portabilidad

Un 95% de la herramienta no requiere instalación y la del componente que la requiere, es muy sencilla, por lo que toma menos de un minuto para hacerlo funcionar.

Manejadores

MySql (completamente probado)
Postgres (en construcción)
Sqlite (en construcción)

Restricciones

El lenguaje de programación es PHP 5.2.3 ó mayor.
La base de datos no debe tener especificaciones de su estructura en mayúsculas. Es decir, las tablas y sus columnas deben ser nombradas totalmente en letra minúscula.

Correcto:

Tabla: atleta
Columnas: nombre_atleta, sexo, facultad, anno_nacimiento

Incorrecto:

Tabla: Atleta
Columnas: Nombre_aTleta, Sexo, FacultAD, AnNo_Nacimiento

2.3 Planificación

2.3.1 Historias del Usuario

Las historias de usuario (HU) en XP es el artefacto correspondiente a la descripción textual de casos de uso en RUP, a diferencia que en estas se deben escribir no más de tres líneas de descripción. Son escritas preferentemente por los clientes, para resumir sus verdaderas necesidades y no se describe la interfaz de usuario. De una buena redacción y una exacta descripción depende el éxito del producto final.

Proporcionan los detalles sobre la estimación del esfuerzo mediante la asignación de números, indicando el tiempo que se tardarán los programadores en cumplir la actividad. La prioridad en el negocio establecida por el cliente y el nivel de complejidad están dados en alta, media o baja.

Tabla No.1 HU Generar capa de acceso a datos

Historia de usuario	
No.1	**Nombre**: Generar capa de acceso a datos.
Usuario: Desarrollador	**Tiempo Estimado**: 3 semanas
Prioridad en el negocio: Alta	**Nivel de complejidad**: Alto
Descripción: Generar la capa de acceso a datos correspondiente a una base de datos relacional existente.	
Información adicional (Observaciones): Da cumplimiento al RF 2 y el RF 5.	

Tabla No.2 HU Manipular la información en forma de Objetos

Historia de usuario	
No.2	**Nombre**: Manipular la información en forma de objetos.
Usuario: Desarrollador	**Tiempo Estimado**: 3 semanas
Prioridad en el negocio: Alta	**Nivel de complejidad**: Alto
Descripción: Manipular la información de la base de datos relacional a través de objetos patrón en código nativo PHP.	
Información adicional (Observaciones): Da cumplimiento al RF 1.	

Tabla No.3 HU Hacer consultas a la BD evitando sentencias SQL

Historia de usuario	
No.3	**Nombre**: Hacer consultas a la base de datos relacional evitando sentencias SQL.
Usuario: Desarrollador	**Tiempo Estimado**: 3 semanas
Prioridad en el negocio: Alta	**Nivel de complejidad**: Alto
Descripción: Hacer consultas a la base de datos relacional evitando sentencias SQL. Utilizando un dialecto propio que facilite la recuperación de información de la base de datos relacional.	
Información adicional (Observaciones): Da cumplimiento al RF 8.	

Tabla No.4 HU Permitir al usuario trabajar con varias conexiones a la BD al mismo tiempo

Historia de usuario		
No.4	**Nombre**: Permitir al usuario trabajar con varias conexiones a la base de datos relacional al mismo tiempo.	
Usuario: Desarrollador		**Tiempo Estimado**: 1 semana
Prioridad en el negocio: Media		**Nivel de complejidad**: Bajo
Descripción: El usuario necesita manejar varias conexiones a la vez. Obligarlo a trabajar con una sola conexión restaría flexibilidad.		
Información adicional (Observaciones): Da cumplimiento al RF 4.		

Tabla No.5 HU Integración con formatos de intercambio de información

Historia de usuario		
No.5	**Nombre**: Integración con formatos de intercambio de información.	
Usuario: Desarrollador		**Tiempo Estimado**: 1 semana
Prioridad en el negocio: Baja		**Nivel de complejidad**: Medio
Descripción: Integrar el framework con alguno de los formatos más usados para el intercambio de información.		
Información adicional (Observaciones): Estos formatos pueden ser XML y JSON. Da cumplimiento al RF 7.		

2.3.2 Iteraciones

Las iteraciones comienzan con las historias de usuario, cada historia de usuario deberá cumplirse al final de cada iteración. Esta debe ser corta y no es conveniente que dure más de tres semanas, desarrollándose sólo lo previsto en la iteración correspondiente.

2.3.2.1 Plan de Iteraciones

En el plan de iteraciones se incluyen las historias de usuarios y la estimación del tiempo de acuerdo a la velocidad de programación verificada con anterioridad. Es importante implementar primero lo esencial, si no alcanza el tiempo es más fácil adicionar o eliminar tareas. No se debe sacrificar la refactorización ni las pruebas de unidad, esto puede atrasar más la liberación de las versiones del producto.

Tabla No.6 Plan de iteraciones

Iteraciones	Orden de las historias de usuario a implementar	Tiempo de trabajo en semanas
Iteración 1	**HU1** Generar capa de acceso a datos.	**3**
Iteración 2	**HU2** Manipular la información en forma de objetos.	**3**
Iteración 3	**HU3** Hacer consultas a la base de datos relacional evitando sentencias SQL.	**3**
Iteración 4	**HU4** Permitir al usuario trabajar con varias conexiones a la base de datos relacional al mismo tiempo. **HU5** Integración con formatos de intercambio de información.	**2**

2.3.3 Plan de entrega

En el plan de entrega es importante tener presente los requerimientos del sistema para poder hacer una estimación correcta del tiempo de desarrollo del producto. Cuando se verifica un cambio de la velocidad de programación durante 2 ó 3 iteraciones, deberá rehacerse el Plan de Entrega y de esta forma no

defraudar al cliente. A continuación se exponen las versiones finales correspondientes al cumplimiento de cada iteración.

Tabla No.7 Plan de entregas

Entregable	Final 1ra Iteración	Final 2da Iteración	Final 3ra Iteración	Final 4ta Iteración
Framework ORM	**-**	**-**	**0.3.1**	**0.4.1**

2.3.4 Conclusiones

La definición de los requerimientos funcionales y no funcionales así como la preparación para la implementación son aspectos importantes a tener en cuenta para minimizar la cantidad de errores durante el proceso de codificación, además de proporcionar una guía para el mismo. Aunque XP es una metodología resistente a los cambios de requerimientos durante el desarrollo del producto, es esencial contar con un punto de partida para el comienzo de la fase de implementación.

El presente capítulo contiene la organización de los componentes y paquetes del producto para su implementación. XP como metodología de desarrollo no propone artefactos estrictos para llevar a cabo las tareas de diseño e implementación que van casi a la par. Esta parte del ciclo de vida de XP es según la predilección del programador, elegir según su preferencia los diagramas y herramientas a utilizar. Es muy práctico el uso de la Ingeniería Inversa cuando es más importante el producto final que la documentación correspondiente, esto no es más que generar la documentación a partir de la solución implementada.

3.1 Diseño del Sistema

3.1.1 Estrategias de Programación

Debido a que el lenguaje utilizado es interpretado, para la optimización del producto se tuvieron en cuenta varios detalles:

- Separación del código de las funcionalidades para usar en modo de desarrollo y modo de producción.
- Para la organización del código no se usaron muchos paquetes, para que el interpretador de PHP tenga que hacer un uso menor del sistema de archivos.
- Uso de caché para guardar información reusable.

3.1.2 Patrones de diseño utilizados

Registro

Es básicamente un almacén de objetos y servicios, que otros objetos dentro del sistema pueden llamar. Es esencialmente usado para implementar el sistema de caché del framework.

Método de Fabricación

Delega en una clase la responsabilidad de instanciar subclases especificadas. En la herramienta propuesta es utilizado para implementar la clase encargada de la conexión a la base de datos relacional y la implementación de sus manejadores.

Solitario o Instancia única

Asegura que una clase tenga una sola instancia y provee un punto global para su acceso. El uso de este patrón se evidencia en varias clases. Su uso más necesario se puede notar en la clase ModelManager, que es la encargada de administrar cada conjunto de clases que mapean una o varias bases de datos relacionales, y en la clase Autoload.

Mapa de Identidad

Asegura que cada objeto sea cargado sólo una vez, manteniendo cada objeto cargado en un mapa, para buscarlo cuando haya una nueva solicitud.

Mapeo de Llave Foránea

Mapea una asociación entre objetos, a una referencia de llave foránea entre tablas. Para el control de la navegabilidad entre los objetos relacionados, es necesario mantener las relaciones mapeadas.

Estado

Utilizado para monitorizar el estado de los objetos instanciados de la clase Record. Al cambiar uno o varios atributos de uno de estos objetos recuperado de la base de datos relacional, se reconoce cuales fueron los cambiados. Solamente estos se salvan enviando la consulta necesaria para efectuar la acción deseada.

Carga Perezosa

Permite la carga de los recursos y objetos utilizados solamente en el momento de la demanda. Es usado como estrategia para la inclusión de las clases del núcleo del sistema, y para cargar los objetos relacionados.

3.1.3 Convenciones

Para la construcción de la herramienta se tomaron en cuenta varias convenciones, con el fin de hacer más fácil la extensión y agregación de componentes al framework.

- Cada clase o interfaz, es implementada en un solo fichero exclusivo y único.
- Cada fichero contenedor de la implementación de una clase, es nombrado con el nombre de la clase seguido de **'.class.php'**, con excepción de las clases de manejadores de base de datos relacionales; por ejemplo: para la clase **'gbRecord'** el nombre del fichero correspondiente es **'gbRecord.class.php'**
- Cada componente manejador de base de datos relacional está compuesto por dos clases. Una clase con el nombre del manejador seguido de **'.drv.php'** (ejemplo: **'mysql.drv.php'**), que implementa la interfaz **gbDbDriver**. Y la otra clase tiene como nombre el manejador seguido de **'Ext'**, y su fichero tiene el nombre del manejador seguido de **'ext.drv.php'** (ejemplo: **'mysql.ext.drv.php'**) e implementa la interfaz **gbDbExtDriver**.

3.1.4 Paquetes y organización

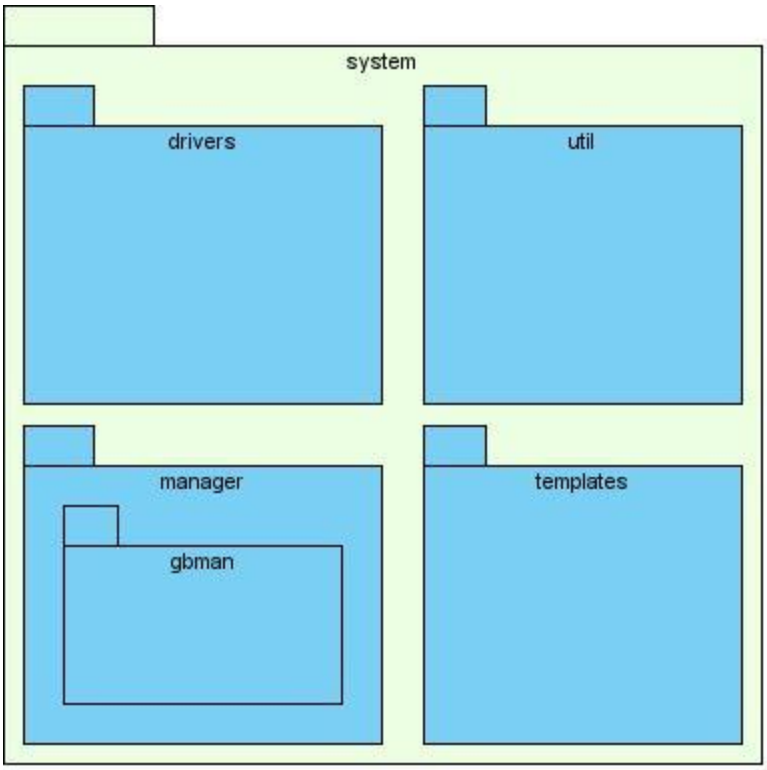

Figura 4 Paquetes y organización

system

Contiene el núcleo funcional del sistema. Dentro de este se encuentran todos los componentes necesarios para el funcionamiento del framework.

system.drivers

Contiene los manejadores de base de datos relacional utilizados para las conexiones y la extracción de los metadatos necesarios para esquematizar la base de datos relacional. Puede ser extendido agregándole componentes manejadores según las convenciones especificadas.

system.util

Es un paquete que contiene funcionalidades adicionales del framework, principalmente usadas para el desarrollo del mismo. Este paquete es extensible siguiendo las convenciones usadas para la construcción y la nomenclatura de ficheros contenedores de implementaciones de clases.

system.manager

Es el paquete contenedor del componente para realizar las tareas básicas del framework a través de una interfaz de usuario.

system.templates

Contiene las plantillas usadas para la generación de código. Este paquete es esencial para los componentes generadores de código; las plantillas que se encuentran en este definen el formato de los ficheros generados.

system.manager.gbman

En su interior se encuentra el componente de interfaz de usuario instalable. Por lo que juega un papel muy importante en la usabilidad del framework.

3.1.5 Diagrama de clases del sistema

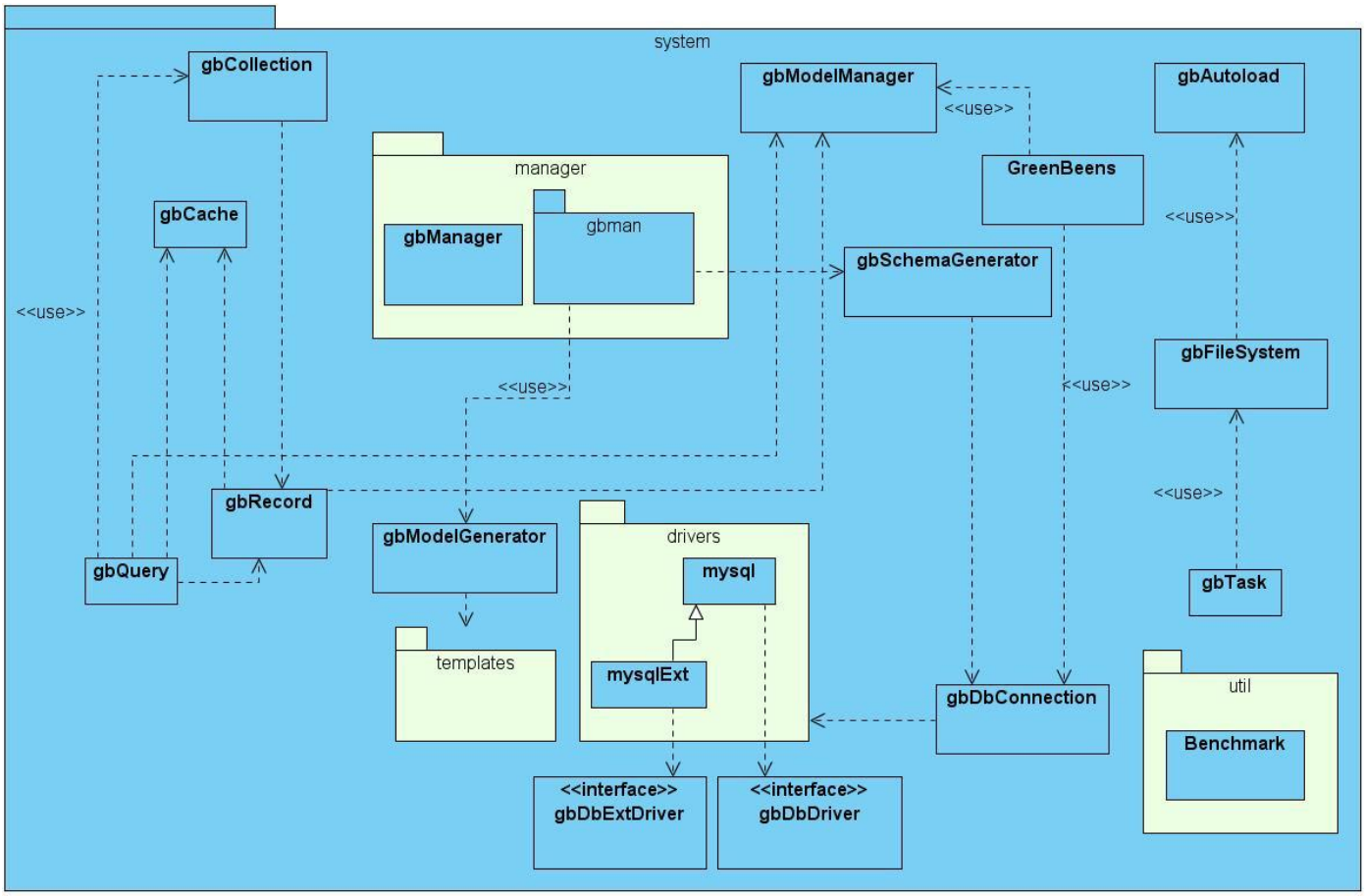

Figura 5 Clases del sistema

3.1.6 ¿Qué es un componente?

Los elementos físicos en UML 1 llamados componentes, son ahora llamados artefactos en UML 2. Un artefacto es una unidad física, como un fichero, ejecutable, script, base de datos y otros. Sólo los artefactos viven en nodos físicos; clases y componentes no tienen un lugar fijo. De todas maneras, un artefacto puede manifestar componentes y otros clasificadores (por ejemplo: clases). Un simple componente puede ser manifestado por múltiples artefactos, los cuales pueden residir en uno o diferentes nodos, entonces un componente puede ser indirectamente implementado en múltiples nodos.(Donald Bell, 2004)

Después de haber analizado el concepto de componente se muestra a continuación el diagrama de componentes perteneciente al framework ORM y seguidamente las especificaciones de cada componente, tales como el fichero existente y la clase e interface que contienen.

3.1.6.2 Diagrama de componentes

El diagrama de componentes muestra la relación entre los componentes del software, sus dependencias, localización y otras condiciones.

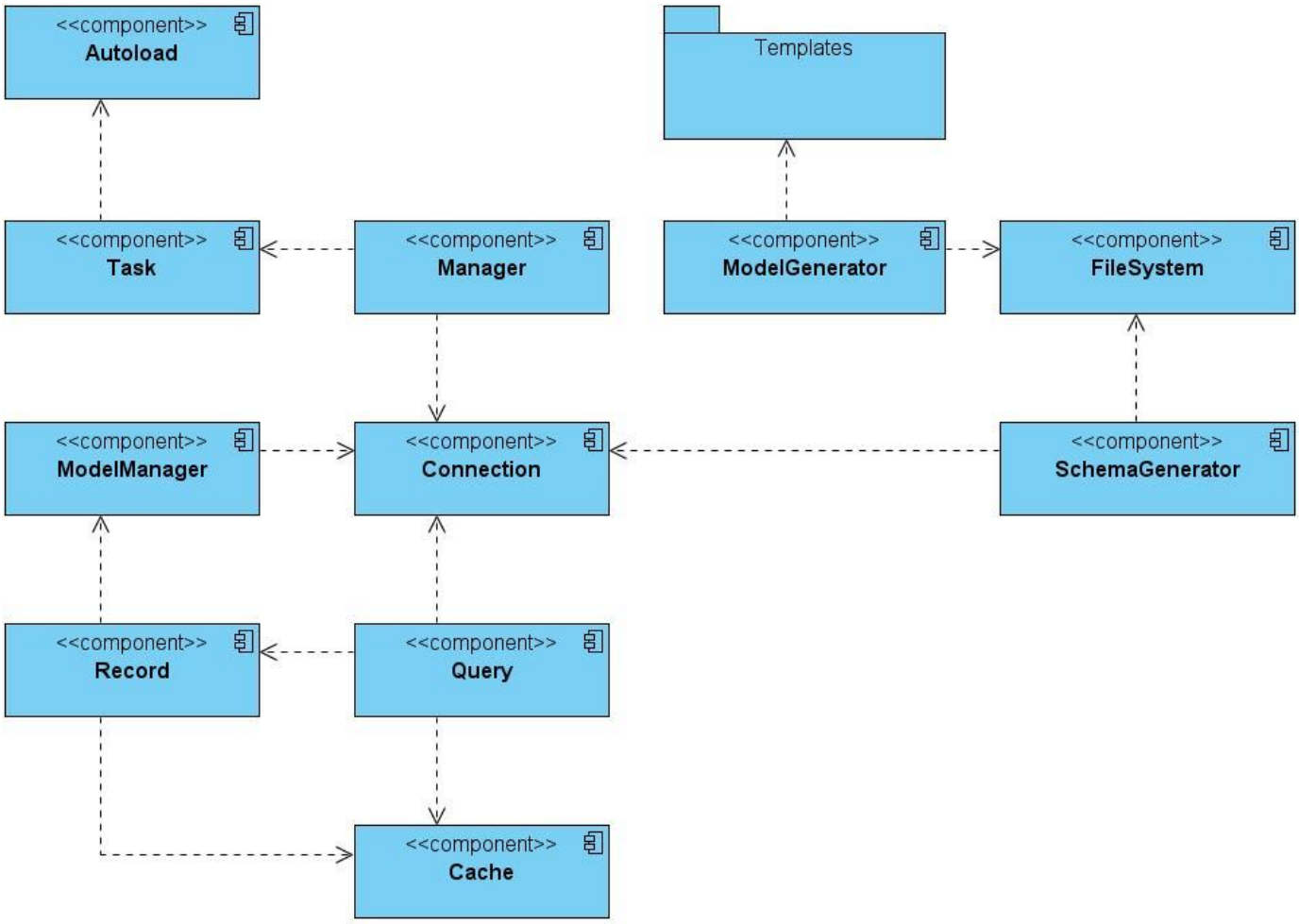

Figura 6 Diagrama de componentes

Tabla No.8 Componente Manager

Componente
Nombre: Manager
Este componente es la parte instalable para realizar las tareas básicas del sistema y ofrecer una interfaz gráfica.

Tabla No.9 Componente Task

Componente	
Nombre: Task	
Clase	**Fichero**
gbTask	gbTask.class.php

Tabla No.10 Componente Autoload

Componente	
Nombre: Autoload	
Clase	**Fichero**
gbAutoload	gbAutoload.class.php

Tabla No.11 Componente Connection

Componente	
Nombre: Connection	
Clase	**Fichero**
gbDbConnection	gbDbConnection.class.php
Interface	**Fichero**
gbDbDriver	gbDbDriver.class.php
gbDbExtDriver	gbDbExtDriver.class.php

Tabla No.12 Componente ModelManager

Componente	
Nombre: ModelManager	
Clase	**Fichero**
gbModelManager	gbModelManager.class.php

Tabla No.13 Componente Query

Componente	
Nombre: Query	
Clase	**Fichero**
gbQuery	gbQuery.class.php

Tabla No.14 Componente Cache

Componente	
Nombre: Cache	
Clase	**Fichero**
gbCache	gbCache.class.php

Tabla No.15 Componente ModelGenerator

Componente	
Nombre: ModelGenerator	
Clase	**Fichero**
gbModelGenerator	gbModelGenerator.class.php

Tabla No.16 Componente FileSystem

Componente	
Nombre: FileSystem	
Clase	**Fichero**
gbFileSystem	gbFileSystem.class.php

Tabla No.17 Componente SchemaGenerator

Componente	
Nombre: SchemaGenerator	
Clase	**Fichero**
gbSchemaGenerator	gbSchemaGenerator.class.php

Tabla No.18 Componente Record

Componente	
Nombre: Record	
Clase	**Fichero**
gbRecord	gbRecord.class.php

3.2 Implementación

XP propone comenzar la implementación con una arquitectura lo más flexible posible, para evitar grandes cambios durante el proceso de codificación del software en iteraciones posteriores. Al inicio de una iteración se tiene en cuenta la retroalimentación y experiencia que pudo aportar la culminación de las iteraciones anteriores.

3.2.1 Iteración 1

El objetivo de esta iteración es darle cumplimiento a la historia del usuario No.1 Generar capa de acceso a datos. Al final de esta iteración sólo se obtiene el componente funcional que genera la capa de acceso a datos que mapea a una base de datos relacional especificada. Para su cumplimiento se trazaron varias tareas.

Tareas de la iteración 1:

1. Estabilizar conexión.
2. Formato de esquema.
3. Formato de clases generadas.
4. Generar código.

Tabla No.19 Estabilizar conexión

Tarea	
No.1	**No. HU**: 1
Nombre de tarea: Estabilizar conexión	
Tipo de tarea: Desarrollo	**Estimación**: 5 días
Fecha inicio: 14/02/10	**Fecha fin**:18/02/10
Descripción: Desarrollar la capa de conexión a la base de datos relacional. Diseñarla de tal forma que sea extensible a través de manejadores para su ampliación en futuras versiones.	

Tabla No.20 Formato de esquema

Tarea	
No.2	**No. HU**: 1
Nombre de tarea: Formato de esquema	
Tipo de tarea: Configuración	**Estimación**: 5 días
Fecha inicio: 19/02/10	**Fecha fin**:23/02/10
Descripción: Documentarse acerca del estándar XML y definir el formato para almacenar el esquema (fichero .schm) construido a partir de la información recopilada de una base de datos relacional.	

Tabla No.21 Formato de clases generadas

Tarea	
No.3	**No. HU**: 1
Nombre de tarea: Formato de clases generadas	
Tipo de tarea: Configuración	**Estimación**: 5 días
Fecha inicio: 24/02/10	**Fecha fin**:28/02/10
Descripción: Definir el formato de las clases generadas a partir del fichero esquema. Construir las plantillas de clases, la cuales son un esqueleto de las clases a generar, que formaran parte de la capa de acceso a datos construida por el framework.	

Tabla No.22 Generar código

Tarea	
No.4	**No. HU**: 1
Nombre de tarea: Generar código	
Tipo de tarea: Desarrollo	**Estimación**: 6 días
Fecha inicio: 1/03/10	**Fecha fin**:6/03/10
Descripción: Desarrollar los componentes encargados de generar el código de un esquema y de las clases de acceso a datos correspondientes.	

3.2.2 Iteración 2

El principal objetivo de esta iteración es contar con la implementación de funcionalidades que permitirán al usuario abstraerse del lenguaje SQL para realizar las tareas básicas de gestión. La historia de usuario que corresponde darle cumplimiento es la No. 2: Manipular la información en forma de objetos. Al terminar la iteración 1 el equipo de desarrollo llegó al consenso de implementar un componente de auto-carga o carga automática, para facilitar el desarrollo de la herramienta y evitar errores de re-declaración de clases y re-inclusión de ficheros.

Tareas de la iteración 2:

1. Componente de auto-carga.
2. Patrón Row Data Gateway.
3. Extender funciones CRUD.

Tabla No.23 Componente de auto-carga

Tarea	
No.1	**No. HU**: 2
Nombre de tarea: Componente de auto-carga	
Tipo de tarea: Desarrollo	**Estimación**: 7 días
Fecha inicio: 7/03/10	**Fecha fin**:13/03/10
Descripción: Desarrollar un componente completamente reusable, para la carga de ficheros contenedores de clases. Este componente reconoce los ficheros siguiendo las convenciones establecidas para el desarrollo del framework.	

Tabla No.24 Patrón Row Data Gateway

Tarea	
No.2	**No. HU** :2
Nombre de tarea: Patrón Row Data Gateway	
Tipo de tarea: Desarrollo	**Estimación**: 7 días
Fecha inicio: 14/03/10	**Fecha fin**:20/03/10
Descripción: Implementar el patrón Row Data Gateway en lo que será la clase base, de la cual hereda cada clase mapeada de la base de datos relacional. Implementar las funciones básicas de gestión (CRUD).	

Tabla No.25 Extender funciones CRUD

Tarea	
No.3	**No. HU**: 2
Nombre de tarea: Extender funciones CRUD	
Tipo de tarea: Desarrollo	**Estimación**: 7 días
Fecha inicio: 21/03/10	**Fecha fin**:27/03/10
Descripción: Extender las funcionalidades implementadas en la tarea No. 2 de la iteración 2. Los objetos portadores de información no contarán solamente con las funciones CRUD.	

3.2.3 Iteración 3

Al finalizar esta iteración el sistema debe brindar una interfaz de consultas. Para esto es necesario darle cumplimiento a la historia de usuario No 3: Hacer consultas a la base de datos relacional evitando sentencias SQL. Con el objetivo de ofrecer un dialecto propio, compacto y orientado a objetos, para realizar pedidos a la base de datos relacional.

Tareas de la iteración 3:

1. Dialecto de consultas.
2. Clase de consultas.
3. Clase colección.

Tabla No.26 Dialecto de consultas

Tarea	
No.1	**No. HU**: 3
Nombre de tarea: Dialecto de consultas	
Tipo de tarea: Configuración	**Estimación**: 4 días
Fecha inicio: 28/03/10	**Fecha fin**: 3/04/10
Descripción: Definir el dialecto que va a implementar la clase proveedora de la interfaz de aplicación para realizar consultas a base de datos relacional.	

Tabla No.27 Clase de consultas

Tarea	
No.2	**No. HU**: 3
Nombre de tarea: Clase de consultas	
Tipo de tarea: Desarrollo	**Estimación**: 4,5 días
Fecha inicio: 4/04/10	**Fecha fin**: 10/04/10
Descripción: Implementar la clase para realizar consultas a la base de datos relacional teniendo en cuenta las especificaciones definidas en la tarea No. 1 de la iteración 3.	

Tabla No.28 Clase colección

Tarea	
No.3	**No. HU**: 3
Nombre de tarea: Clase colección	
Tipo de tarea: Desarrollo	**Estimación**: 3 días
Fecha inicio: 11/04/10	**Fecha fin**: 17/04/10
Descripción: Implementar una clase colección, contenedora de los objetos obtenidos como resultado de una consulta.	

3.2.4 Iteración 4

Tiene como principal objetivo dar cumplimiento a la historia de usuario No. 4: Permitir al usuario trabajar con varias conexiones a la base de datos relacional al mismo tiempo y la No. 5: Integración con formatos de intercambio de información. Al finalizar esta iteración, el framework contará con un sistema de memoria caché, con el objetivo de optimizar y reutilizar información.

Tareas de la iteración No. 4:

1. Clase administrador de modelos.
2. Clase caché.
3. Integración con JSON.

Tabla No.29 Clase administrador de modelos

Tarea	
No.1	**No. HU**: 4
Nombre de tarea: Clase administrador de modelos	
Tipo de tarea: Desarrollo	**Estimación**: 5 días
Fecha inicio: 18/04/10	**Fecha fin**: 22/04/10
Descripción: Desarrollar un sistema de administración de conexiones para el framework, capaz de manejar varias conexiones a la vez en tiempo de ejecución.	

Tabla No.30 Clase caché

Tarea	
No.2	**No. HU**: -
Nombre de tarea: Clase caché	
Tipo de tarea: Desarrollo	**Estimación**: 5 días
Fecha inicio: 23/04/10	**Fecha fin**: 27/04/10
Descripción: Desarrollar una clase de memoria caché, que permita guardar y recuperar información desde la memoria.	

Tabla No.31 Integración con JSON

Tarea	
No.3	**No. HU**: 5
Nombre de tarea: Integración con JSON	
Tipo de tarea: Desarrollo	**Estimación**: 4 días
Fecha inicio: 28/04/10	**Fecha fin**: 1/05/10
Descripción: Implementar la compatibilidad de la estructura de objetos portadores de información y colecciones, con el formato de intercambio de información JSON.	

3.3 Conclusiones

Después de haber desarrollado el concluyente capítulo, se ha obtenido una importante experiencia. Una buena forma de diseñar un framework, es primeramente concibiendo el concepto del objeto del cual se pretende agilizar o ayudar a su desarrollo, después definiendo las características que debe tener este objeto, y posteriormente buscar las soluciones.

El trabajo por iteraciones es importante en el desarrollo de software. Cada iteración pretende en su culminación, entregar una versión funcional no muy lejana de la entregada al finalizar la anterior. Consiste en incrementar gradualmente el espectro funcional del producto.

Capítulo 4 Pruebas

El presente capítulo aborda el ambiente de pruebas para la versión 0.4.1 del framework ORM en desarrollo. Se explican los diferentes tipos de pruebas hechas para comprobar el funcionamiento de la herramienta. XP propone durante todo el desarrollo de un software, la realización constante de pruebas unitarias, con el objetivo de verificar cada funcionalidad implementada o refactorizada.

4.1 Pruebas

Unas de las características de XP, es el hecho de ser una metodología de desarrollo dirigido por pruebas (TDD del inglés Test Driven Development).
Hay tres tipos de pruebas realizadas durante desarrollo de la herramienta, aplicadas de la siguiente manera:
Pruebas Unitarias: Estas pruebas son hechas durante el proceso de codificación, probar cada porción de código al terminar cada funcionalidad implementada. Las pruebas unitarias son características del periodo de implementación.
Pruebas de Integración: Al terminar la implementación de cada componente, es necesario hacer pruebas centradas en función del modulo construido. Además de validar su integración con los demás implementados anteriormente.
Pruebas de Aceptación: Estas pruebas son realizadas con el objetivo de probar el framework en todo su conjunto, a partir de las historias de usuario.

4.1.1 Pruebas de Aceptación

El cliente antes de recibir oficialmente la entrega de un producto esperado, hace pruebas de aceptación o caja negra. El objetivo es validar desde la perspectiva del usuario final, el cumplimiento de las especificaciones de los requerimientos del producto.

A continuación se muestran los casos de pruebas de aceptación realizados a las historias de usuario número uno y dos de la solución propuesta:

Tabla No.32 Caso de Prueba de aceptación No.1, HU1

Caso de Prueba de aceptación	
No.1	**Historia de usuario**: 1
Nombre: Crear esquema de una base de datos relacional.	
Descripción: Prueba funcional, partiendo de una conexión, generar el esquema de la base de datos relacional correspondiente.	
Condiciones de ejecución: Tiene que haber una base de datos relacional montada, sobre un sistema de base de datos relacional soportada por el framework.	
Ejecución: • Abrir la interfaz gráfica para especificar la conexión. • Pulsar el botón 'Generate'.	
Resultado esperado: Un fichero .schm que contiene el esquema estructural de la base de datos relacional a la cual se ha conectado.	
Evaluación: Satisfactoria	

Tabla No.33 Caso de Prueba de aceptación No.2, HU1

Caso de Prueba de aceptación	
No.2	**Historia de usuario**: 1
Nombre: Generar capa de acceso a datos	
Descripción: A partir de un esquema (.schm), generar las clases correspondientes, para el acceso a datos.	
Condiciones de ejecución: Un esquema previamente generado y ubicado en la carpeta 'schemes' del manager instalado.	
Ejecución: • Abrir la interfaz gráfica para seleccionar el esquema. • Pulsar el icono para generar el modelo del esquema escogido.	
Resultado esperado: Conjunto de clases mapa de una base de datos relacional especificada en un esquema (.schm)	
Evaluación: Satisfactoria	

Tabla No.34 Caso de Prueba de aceptación No.1, HU2

Caso de Prueba de aceptación	
No.1	**Historia de usuario**: 2
Nombre: Cargar Por	
Descripción: Para validar la funcionalidad Cargar Por de los objetos.	
Condiciones de ejecución: Tiene que haber una base de datos relacional montada, sobre un sistema de base de datos relacional soportada por el framework. Y debe estar previamente mapeada.	
Ejecución: • Se crea un nuevo objeto entidad. • Se invoca el método 'LoadBy' con los parámetros necesarios.	
Resultado esperado: El nuevo objeto entidad debe contener toda la información traída de la base de datos relacional, que corresponde a los parámetro de carga especificados en la invocación del método.	
Evaluación: Satisfactoria	

Tabla No.35 Caso de Prueba de aceptación No.2, HU2

Caso de Prueba de aceptación	
No.2	**Historia de usuario**: 2
Nombre: Eliminar	
Descripción: Para validar la funcionalidad Eliminar de los objetos.	
Condiciones de ejecución: Tiene que haber una base de datos relacional montada, sobre un sistema de base de datos relacional soportada por el framework. Y debe estar previamente mapeada.	
Ejecución: • Se crea un nuevo objeto entidad. • Se invoca el método 'LoadBy' con los parámetros necesarios. • Se invoca el método 'Delete'.	
Resultado esperado: La tupla correspondiente al objeto entidad cargado desde la base de datos relacional, debe ser borrada.	
Evaluación: Satisfactoria	

Tabla No.36 Caso de Prueba de aceptación No.3, HU2

Caso de Prueba de aceptación	
No.3	**Historia de usuario**: 2
Nombre: Obtener objeto relacionado.	
Descripción: Para validar la funcionalidad obtener objeto relacionado.	
Condiciones de ejecución: Tiene que haber una base de datos relacional montada, sobre un sistema de base de datos relacional soportada por el framework. Y debe estar previamente mapeada.	
Ejecución: • Se crea un nuevo objeto entidad. • Se invoca el método 'LoadBy' con los parámetros necesarios. • Se invoca el método 'getRelated' con los parámetros necesarios especificados.	
Resultado esperado: Devolver un objeto relacionado al nuevo objeto cargado desde la base de datos relacional.	
Evaluación: Satisfactoria	

Tabla No.37 Caso de Prueba de aceptación No.4, HU2

Caso de Prueba de aceptación	
No.4	**Historia de usuario**: 2
Nombre: Relacionar y Salvar	
Descripción: Para validar la funcionalidad de relacionar y salvar objetos.	
Condiciones de ejecución: Tiene que haber una base de datos relacional montada, sobre un sistema de base de datos relacional soportada por el framework. Y debe estar previamente mapeada.	
Ejecución: • Se crea un nuevo objeto a de una entidad A. • Se invoca el método 'LoadBy' de A con los parámetros necesarios. • Se crea un nuevo objeto b de una entidad B. • Se llenan las propiedades e b. • Se invoca el método de 'Bind' de a pasándole por parámetro el objeto b. • Se invoca el método 'Save' de a con los parámetros necesarios.	
Resultado esperado: Debe encontrarse en la tabla mapeada por la entidad B una tupla con los valores correspondientes al objeto b relacionado con la tupla de la tabla mapeada por la entidad A que corresponde al objeto a.	
Evaluación: Satisfactoria	

Tabla No.38 Caso de Prueba de aceptación No.5, HU2

Caso de Prueba de aceptación	
No.5	**Historia de usuario**: 2
Nombre: Insertar	
Descripción: Para validar la funcionalidad de insertar	
Condiciones de ejecución: Tiene que haber una base de datos relacional montada, sobre un sistema de base de datos relacional soportada por el framework. Y debe estar previamente mapeada.	
Ejecución: • Se crea un nuevo objeto de una entidad. • Se llenan sus propiedades. • Se invoca el método 'Save'.	
Resultado esperado: Debe encontrarse en la base de datos relacional, una nueva tupla correspondiente al nuevo objeto insertado.	
Evaluación: Satisfactoria	

Tabla No.39 Caso de Prueba de aceptación No.6, HU2

Caso de Prueba de aceptación	
No.6	**Historia de usuario**: 2
Nombre: Actualizar	
Descripción: Para validar la funcionalidad de actualizar	
Condiciones de ejecución: Tiene que haber una base de datos relacional montada, sobre un sistema de base de datos relacional soportada por el framework. Y debe estar previamente mapeada.	
Ejecución: • Se crea un nuevo objeto de una entidad. • Se invoca el método 'LoadBy' con los parámetros necesarios. • Se modifican sus propiedades. • Se invoca el método 'Save'.	
Resultado esperado: Debe encontrarse en la base de datos relacional, la tupla correspondiente al objeto recuperado de la base de datos relacional, modificada con los nuevos parámetros.	
Evaluación: Satisfactoria.	

4.2 Test de usabilidad

La **usabilidad** se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso. (ISO/IEC 9126)

Algunas personas piensan que la usabilidad es muy costosa y compleja y esas pruebas de usuarios deben ser reservadas para proyectos de un diseño Web raro, con un gran presupuesto y con tiempo de sobra en la planificación. Esto no es verdad. Las pruebas detalladas de usabilidad son una pérdida de recursos. Los mejores resultados se obtienen desde pruebas de no más de 5 usuarios y corriendo tantas pruebas pequeñas como pueda permitirse (Nielsen Jakob, 2000).

Después de haber realizado un estudio, para corroborar la usabilidad del sistema se utiliza el test de usabilidad, donde se seleccionaron un grupo de usuarios finales. La encuesta aplicada (Ver Anexo 2) se elabora según los principios que recomienda la universidad de Texas en Austin. Estos son:

- Información clave que los usuarios puedan encontrar en tu producto.
- Lo más importante. Si tienes mucha información clave, elegir las más importantes.
- Versión para la audiencia. Se debe hablar en el lenguaje de los usuarios finales.
- No usar preguntas dirigidas, más bien generales, para que los usuarios puedan opinar.
- Simpleza. La pregunta debe ser capaz de permanecer en la memoria del usuario mientras el usuario realiza alguna tarea para responderla, sin mirar de nuevo el cuestionario.
- Escenarios realistas. Las preguntas y tareas deben identificar las necesidades del usuario final.

(The University of Texas at Austin)

En la encuesta realizada todos coincidieron en que la API para realizar las consultas era fácil de usar y en la preferencia de la interfaz gráfica provista por el framework, sobre una interfaz de líneas de comandos. Se experimentaron adjetivos como: amigable, sencilla, fácil de usar, completa, flexible. En escala del 1 al 5, siendo 5 mayor complejidad de uso, se obtuvo una puntuación promedio de 1.85. Siendo de forma general la opinión positiva de los usuarios respecto a la usabilidad de la herramienta y la decisión de

seguir desarrollando la misma. Se plantearon algunas nuevas funcionalidades como la integración con frameworks existentes, incluir otros manejadores de base de datos, hacer pruebas sobre sistemas UNIX y continuar con la premisa de rapidez y usabilidad. Primaron las opiniones de ser un frameworks con grandes perspectivas.

Fueron encuestados:

1. Miguel Alejandro Martinez Rodriguez.
2. Fredy Hector Rios Morales.
3. Jorge Fernando Valdés Rodríguez.
4. Ing. Yunier Alexander Pimienta Fernández.
5. Ing. Adrian Gracia Aguila.
6. Ing. Alain Sánchez Gutiérrez.
7. Ing. Rolando Toledo Fernandez.

4.3 Conclusiones

La orientación a pruebas constituye una característica que fortalece la metodología usada. La constante realización de pruebas unitarias y pruebas de aceptación, favorece la estabilidad de cada versión consumada al final de cada iteración, la cual es la base para la iteración siguiente.

CONCLUSIONES

Una vez finalizado el desarrollo de este trabajo, se tiene la satisfacción de haber desarrollado un framework de mapeo objeto-relacional con una interfaz grafica para realizar sus tareas básicas, fácil de usar e integrado con el formato JSON. Uno de los aspectos que lo distingue de las demás herramientas de función similar, es la interfaz de programación de aplicación para realizar las consultas y la interfaz de usuario gráfica, los cuales aporta una gran usabilidad y componen potentes características para el desarrollo de capas de acceso a datos en PHP. La capacidad de extensión de la herramienta para la adición de nuevas funcionalidades, es una de las características que desde el principio formaron parte de su concepto, es fundamental para el desarrollo con base en componentes reusables y extensibles.

XP es la metodología que hizo posible la obtención de un framework funcional en muy poco tiempo. Provee al equipo de desarrollo de recursos flexibles en combinación con su preferencia, siendo su adaptación a los cambios lo que permite que un producto en desarrollo, dependiente de nuevas y cambiantes tendencias, se mantenga actualizado. PHP es un lenguaje de programación muy robusto, para la creación de aplicaciones con contenidos dinámicos, dependientes de base datos. Las mejoras que incorpora PHP 5 en cuanto a la programación orientada a objetos, lo hace un gran competidor frente otros lenguajes de programación y anima a su comunidad a aprovechar sus bondades en aplicaciones de escritorio.

RECOMENDACIONES

1. Continuar con el desarrollo de la herramienta, dotándola de nuevas funcionalidades.
2. Trabajar en la incorporación de características que la mantengan actualizada tanto en el ámbito de los frameworks ORM, como en el campo de los frameworks de abstracción de datos.
3. Integrar el framework con PDO (Objetos de Datos de PHP, extensión de PHP que provee una interfaz de programación para el trabajo con varios sistemas de base de datos).
4. Construir plugins de integración para otros frameworks de propósito general, para que los usuarios familiarizados con otras herramientas, puedan beneficiarse de su uso sin sacrificar compatibilidad ni preferencias.
5. Incluir validadores y soporte para otros sistemas de base de datos, como por ejemplo: Microsoft SQL Server, Postgres, SQLite y otros.
6. Proveer un componente para el manejo de excepciones.

REFERENCIA BIBLIOGRÁFICA

- MasterMagazine Archivo. Definición SQL. 2004. [citado febrero 22, 2010], Disponible en http://www.mastermagazine.info/termino/6771.php.
- K. Berry Douglas, Object-relational mapping articles. [citado febrero 9, 2010]. Disponible en http://www.service-architecture.com/object-relational-mapping/articles/.
- Departamento de Sistemas y Computación del Instituto Tecnológico de La Paz Introducción a los conceptos de bases de datos. [citado febrero 10, 2010]. Disponible en http://sistemas.itlp.edu.mx/tutoriales/basedat1/tema1_1.htm.
- Katrib Miguel, Programación orientada a objetos en C++. [citado febrero 10, 2010]. Disponible en http://eva.uci.cu/mod/resource/view.php?id=10652&subdir=/Miguel_Katrib_Programacion_Orientada_a_Objetos_en_CPP.
- Rojo Iñaki I., Open Source: los programas íntegros. octubre 1999. [citado febrero 22, 2010]. Disponible en http://www.baquia.com/com/legacy/8512.html.
- Universidad de alicante, El framework de colecciones de Java. 2009. [citado febrero 22, 2010], Disponible en http://www.dccia.ua.es/dccia/inf/asignaturas/RG/pdf/colecciones-java.pdf
- Zorrilla Unai, Hernández Octavio, Quintás Eduardo, Ado.Net Entity Framework, 2008. [citado febrero 22, 2010], Disponible en http://www.krasis.com/CampusMVP/libros/indices/Indice_Libro_Entity_Framework_Krasis_Press.pdf.
- Object Management Group.Introduction to OMG's Unified Modeling Language™ (UML®). [Citado febrero 16, 2010]. Disponible en http://www.omg.org/gettingstarted/what_is_uml.htm.
- Fowler Martin, P of EAA Catalog. [citado febrero 5, 2010]. Disponible en http://martinfowler.com/eaaCatalog/index.html.
- Leech, Mapeo Relacional de Objetos (ORM) en PHP. [citado enero 9, 2010]. Disponible en http://www.dr-leech.com.ar/2007/10/08/mapeo-relacional-de-objetos-orm-en-php/.

- XMundo, El sitio de Mozilla Addons cambia CakePHP por Django. noviembre 18, 2009. [citado diciembre 5, 2009]. Disponible en http://vivaphp.com.ar/frameworks/mozilla-addons-cambia-cakephp-por-django.
- Lim John, ADOdb Active Record. [citado febrero 5, 2010]. Disponible en http://phplens.com/lens/adodb/docs-active-record.htm.
- ezpdo4php, An Introduction to EZPDO, mayo 9, 2007. [citado febrero 6, 2010] Disponible en http://www.ezpdo.net/blog/?p=840.
- IBM Corporation. Eclipse Plataform Technical Overview. febrero 2003 actualizado para la versión 2.1, originalmente publicado julio 2001.
- Serafín Ana M., Quesada Alexander, Sistema automatizado para el control y gestión del transporte en el grupo de la electrónica, 2007. [citado marzo 5, 2010] Disponible en http://bibliodoc.uci.cu/TD/TD_0841_07.pdf.
- Armenteras Comellas Márius, Registry Pattern en PHP5, diciembre 18, 2007. [citado abril 5, 2010] Disponible en http://phparmy.wordpress.com/.
- Garzás Javier, Introducción a los patrones de diseño, 2000. [citado abril 5, 2010] Disponible en http://kybele.escet.urjc.es/documentos/SI/%5BSI-2006-07%5DT9_Patrones.pdf.
- Jaramillo Daniel E., Patrones de diseño, 2008. [citado abril 5, 2010] Disponible en http://www.slideshare.net/2008PA2Info3/info-exposicion-temas-de-revision.
- Paredes Adrián M., Patrones Enterprise (parte 2), diciembre 18, 2008. [citado abril 5, 2010] Disponible en http://elblogdelfrasco.blogspot.com/2008/12/patrones-enterprise-parte-2.html.
- Digital Equipment Corporation, Database Language SQL. Maynard, Massachusetts. julio 30, 1992. [citado abril 24, 2010] Disponible en http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt.
- Fabien Potencier, The Definitive Guide to symphony. Chapter 15 - Unit And Functional Testing. [citado febrero 18, 2010]. Disponible en http://www.symfony-project.org/book/1_0/15-Unit-and-Functional-Testing.

- Donald Bell, UML basics: The component diagram, 15 Diciembre 2004. [citado junio 15, 2010]. Disponible en http://www.ibm.com/developerworks/rational/library/dec04/bell.
- ISO 9241-11. Ergonomic requirements for office work with visual display terminals. ISO, 1998. [citado junio 13, 2010].
- Nielsen Jakob, Why You Only Need to Test with 5 Users, marzo 19, 2000. [citado junio 22, 2010]. Disponible en http://www.useit.com/alertbox/20000319.html.
- The University of Texas at Austin, Develop the Usability Test Documents, marzo 9, 2009 [citado junio 23, 2010] Disponible en http://www.utexas.edu/learn/usability/test.html.

BIBLIOGRAFÍA

- Hernández Sampieri, R. y otros 2003. Metodología de la Investigación. Segunda Edición. Ciudad de La Habana, Cuba. Editorial Félix Varela.
- Álvarez C. Z. (1994). Metodología de la investigación científica. Santiago de Cuba, Cuba. Centro de Estudio de Educación Superior, Universidad de Oriente.
- Doctrine query memory usage. noviembre 1, 2009. [citado noviembre 21, 2009]. Disponible en http://stackoverflow.com/questions/1412762/doctrine-query-memory-usage.
- Doctrine vs. Propel. mayo 16, 2009. [citado noviembre 25, 2009]. Disponible en http://codeutopia.net/blog/2009/05/16/doctrine-vs-propel-2009-update/.
- Comparing Propel And Doctrine. [citado diciembre 4, 2009]. Disponible en http://trac.symfony-project.org/wiki/ComparingPropelAndDoctrine.
- Improving Performance. [citado enero 25, 2010]. Disponible en http://www.doctrine-project.org/documentation/manual/1_1/en/improving-performance.
- Mato García Rosa María. Diseño de Bases de Datos. octubre 1999. [citado diciembre 9, 2009] Disponible en http://eva.uci.cu/file.php/75/Bibliografia_Basica/Libro_de_BD_de_Rosa_Maria.pdf.
- Deco Claudia, Bender Cristina. Tendencias actuales de Investigación en Base de Datos. [citado enero 28, 2010].
- Principles behind the Agile Manifesto, [citado enero 11, 2010] Disponible en http://agilemanifesto.org/principles.html.
- W. Ambler Scott, The Object-Relational Impedance Mismatch [citado enero 29, 2010]. Disponible en http://www.agiledata.org/essays/impedanceMismatch.html.
- UML Resource Page. [citado febrero 8, 2010]. Disponible en http://www.uml.org/.

- Lanzillotta Analía .MasterMagazine Archivo, Definición de Base de datos. 2004. [citado febrero 22, 2010]. Disponible en http://www.mastermagazine.info/termino/4012.php.
- W. Ambler Scott, Agile Data Home Page. 2009 [citado 25 febrero 2010]. Disponible en http://www.agiledata.org/.
- W. Ambler Scott, Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process. 2002 [citado 25 febrero2010].
- The Rules of Extreme Programming, 1999. [citado25 febrero 2010] Disponible en http://www.extremeprogramming.org/rules.html.
- Calero Solís Manuel, Una explicación de la programación extrema, 2003. [citado 26 febrero 2010] Disponible en http://www.apolosoftware.com/.
- López Barrios C, Metodología de Desarrollo (2): Programación Extrema, noviembre 29, 2005. [citado 26 febrero 2010].
- W. Ambler Scott, Introducción a las historias de los usuarios. 2009 [citado26 febrero 2010] Disponible en http://translate.googleusercontent.com/translate_c?hl=es&sl=en&u=http://www.agilemodeling.com/artifacts/userStory.htm&prev=/search%3Fq%3DScott%2BAmbler%2B%2BXP%26hl%3Des&rurl=translate.google.com.cu&usg=ALkJrhi4YvYBXoojRW6oGL2xJ55nXYoslg.
- Patrones de Acceso a Datos: Active Record, noviembre 22, 2008 [citado 27 febrero 2010] Disponible en http://grimpidev.wordpress.com/2008/11/22/patrones-de-acceso-a-datos-active-record/.
- Reglas ortográficas del español, 1998. [citado abril 2, 2010].
- Soto Lauro, Comparación Entre Bases De Datos Orientadas a Objetos Y Las Bases De Datos Relacionales Orientadas a Objetos. [citado 11 abril 2010]. Disponible en http://www.mitecnologico.com/Main/ComparacionEntreBasesDeDatosOrientadasAObjetosYLasBasesDeDatosRelacionalesOrientadasAObjetos.
- Martha Esthela Ruiz Tijerina, Bases de Datos, 2000. [citado 11 abril 2010]. Disponible en http://www.elrinconcito.com/articulos/BaseDatos/BasesDatos.htm.

- El ataque de los frameworks, 2008. [citado 15 mayo 2010]. Disponible en http://www.webtaller.com/maletin/articulos/el-ataque-de-los-frameworks.php.
- Welcome the Propel Website, 2009. [citado 15 mayo 2010].Disponible en http://www.propelorm.org/.
- Proyecto Nipe, Capítulo 5: Metodología aplicable a las normas NE AI. [citado 13 junio 2010]. Disponible en http://www.msc.es/estadEstudios/estadisticas/docs/08Capitulo5.pdf.
- Almansa Sáez Carmen, Martínez Paz José Miguel, Acercando Posturas sobre el Descuento Ambiental: Sondeo de Delphi a expertos en el ámbito internacional. [citado 13 junio 2010]. Disponible en http://www.scribd.com/doc/16166360/Ejemplo-de-validacion-de-investigacion.
- Verificación y validación. [citado 13 junio 2010]. Disponible en http://wwwdi.ujaen.es/asignaturas/computacionestadistica/pdfs/tema6.pdf.
- MC Martín Arribas Diseño y validación de cuestionarios. [citado 13 junio 2010]. Disponible en http://www.scribd.com/doc/687689/Diseno-y-Validacion-de-Cuestionarios.
- ISO 9241-11. Ergonomic requirements for office work with visual display terminals. ISO, 1998. [citado junio 13, 2010].
- J. Nielsen. Usability Engineering. AP Professional, 1993. [citado junio 13, 2010].
- Ferré Grau Xavier, Principios Básicos de Usabilidad para Ingenieros Software. [citado junio 14, 2010]. Disponible en http://is.ls.fi.upm.es/xavier/papers/usabilidad.pdf.
- Elementos de UML. [citado 15 junio 2010]. Disponible en http://docs.kde.org/stable/es/kdesdk/umbrello/uml-elements.html.
- Nielsen Jakob, Why You Only Need to Test with 5 Users, marzo 19, 2000. [citado junio 22, 2010]. Disponible en http://www.useit.com/alertbox/20000319.html
- The University of Texas at Austin, Develop the Usability Test Documents, marzo 9, 2009 [citado junio 23, 2010] Disponible en http://www.utexas.edu/learn/usability/test.html

- La RAE y las reglas de acentuación para el adverbio solo y los pronombres este, ese o aquel, 2010. [citado junio 24, 2010] Disponible en http://algomasquetraducir.com/2007/10/31/la-rae-y-las-reglas-de-acentuacion-para-el-adverbio-solo-y-los-pronombres-este-ese-o-aquel/

ANEXO 1

Pruebas de comparación de rendimiento entre Doctrine 1.5 y GreenBeens 0.4.1a

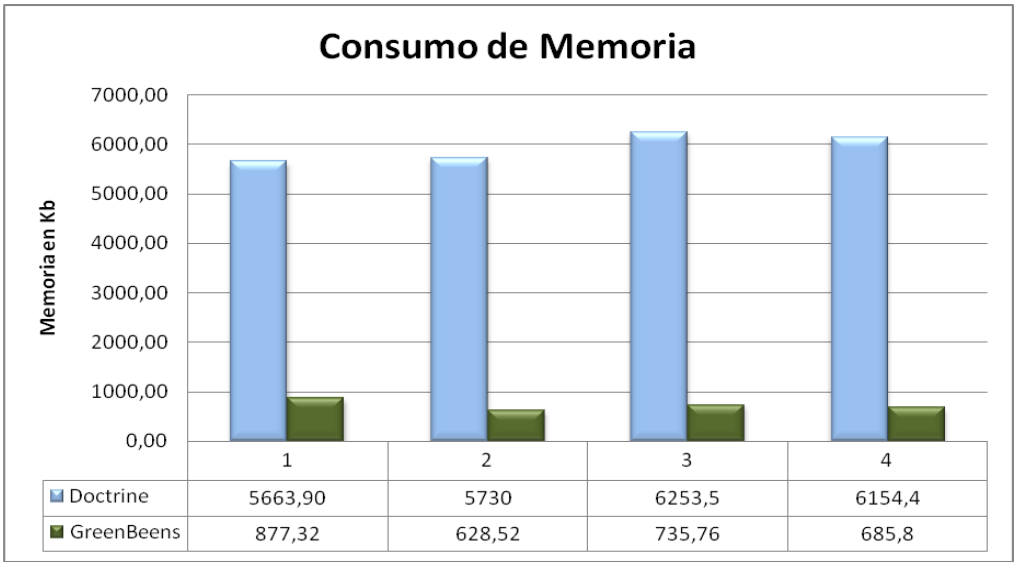

Figura 7 Consumo de memoria

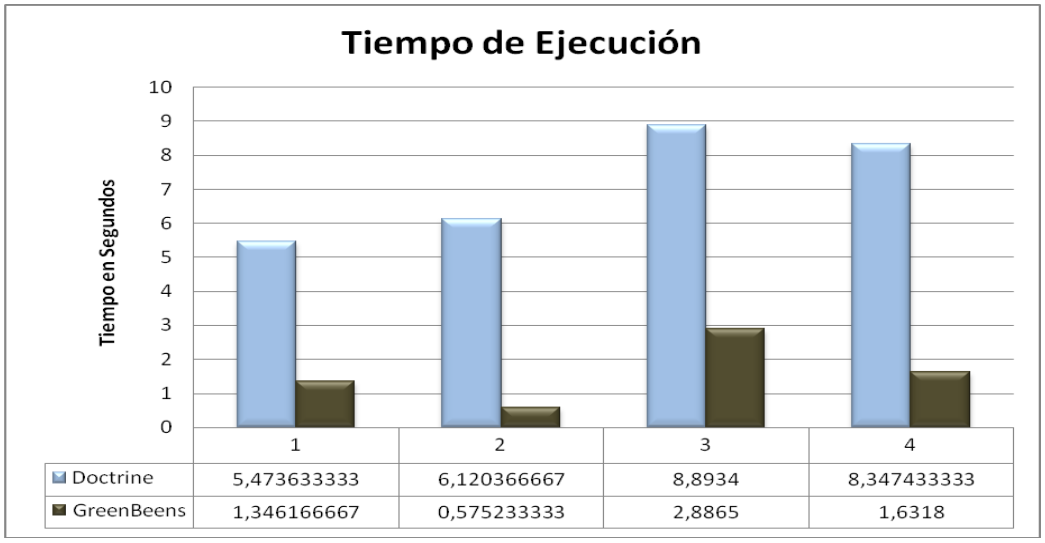

Figura 8 Tiempo de ejecución

ANEXO 2

Sobre el framework de Mapeo Objeto-Relacional GreenBeens opine.

1- En cuanto a facilidad de uso ¿Cree usted que la interfaz gráfica de usuario provista por este framework es un paso de avance comparado con la interfaz de línea de comando que ofrecen otros frameworks? ¿Por qué?

Si ___ No ___

Comentario:

2- En una oración, diga su opinión sobre la API para realizar consultas.

Comentario:

3- ¿Considera que la API para hacer consultas es fácil de usar?

Si ___ No ___

4- ¿Qué nivel de complejidad de uso usted cree que tiene la herramienta?

En escala del 1 al 5 (1 = fácil, 5 = compleja) Nivel de complejidad ___

5- ¿Cree usted que esta herramienta se debe seguir desarrollando? Si considera que si ¿En qué sentido cree que se debería mover el futuro desarrollo de la herramienta?

Si ___ No ___

Comentario:

6- En una oración, diga su opinión sobre el framework.

Comentario:

Nombre y Apellidos: