Universidad de las Ciencias Informáticas

Facultad 15

"Desarrollo de la versión 2.0 de la herramienta Doctrine Generator para la generación de ficheros de mapeo basado en Doctrine empleando la tecnología PHP"

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores: Maikel Yulier Sañudo Clemente

Leandro Luis Rojas Cuesta

Tutor: Ing. René Rodrigo Bauta Camejo

Ing. Yoel Blanco Torriente

Ciudad de La Habana

Junio, 2010

DECLARACIÓN DE AUTORÍA

Declaramos ser autores del presente trabajo y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmamos el presente a los ____ días del mes de _____ del año _____.

Maikel Yulier Sañudo Clemente

_____

Firma del Autor

Leandro Luis Rojas Cuesta

_____

Firma del Autor

Ing. René Rodrigo Bauta Camejo

_____

Firma del Tutor

RESUMEN

El uso de framework[1] que permitan la persistencia de bases de datos y tecnologías para la generación de ficheros de mapeo se ha convertido en una práctica muy difundida en los equipos de desarrollo de software, por lo que este tema está ocupando un papel primordial en su producción. En el proyecto ERP – Cuba[2] se emplea la herramienta Doctrine Generator para automatizar el mapeo de sus bases de datos mediante el framework de persistencia Doctrine. Sin embargo, muchas de las características de este sistema constituyen para el proyecto claras limitantes. Por ser una aplicación de escritorio implementada con software propietario el departamento de tecnología del proyecto no puede incluirla dentro de su marco de trabajo como herramienta, se hace dependiente del sistema operativo Windows y de sus framework correspondientes y es imposible su distribución.

El presente trabajo constituye la solución a los inconvenientes del Doctrine Generator mediante la implementación de una versión 2.0 que cumple con las necesidades del proyecto. Para lograrlo se desarrolla un marco teórico donde se estudian las herramientas y tecnologías a utilizar, con vistas al desenlace de las etapas de análisis, diseño e implementación de la solución. El resultado es una aplicación web guiada por las políticas de desarrollo del proyecto, de código abierto y multiplataforma, de la cual se validó cada etapa gracias a la aplicación de métricas a su diseño, pruebas de caja blanca a su implementación y de caja negra a sus requisitos funcionales descritos en el análisis.

Palabras Claves: framework, mapeo, persistencia, bases de datos.

[1] **Framework** estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación.

[2] **ERP – Cuba** es un proyecto productivo de la Universidad de Ciencias Informáticas, dedicado a la producción de software de gestión para las empresas cubanas. El mismo consta de un equipo multidisciplinario integrado por estudiantes, profesores y otros profesionales que trabajan en su conjunto para la obtención del producto final (Cedrux).

ÍNDICE DE CONTENIDOS

INTRODUCCIÓN

Con el transcurso de los años la diversificación y el crecimiento vertiginoso de los paradigmas informáticos que marcan el desarrollo de nuestros días, el software libre ha ganado un gran respeto e importancia como alternativa de soluciones conjuntas a problemas comunes. Así se evidencia en los cientos de miles de usuarios que lo comparten y las innumerables listas de empresas productoras de software que son cada día más rentables y lo asumen en su producción. De ahí que el software libre sea reconocido como una de las tecnologías más difundidas en la actualidad. Todo esto y más hacen que hoy en día sea este tipo de producto, uno de los líderes en áreas como la web, las infraestructuras básicas de red (incluyendo internet), la supercomputación, los pequeños electrodomésticos, las videoconsolas, el software para automoción, el inmenso mercado de software para móviles, el crítico software de tiempo real y otros. Tanto es así que se puede sintetizar que de una u otra manera un número elevado de los ciudadanos de los países del primer mundo son usuarios de software libre.

La evolución de esta tendencia no exceptúa a quienes la adoptan de posibles fallos y colisiones a la hora de entrar en los mercados de producción de software, donde casi siempre el cliente acostumbra a demandar mucho más de lo que la empresa es capaz de ofrecer, lo que resulta de vital importancia para la empresa a la hora de trazar sus estrategias y métodos, permitiéndole un mayor desarrollo en el ámbito profesional. Resultan soluciones válidas adoptadas en el mundo de la informática la creación de herramientas propias que sirven de ayuda para encaminar a la empresa hacia una solución óptima, así como la creación de nuevos framework y plugins (mejoras) que facilitan el desarrollo de la misma. Durante la obtención de un producto se pueden desarrollar varias herramientas que aunque no constituyen la solución final, contribuyen a la creación del resultado.

Un ejemplo de inconvenientes que conllevan al uso de herramientas como estas, son las tareas relacionadas con el manejo de datos. Las tendencias de la contemporaneidad están encaminadas hacia la programación orientada a objetos, la que como su nombre lo indica, maneja la información en forma de objetos que representan valores no escalares. Esta situación implica que el desarrollador debe ser capaz de convertir los valores de los objetos en grupos de valores simples para almacenarlos en la base de datos y luego de recuperarlos de estas, realizar el proceso inverso. El mapeo relacional de objetos (ORM) es utilizado en la implementación de esta aproximación, como técnica de programación para convertir

sistemas de datos de tipos diferentes. Técnicamente, se crea una base de datos virtual orientada a objetos a partir de la base de datos relacional existente y es justo con la primera que interactúa el programador, puesto que la misma brinda las características esenciales de la orientación a objetos basados en herencia y polimorfismo, permitiendo mayor robustez y potencialidad de implementación. En la actualidad existen diversos paquetes tanto libres como comerciales, que ayudan con el mapeo relacional de objetos, aunque, todavía muchas empresas y desarrolladores independientes prefieren implementar sus propias herramientas que solucionen el problema.

En la Universidad de Ciencias Informáticas (UCI) existen proyectos productivos en los que se desarrollan software fundamentalmente basados en la programación orientada a objetos, por lo que se hace notoria la utilización de diversos ORM seleccionados de acuerdo a su compatibilidad con los lenguajes de programación, la lógica del negocio y el diseño de software definidos por dichos proyectos. Dentro de ellos se pueden citar el potente Hibernate, utilizado en proyectos como Nefrología y Hospitales que tienen su implementación en Java, así como Propel para los desarrolladores en Symphony[3] del proyecto Control Sanitario Internacional, pertenecientes a la facultad 7 de la universidad y NHibernate para los desarrolladores en .NET[4] del proyecto Registro y Notarías de la facultad 15. Por su parte, el proyecto ERP-Cuba adoptó para su desarrollo el uso de Doctrine como ORM candidato en la persistencia de datos, que goza actualmente de una amplia comunidad de desarrollo y documentación disponible superior y más eficiente que la de muchos otros framework de persistencia, que ha ido en aumento en los últimos tiempos y lo convierte en uno de los más potentes para PHP. Además, por las facilidades que brinda su integración con Zend Framework[5], adoptado también por el proyecto, sin mencionar su importante lenguaje de consultas orientado a objetos denominado DQL (Lenguaje de Consultas Doctrine) e inspirado en el HQL (Lenguaje de Consultas Hibernate) de Hibernate.

Pero el simple hecho de definir una buena técnica de programación para el mapeo en el proyecto ERP y las ventajas que esta brinda, aun no resuelven todos los problemas, ya que a medida que se comenzó a utilizar esta práctica se observaron algunas limitantes que en un principio concebían más engorroso el manejo de los datos, con probabilidad de errores y requiriendo un mayor esfuerzo del programador. La ausencia en el mapeo de las relaciones entre las tablas y de las secuencias en sus campos llaves, así

[3] **Symfony** es un poderoso framework para crear aplicaciones web en PHP y la forma más sencilla de aumentar la productividad y calidad de tu trabajo.
[4] **.NET** es un framework propietario de Microsoft para el desarrollo de software. Provee un extenso conjunto de soluciones predefinidas para necesidades generales de la programación de aplicaciones.
[5] **Zend Framework** (ZF) es un framework de código abierto para desarrollar aplicaciones y servicios web con PHP 5 o superior basado en la programación orientada a objetos.

como la introducción de consultas DQL eran procesos que tenían que hacerse manualmente, y en bases de datos muy grandes solía ser trabajo de mucho tiempo. Estos inconvenientes atentaban contra el eficaz desarrollo en tiempo de las actividades y la robustez del producto a desarrollar.

El Doctrine Generator es una herramienta que surge para facilitar la persistencia de datos y aminorar las dificultades que existían en esta operación y de esta manera enriquecer el trabajo e intercambio de datos en forma de objetos con las bases de datos relacionales del proyecto. Dicha herramienta trajo grandes ventajas para los desarrolladores del proyecto, evitando atrasos en las actividades del cronograma y la introducción de errores en la implementación del producto Cedrux obtenido en el proyecto. Pero unido a sus ventajas se sumó la dificultad de incluirlo dentro del paquete de herramientas (PACSOFT) desarrolladas en el proyecto**,** y de su paquete de instalación, por estar implementado en .NET, convirtiéndose en un software propietario con políticas y características que se alejan de los estándares seguidos por el proyecto ERP y el país. Otra de sus limitantes lo constituye el hecho de que esta sea una aplicación de desktop (escritorio), por lo que debe ser instalada en cada puesto de trabajo que se vaya a utilizar, imponiendo como condiciones el uso de un único sistema operativo (Windows en este caso) y la previa instalación del framework necesario para el uso de aplicaciones desarrolladas en esta plataforma, lo cual limita su usabilidad. Además, como todo software propietario se hace imposible su distribución e inminente el pago de su licencia, en momentos en que Cuba se encuentra en una situación económica difícil.

De acuerdo a lo antes expuesto queda definido el siguiente **problema científico:**

La implementación de la herramienta Doctrine Generator para la generación de ficheros de mapeo basada en Doctrine empleando la tecnología propietaria .NET impide su integración con el paquete de herramientas del marco de trabajo del ERP.

Tomando en este caso como **objeto de estudio:**

La persistencia de objetos en esquemas de base de datos relacionales.

Centrado mayormente en el **campo de acción:**

Los software para la generación de ficheros de mapeo en base de datos relacionales.

Para dar solución a esta problemática se determinó como **objetivo general:**

Desarrollar la versión 2.0 de la herramienta Doctrine Generator empleando tecnología PHP, para integrarla con el paquete de herramientas del Marco de Trabajo del ERP.

Del objetivo general se desglosan los siguientes **objetivos específicos:**

1. Elaborar el marco teórico de la investigación.
2. Desarrollar la aplicación propuesta.
3. Analizar los resultados obtenidos de la validación del diseño propuesto y de la calidad del sistema.

En correspondencia con el objetivo general se establecieron las siguientes **tareas investigativas:**

1. Realizar un estudio del estado del arte de las principales herramientas y framework relacionados con el objeto de estudio de la investigación.
2. Realizar un estudio y análisis de la herramienta para la generación de ficheros de mapeo Doctrine Generator.
3. Definir la metodología y las tecnologías a emplear en el desarrollo de la herramienta.
4. Desarrollar el modelo de diseño del sistema.
5. Desarrollar el modelo de implementación del sistema.
6. Diseñar e implementar los algoritmos a usar en la herramienta teniendo en cuenta buenas prácticas de programación.
7. Evaluar la calidad del diseño de la solución a través de métricas de validación.
8. Evaluar la calidad del software desarrollado a través de las pruebas definidas por el Grupo de Calidad del CEIGE (Centro de Informatización de la Gestión de Entidades).

La presente investigación tiene como **idea a defender:**

Si se realiza el diseño e implementación de la versión 2.0 de la herramienta Doctrine Generator para la generación de ficheros de mapeo empleando la tecnología PHP, se podrá integrar con el paquete de herramientas del marco de trabajo del ERP.

Para el estudio investigativo realizado se utilizaron diversos métodos tanto teóricos como empíricos, ellos son:

Métodos Teóricos

Histórico – Lógico

Vinculado al conocimiento de las distintas etapas de los objetos en su sucesión cronológica. Para conocer la evolución y desarrollo del objeto o fenómeno de investigación se hace necesario revelar su historia, las etapas principales de su desenvolvimiento y las conexiones históricas fundamentales.

Este método científico, permitió realizar un estudio del estado del arte de los principales ORM existentes para PHP que se utilizan en la actualidad, así como de las posibles herramientas generadoras de los ficheros de mapeo para la persistencia de esquemas de bases de datos, dentro de las que se incluye la

primera versión del Doctrine Generator. Además, facilitó el análisis evolutivo de dichas herramientas en materias de arquitectura y diseño, así como su tendencia actual.

Analítico – Sintético

Su objetivo es distinguir los elementos de un fenómeno y se procede a revisar ordenadamente cada uno de ellos por separado. Consiste en la extracción de las partes de un todo, con el objeto de estudiarlas y examinarlas por separado. Estas operaciones no existen independientes una de la otra; el análisis de un objeto se realiza a partir de la relación que existe entre los elementos que conforman dicho objeto como un todo; y a su vez, la síntesis se produce sobre la base de los resultados previos del análisis.

La utilización de este método facilitó el estudio por separado de cada una de las herramientas generadoras de ficheros de mapeo y los ORM que más utiliza PHP en la actualidad, determinando las características que presentan en común y estableciendo una series de parámetros de acuerdo a sus objetivos fundamentales, con el fin de lograr una comparación entre ellos y valorar los resultados de interés para la investigación en curso.

Inducción – Deducción

A través de este se estudian los caracteres y conexiones necesarios del objeto de investigación, relaciones de causalidad, entre otros. Este método se apoya en métodos empíricos como la observación y la experimentación.

Al aplicar este método se estudiaron los generadores de ficheros de mapeo y el Doctrine como ORM seleccionado por el proyecto, con el fin de definir sus características particulares, y basado en todas ellas se definieron, requisitos y cualidades que debe cumplir el sistema que se propone.

Métodos Empíricos

Observación

Con su utilización se centró toda la atención en la situación actual existente en el proyecto ERP- Cuba y en los inconvenientes del Doctrine Generator como alternativa de mapeo desarrollada y en uso, a la hora de incluirlo como herramienta del marco de trabajo del proyecto por su implementación incompatible con los estándares de diseño y desarrollo de software seguidos por el proyecto.

El presente trabajo se encuentra estructurado en tres capítulos, resumidos de la siguiente forma:

Capítulo 1

Se realiza la fundamentación teórica del trabajo, incluyendo un estudio del estado del arte del tema a nivel nacional e internacional, de las tecnologías, metodologías y software usados en la actualidad con el mismo objetivo, para llegar a los resultados que se esperan obtener.

Capítulo 2

Brinda una fundamentación de la solución propuesta, a partir de la cual se describen las actividades de análisis de la solución, seguidas por la descripción de los procesos del sistema y de las etapas de diseño e implementación que conllevan a la obtención del software.

Capítulo 3

Valida la calidad del diseño del software mediante la aplicación de métricas basadas en atributos de calidad y donde además, se aplican pruebas de caja negra a la solución en aras de garantizar el cumplimiento de los requisitos propuestos y la satisfacción del cliente.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se hace referencia a la fundamentación teórica del trabajo, donde se realiza un estudio del estado del arte del framework de persistencia Doctrine y de los sistemas existentes que automatizan el proceso de mapeo de los datos en bases de datos relacionales. Debido a que el software a desarrollar formará parte de las aplicaciones creadas en el proyecto ERP, se estudiaron las tecnologías y herramientas informáticas definidas por este para el desarrollo, tales como los IDE de desarrollo, framework, lenguajes de programación, herramientas case, controladores de versiones y navegadores web.

También se hace un análisis de las metodologías de desarrollo de software candidatas a seguir para el desarrollo de la aplicación, con las que se definen los artefactos fundamentales a generar con vistas a la posible solución que conllevará este trabajo.

1.2 Diseño de interfaces

La aplicación de correctos estándares de diseño de software que cumplan con las políticas y normativas trazadas por la dirección del proyecto influyen considerablemente en la calidad y eficiencia de los resultados a obtener en el mismo. Condicionando además, la aceptación del cliente y calibrando a su vez la capacidad del proyecto o equipo de desarrollo de producir un software de calidad.

La consecución de los objetivos perseguidos a través de la puesta a disposición del público de cualquier aplicación Web está condicionada por la satisfacción del usuario final. Con el propósito de lograr un estándar de diseño que satisfaga las necesidades y exigencias de los clientes, sea compatible con las tecnologías utilizadas, cumpla con todos los objetivos perseguidos y que represente al proyecto a manera de sello y presentación de su trabajo, a continuación se presentan algunas de las propuestas que constituirán una pauta a cumplir por todas las aplicaciones Web de tipo gestión a desarrollar.

Algunas de las reglas a cumplir:

- *La interfaz corresponderá con las características, metas y nivel de experiencia de los usuarios de la aplicación, siempre que no se atente contra las pautas para el diseño.*
- *La información estará libre de errores gramaticales, ortográficos y tipográficos.*
- *La taxonomía de los títulos encima de los controles estará según la estipulada, primera letra con mayúscula, el resto con minúscula.*

- *En controles de tipo [input, select, text-área] cambiará el color de fondo indicando que el cursor está situado en el para modificar los datos. Una vez que el cursor deja de estar situado sobre ellos, regresarán a su estado normal. (Garantizado por Ext.).*
- *En controles de tipo button cambiará el color de fondo indicando que el cursor está situado en el para presionar. Una vez que el cursor deja de estar situado sobre ellos, regresarán a su estado normal. El cursor mantendrá la forma de puntero. (Garantizado por Ext.).*
- *Los botones en todas las barras de herramientas se corresponderán con los comandos de un menú. (Garantizado por Ext.).*
- *Los botones de comando serán de tamaño y forma similares. (Garantizado por Ext.).*
- *Solo se presentará al usuario la información que realmente necesita.*

(UCID, 2008)

1.3 Tecnologías

Una tecnología es un conjunto ordenado científicamente de conocimientos técnicos, instrumentos, procedimientos y métodos aplicados en las distintas ramas industriales con el principal objetivo de satisfacer las necesidades de las personas, con esta se ayuda la producción, en algunos casos puede reducir los costos pero también trae como consecuencias: contaminación, despido masivos de obreros y costo social alto. (Universidad Nacional Autónoma de México, 2008)

Es de suma importancia que los que utilicen las tecnologías sepan aplicarla para el bien de la sociedad para que la misma influya en el progreso social y económico.

Desde el punto de vista informático, las tecnologías no son más que el estudio, diseño, desarrollo, puesta en práctica, ayuda o gerencia de los sistemas informáticos computarizados, particularmente usos del software y hardware, es decir, los programas e instrumentos que se utilizan para la elaboración de software, el uso de computadoras y de software electrónico para convertir, almacenar, proteger, procesar, transmitir y recuperar la información.

1.3.1 Programación orientada a objetos

La programación orientada a objetos (POO) es una forma de estructurar un programa sobre la base de objetos. Cada elemento o componente en un programa que se base en esta técnica es concebido como

un objeto que tiene propiedades y métodos. La ejecución de un programa depende pura y exclusivamente de una interacción de los objetos que lo componen.

Las propiedades y los métodos de los objetos se especifican en su clase. Una clase de objeto vendría a ser el molde de cada instancia particular del objeto. Por lo tanto, cuando se programa una aplicación orientada a objetos, se definen clases y luego se crean instancias de objetos a través de esas clases para que interaccionen entre sí.

Como parte de este tipo de programación en el libro "*Programación y algoritmos" de* Maximiliano Bonanata se define el concepto de herencia como *"(…) una característica importantísima en los objetos. Permite definir propiedades y métodos de un objeto a partir de las propiedades y los métodos de otros objetos. De esta forma, se produce un efecto de herencia de propiedades y métodos entre ellos (…)"* y el de polimorfismo como *"(…) una característica que nos permite definir un mismo método en varios objetos, pero con distintos comportamientos."* (Bonanata, 2003)

1.3.2 Lenguajes de programación

"Un lenguaje de programación es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse en la creación de programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana". (O'Reilly, 2003)

Los lenguajes de programación son herramientas que permiten crear programas y software. Disponen de formas adecuadas que permiten ser leídas y escritas por personas y a su vez resultan independientes del modelo de computador a utilizar, representan en forma simbólica y en manera de un texto los códigos que podrán ser leídos por una persona e interpretados por dicho computador.

PHP

PHP es un lenguaje de programación interpretado, originalmente diseñado para la creación de páginas web dinámicas. Sigue un estilo clásico ya que es un lenguaje de programación con variables, sentencias condicionales, bucles, funciones, etc. Está más cercano a JavaScript[6] o al lenguaje C[7], pero a diferencia de Java o JavaScript que se ejecutan en el navegador, PHP se ejecuta en el servidor, lo que permite

[6] **JavaScript** es un lenguaje interpretado, desarrollado por la compañía de Netscape para incrementar las funcionalidades del lenguaje HTML de páginas web.

[7] **C** es un lenguaje de programación creado en 1972 por Dennis M. Ritchie en los Laboratorios Bell. Es apreciado por la eficiencia del código que produce y es muy utilizado para crear software de sistemas y aplicaciones.

acceder a los recursos que este tenga, como es una base de datos y el resultado es enviado al navegador, el cual podría ser una página HTML o de cualquier otro tipo. (Universidad Tecnica Particular de Loja, 2009)

PHP no necesita que el navegador lo soporte, es independiente de este, pero sin embargo para que sus páginas funcionen, el servidor donde están alojadas debe soportar este lenguaje, que tiene numerables ventajas sobre otros lenguajes de programación que se ejecutan de igual manera (como podrían ser los script CGI Perl), permitiendo intercalar las sentencias PHP en las páginas HTML.

ExtJS

No es más que una librería JavaScript ligera y de alto rendimiento, compatible con la mayoría de los navegadores web, que permite crear páginas e interfaces web dinámicas usando tecnologías AJAX, DHTML y DOM. Está librería incluye componentes UI (Interfaces de Usuario) de alto funcionamiento y personalizables, modelo de componentes extensibles, una API (interfaz de programación de aplicaciones) fácil de usar y licencias Open Source (código abierto) y comerciales.

Antes de poder entrar a examinar ExtJS se hace necesario mencionar los RIA, acrónimo de Rich Internet Applications (Aplicaciones Ricas en Internet). Intenta proveer lo que siempre ha adolecido la web, una experiencia de usuario muy parecida o igual a la que se tiene en las aplicaciones de escritorio. Las aplicaciones web tradicionales tienen problemas como la recarga continua de las páginas cada vez que el usuario pide un nuevo contenido, o la poca capacidad multimedia, para lo cual se han hecho necesarias mejoras externas.

En los últimos años han surgido diversas tecnologías, basadas en Flash (Adobe Macromedia), Java (Sun). Todas muy interesantes, pero con la desventaja de necesitar algún tipo de extensión en los navegadores que podría no estar presente. Ha sido esta limitante lo que le ha dado la victoria (al menos por el momento) al uso combinado de JavaScript y la “nueva” tecnología conocida como AJAX.

ExtJS encaja dentro de este esquema como un motor que permite crear aplicaciones RIA mediante JavaScript. Una de las grandes ventajas que posibilita el uso de ExtJS es que permite la creación de aplicaciones complejas utilizando componentes predefinidos así como un manejador de layouts (trazados) similar al que provee Java Swing, gracias a esto provee una experiencia consistente sobre cualquier navegador (Firefox, IE, Safari, etc.), evitando el tedioso problema de validar que el código escrito funcione bien en cada uno de ellos.

Usar un motor de render (dibujado) como ExtJS permite tener además otros beneficios:

- Existe un balance entre Cliente – Servidor. La carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar un número mayor de clientes al unísono.
- Comunicación asíncrona. En este tipo de aplicación el motor de render puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se dé cuenta.
- Eficiencia de la red. El tráfico de red puede disminuir al permitir que la aplicación elija que información desea transmitir al servidor y viceversa, sin embargo, la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico.

1.3.3 Entorno de desarrollo integrado (IDE)

IDE (acrónimo que significa integrated development environment) en español sería entorno de desarrollo integrado, es un programa informático compuesto por un conjunto de herramientas de programación, en su ambiente de trabajo pueden utilizarse uno o varios lenguajes, por lo que son empaquetados en una única aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDE pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. (Universidad de la Republica de Uruguay, 2009)

En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto.

Es posible que un mismo IDE trabaje con varios lenguajes de programación. Este es el caso de Eclipse, al que mediante plugins, se le puede añadir uno o más soportes de lenguajes adicionales.

Zend Studio

Zend Studio o Zend Development Environment es un completo entorno de desarrollo integrado para el lenguaje de programación PHP, fue elaborado principalmente para este lenguaje de programación, pero también ofrece soporte básico para otros lenguajes web, como HTML, JavaScript y XML. Está escrito en el lenguaje Java y está disponible para las plataformas de Microsoft Windows, Mac OS X y GNU/Linux.

Dentro de sus características principales es válido señalar que no requiere la instalación previa de PHP ni del entorno de ejecución de Java, tiene soporte para PHP 4 y PHP 5, tiene detección de errores de

sintaxis en tiempo real, soporte para gestión de grandes proyectos de desarrollo y un manual de PHP integrados.

Aptana

Aptana es un entorno de desarrollo especializado en la programación de aplicaciones dinámicas para web, con especial soporte para JavaScript. Su uso es muy sencillo e intuitivo, basado en Eclipse[8], con la capacidad de ofrecer subida, descarga y sincronización con FTP/SFTP[9]. Además, cuenta con interesantes detalles como un novedoso índice de funciones y un depurador integrado capaz de avisar si hay errores en el código.

1.3.4 Framework

"Framework se define en términos generales como un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que puede ser utilizada como referencia para enfrentar y resolver nuevos problemas de índole similar.

Más asociada a la informática en el desarrollo de software es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otras aplicaciones para ayudar a desarrollar y unir los diferentes componentes de un proyecto, es decir, son soluciones completas que llevan incorporado herramientas de apoyo a la construcción (ambiente de trabajo o desarrollo) y motores de ejecución (ambiente de ejecución).

Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio". (Commons, 2009)

Zend Framework

"Zend Framework es un framework de código abierto para desarrollar aplicaciones y servicios web con PHP5. Zend Framework es una implementación que usa un código totalmente orientado a objetos. La estructura de los componentes de Zend Framework es algo único; cada componente está construido con

[8] **Eclipse** es un entorno de desarrollo integrado (IDE) gratuito, que sirve para muchos lenguajes. Tiene interesantes utilidades para Java evidenciadas en sus módulos o plug-in.

[9] **FTP/SFTP** es un servidor especial que se ejecuta en un equipo servidor. Su función es permitir el intercambio de datos entre diferentes servidores/ordenadores conectados a la misma red.

una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado. A menudo se refiere a este tipo de diseño como "use-at-will" (uso a voluntad).

Aunque se pueden utilizar de forma individual, los componentes de la biblioteca estándar de Zend Framework conforman un potente y extensible framework de aplicaciones web al combinarse. Zend Framework ofrece un gran rendimiento y una robusta implementación MVC[10], una abstracción de base de datos fácil de usar, y un componente de formularios que implementa la prestación de formularios HTML, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos. Otros componentes, como Zend_Auth y Zend_Acl, proveen autentificación de usuarios y autorización diferentes a las tiendas de certificados comunes. También existen componentes que implementan bibliotecas de cliente para acceder de forma sencilla a los servicios web más populares.

El principal patrocinador del proyecto Zend Framework es Zend Technologies, pero muchas empresas han contribuido con componentes o características importantes para el marco. Empresas como Google, Microsoft y StrikeIron se han asociado con Zend para proporcionar interfaces de servicios web y otras tecnologías que desean poner a disposición de los desarrolladores de Zend Framework". (Zend Technologies Inc., 2005)

1.3.5 Herramientas CASE

Una herramienta CASE no es más que un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software, tanto en el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores. Estas herramientas son utilizadas principalmente para aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y dinero.

Estas, cuando son bien utilizadas se puede mejorar la productividad en el desarrollo, mantenimiento y calidad del software, reducir el tiempo y costo de desarrollo y mantenimiento de los sistemas informáticos, mejorar la planificación de un proyecto, aumentar la biblioteca de conocimiento informático de una

[10] **MVC** es un patrón de diseño de software dirigido a su arquitectura, en el cual todo el proceso está dividido en 3 capas. Típicamente estas capas son el Modelo, la Vista y el Controlador.

empresa ayudando a la búsqueda de soluciones para los requisitos, automatizar el desarrollo del producto, la documentación, la generación de código, las pruebas de errores y la gestión del proyecto, ayuda a la reutilización del software, portabilidad y estandarización de la documentación, gestión global en todas las fases de desarrollo de software con una misma herramienta y facilitar el uso de las distintas metodologías propias de la ingeniería del software.

Visual Paradigm

Visual Paradigm es una herramienta CASE profesional, de prototipo visual usada para el modelado UML, que puede ser utilizada por muchos usuarios, tales como ingenieros de software, analistas de sistemas y del negocio, arquitectos y desarrolladores. Este programa está orientado a la creación de diseños usando el paradigma de programación orientado a objetos. Además, incluye una herramienta llamada Visual Architect que permite la generación de código para el manejo de la base de datos y puede generar códigos para lenguajes como PHP, JAVA y C# y gestores de base de datos PostgreSql, SQL, MySQL, entre otros.

1.3.6 Navegador Mozilla Firefox

Mozilla Firefox es un navegador de código abierto, multiplataforma, basado en el código de base de Mozilla que proporciona una navegación más rápida, segura, y eficiente que otros navegadores. Entre sus principales características se encuentran:

- Velocidad: las páginas se abren en menor tiempo y se navega con comodidad en ellas.
- Seguridad: es un software de código abierto, esto permite que las fallas de seguridad se corrijan al instante, tienen un grupo de colaboradores encargados de esta cuestión. Al mismo tiempo, Mozilla Firefox usa su propio motor de dibujado de páginas, Gecko, lo que lo hace inmune a las fallas de seguridad del Internet Explorer, que también tienen todos los navegadores basados en sí mismo.
- Desde sus inicios de modo sencillo y eficaz Mozilla Firefox procuró evitar las molestas ventanas emergentes de publicidad.
- La navegación por pestañas permite tener varias páginas abiertas en una misma ventana, lo que garantiza una navegación mucho más cómoda.
- Se pueden usar varios perfiles de usuario con configuraciones diferentes para cada uno.

Este navegador consta además de un innumerable conjunto de extensiones y temas dentro de su paquete de complementos, que le permiten modificar la versión original instalada y personalizarla más al usuario, así como brindarle mayores comodidades. En el desarrollo de aplicaciones web es muy utilizado el complemento firebug, a través del cual los desarrolladores pueden llevar un control del tráfico de información desde el cliente hasta el servidor, brindando grandes facilidades a la hora de conectar la programación de la capa de presentación con la del negocio.

1.3.7 Controlador de versiones TortoiseSVN

TortoiseSVN es una herramienta muy fácil de usar para la revisión y el control de versiones, así como fuente de control de software para Windows. Se basa en Subversion, para el cual provee una interfaz de usuario amigable y sencilla. Es desarrollado bajo licencia GPL, lo cual significa que es completamente libre, incluyendo su código fuente, por lo que puede ser utilizado libremente para el desarrollo de aplicaciones comerciales o de uso particular. Desde que dejó de estar integrado a los IDE de desarrollo como Visual Estudio, Eclipse y otros, puede ser utilizado con cualquier herramienta de desarrollo.

Como cliente de Subversion tiene todas sus características, tales como:

- La mayoría de las características de CVS[11] actual.
- Los directorios, renombrados y archivos de metadatos son todos versionados.
- La opción de Commits es realmente atómica.
- Manejo eficiente de archivos binarios.

1.4 Aplicaciones Web

Son un tipo de aplicación en la cual los usuarios por medio de un navegador realizan peticiones a una aplicación remota accesible a través de Internet (o a través de una intranet) y que recibe una respuesta que se muestra en el propio navegador. Estas se codifican en un lenguaje (HTML, JavaScript, Java, asp.net, php, etc.) soportado por los navegadores web en la que se confía la ejecución al navegador. Son conocidas por las aplicaciones como cliente ligero y también por la facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software a miles de usuarios potenciales. Se puede decir también que una página web es importante porque puede contener elementos que permiten una comunicación

[11] **CVS** es una aplicación informática que implementa un sistema de control de versiones: mantiene el registro de todo el trabajo y los cambios en los ficheros controlados por esta.

activa entre el usuario y la información. Esto permite que el usuario acceda a los datos de modo interactivo, gracias a que la página responderá a cada una de sus acciones, como por ejemplo rellenar y enviar formularios, participar en juegos diversos y acceder a gestores de bases de datos de todo tipo.

1.5 Bases de datos relacionales

Las bases de datos relacionales se basan en el modelo relacional, cuya estructura principal es la relación, es decir, una tabla bidimensional compuesta por filas y columnas. Una base de datos relacional es un tipo de base de datos en donde toda la información visible al usuario está organizada estrictamente como tablas de valores. Todas sus operaciones se realizan sobre estas tablas. Estas bases de datos son percibidas por los usuarios como una colección de relaciones normalizadas de diversos grados que varían con el tiempo. Entre las ventajas de las bases de datos relacionales se encuentran:

- Garantizar herramientas que eviten la duplicidad de registros, a través de campos clave o llaves.
- Garantizar la integridad referencial. Si se excluye un registro se eliminan todos los registros relacionados dependientes.
- Favorecer la normalización por ser más comprensibles y aplicables.

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD (Berkeley Software Distribution) para Bases de Datos y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para lograr la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

Sus características técnicas la hacen una de las bases de datos más potentes y robustas del mercado. Su desarrollo comenzó hace más de 15 años y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema.

Ventajas

"Características: PostgreSQL tiene muchas de las funcionalidades de las bases de datos comerciales, y algunas que ni siquiera éstas tienen.

Funcionamiento: PostgreSQL se ejecuta en dos modos. El modo normal fsync, que guarda en el disco toda transacción completada, es más lento que la mayoría de las bases de datos comerciales, en parte porque son pocas las que hacen esto. El modo no-fsync, que es menos fiable que el anterior, es más rápido que las bases de datos comerciales, aunque en este modo un choque del sistema puede significar la pérdida de alguna transacción.

Fiabilidad: los desarrolladores de PostgreSQL se preocupan de producir versiones que han estado al menos un mes en fase de prueba beta, estables y con el menor número de errores posible. Además, una comunidad enorme de usuarios y desarrolladores se preocupan de revisar constantemente el código y publicar nuevas mejoras.

Precio: PostgreSQL es gratuito, tanto para fines no comerciales como para fines comerciales. Lo mejor es que su futuro no está sujeto a las decisiones de ninguna empresa.

Soporte: Existe abundante documentación, listas de correo, acceso directo a los desarrolladores, empresas que ofrecen consultoría y, lo más importante, el código fuente". (Martinez, 2009)

Trabajo con los catálogos

Los catálogos del PostgreSQL no son más que el lugar en que la base de datos relacional almacena todos los metadatos sobre sus esquemas, tales como información sobre sus tablas y columnas. Técnicamente, los catálogos son tablas normales con una descripción completa de la estructura de la base de datos y sus restricciones que permite conocer los datos existentes sin acceder a ellos directamente. Los nombres de todos los catálogos comienzan por la combinación "pg_". A continuación se ilustran todos los catálogos del PostgreSQL según (PostgreSQL Global Development Group , 1996)

Dentro de estos, algunos de los más utilizados son:

Tabla 1. Catálogos de Postgres más utilizados.

Nombre del Catálogos	Descripción
pg_database	base de datos en el sistema
pg_class	contiene las tablas, índices (primarios), secuencias, vistas ("relaciones")
pg_attribute	atributos de cada "clase" que aparece en pg_class
pg_index	índices secundarios
pg_proc	procedimientos existentes para programar funciones, etc.
pg_type	tipos de datos usados: base y compuesto

pg_operator	operadores (+-*/<>=~!@#%^&|'?)-> y nuevos tipos!!!
pg_aggregate	Almacena "funciones de agregación". Son funciones que operan sobre un conjunto de valores, por ejemplo: sum, count y max.
pg_am	qué "método de acceso por índices" existen
pg_authid	contiene información sobre identificadores de autorización de acceso a la base de datos (roles).

1.6 Framework de Persistencia de Objetos Relacionales

1.6.1 Persistencia

"Es la capacidad del programador para conseguir que sus datos sobrevivan a la ejecución del proceso que los creó, de forma que puedan ser reutilizados en otro proceso. Cada objeto, independiente de su tipo, debería poder llegar a ser persistente sin traducción explícita. También debería ser implícito que el usuario no tuviera que mover o copiar los datos expresamente para ser persistentes". (Castillo, 2009)

El concepto de persistencia está relacionado con el almacenamiento secundario de instancias de objetos. Un objeto se dice persistente cuando es almacenado en un archivo u otro medio permanente. Un programa puede grabar objetos persistentes y luego recuperarlos en un tiempo posterior.

La persistencia de datos es sin duda alguna, una de las partes más importantes en el desarrollo de un software. En caso de que la aplicación esté basada en programación orientada a objetos, la persistencia se logra mediante la serialización de los objetos o su almacenando en una base de datos. La segunda opción es muy utilizada, pero también mucho más compleja, puesto que la mayoría de las bases de datos siguen un modelo relacional y este difiere en gran medida del modelo objetos.

1.6.2 Framework de persistencia

Es una estructura de software orientada a mejorar la productividad a través de la reutilización de código. Es un conjunto de clases creadas para apoyar la escritura de código en un contexto determinado. *"Un framework de persistencia es, por lo tanto, una librería de clases que facilita la tarea del programador al permitirle guardar objetos en bases de datos relacionales de manera lógica y eficiente, de otra manera tocaría hacerlo manualmente, y este es, potencialmente, un proceso tedioso, repetitivo y propenso a errores".* (Cadavid, 2008)

1.6.3 Framework de persistencia más usados en la UCI

Hibernate

Dentro de los framework de persistencias más usados en la Universidad de Ciencias Informáticas se encuentran como se mencionó anteriormente, el Hibernate que es una herramienta para el lenguaje Java que facilita el mapeo de atributos entre una base de datos relacional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones. Parte de una filosofía de mapear objetos Java "normales", también conocidos en la comunidad como "POJOs" (viejos objetos de Java, por sus siglas en inglés). Proporciona un potente lenguaje de consultas HQL (Lenguaje de Consultas Hibernate): subconsultas, uniones externas, ordenamientos, proyección (consultas de reportes), paginación, etc. Lenguaje intermedio que según la base de datos que se utilice y el dialecto especificado será traducido al SQL (Lenguaje de Consultas Estructurado) dependiente de cada base de datos de forma automática y transparente.

Propel

Es un ORM para PHP que facilita la labor de desarrollo de aplicaciones web, gracias a la capa que transforma el tratamiento de la base de datos mediante objetos, con la que se puede recuperar, insertar y modificar datos. No es necesario preocuparse por las conexiones de la base de datos y escribir SQL. Tampoco es necesario escapar datos o realizar conversiones. Tan solo es necesario definir la base de datos en formato XML u obtener la definición desde una base de datos ya existente. Provee un soporte básico para implementar herencia orientada a objetos (subclases). Hay varias opciones de implementación para mapear clases entidades y subclases para las tablas de las bases de datos. Propel usa la mayoría de modelos eficientes desde SQL y perspectivas óptimas de consulta: una tabla es usada para todas las subclases.

1.7 Framework de persistencia de objetos relacionales Doctrine

En la actualidad la mayoría de las bases de datos existentes siguen una estructura relacional, básicamente esto se traduce en un conjunto de tablas relacionadas compuestas por valores escalares. Sin embargo, la programación orientada a objetos (POO), muy difundida en los días de hoy, devino en su desarrollo posterior al surgimiento de la estructura relacional. Esto conlleva a que no sea factible el manejo de la estructura de esas bases de datos siguiendo una lógica de negocio orientada a objetos y sea

necesario convertir este conjunto de tablas y relaciones en clases y objetos válidos para la programación. Como solución a dicha problemática surgen los ORM.

1.7.1 Doctrine

"Doctrine no es más que un ORM para la persistencia de datos que trabaja con lenguaje de programación PHP 5.2.3 o superior, situándose sobre su poderosa capa de abstracción de la base de datos (PHP DBAL). Es uno de los más conocidos que interactúa con este lenguaje y su utilización brinda grandes facilidades a los desarrolladores a la hora de su trabajo con los datos, automatizando la generación de ficheros que responden a las tablas relacionales de la base de datos". (Sitio Oficial de Doctrine)

1.7.2 Principales características de Doctrine

Doctrine se divide en dos capas fundamentales que interactúan en conjunto, la DBAL del inglés Database Abstraction Layer o Capa de Abstracción de la Base de Datos y la compuesta por el ORM reflejadas en la siguiente imagen.

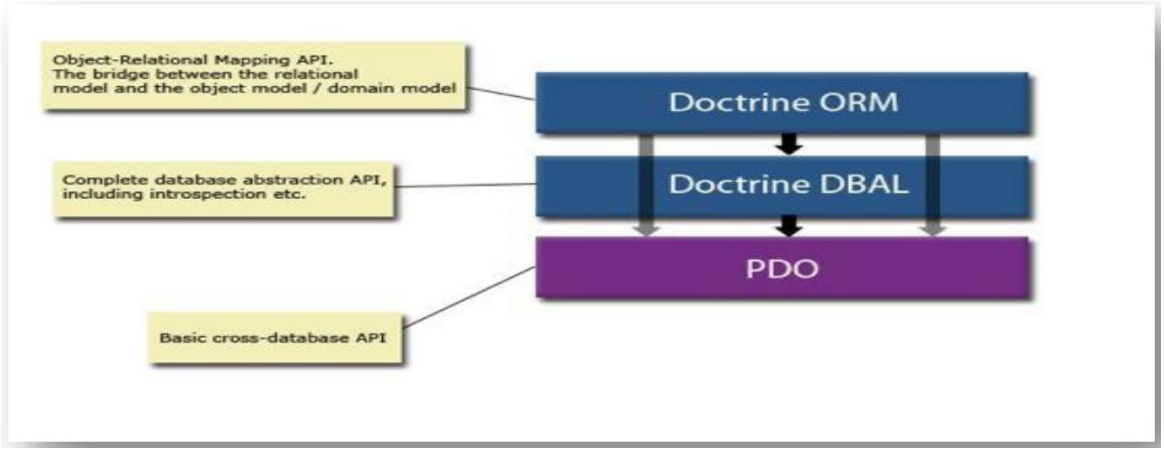

Figura 1. Estructura lógica del framework de persistencia Doctrine. (SENCIOLABS, 2009)

Doctrine es un framework desarrollado básicamente sobre los principios de los patrones de Data Mapping (Mapeo de Datos), Meta Data Mapping (Mapeo de Metadatos) y ActiveRecord (Registro Activo). Mediante una clase base nombrada Doctrine_Record, todas sus clases hijas asumen la interfaz común de ActiveRecord (permite salvar, eliminar, actualizar, etc.) facilitándole a Doctrine un sencillo uso y monitoreo

del ciclo de vida de sus registros. Sin embargo, el trabajo de mapeo y creación de ficheros es mayormente reenviado a otros componentes como por ejemplo la clase Doctrine_Table.

Este framework tiene la facultad de construir consultas a la base de datos relacional utilizando un lenguaje OO (orientado a objetos) denominado DQL (Doctrine Query Language o Lenguale de Consultas Doctrine) inspirado en el potente HQL (Hibernate Query Language) de Hibernate. Además, implementa un patrón CRUD (Crear, Recuperar, Actualizar y Eliminar) para operaciones comunes, ya sea desde crear un nuevo registro o actualizar los registros existentes. Automatiza la generación de clases PHP del modelo basado en YAML, y de sus respectivas relaciones desde una base de datos existente. Soporta varios motores de bases de datos e incluye dos tipos de herencia: Simple, para cuando todas las clases tienen las mismas columnas, y de agregación, que almacena un valor adicional en la tabla y permite la instanciación automática del tipo de modelo correcto cuando se realiza una consulta.

Doctrine es capaz de aplicar varios comportamientos a los modelos, por ejemplo para un modelo Timestampable automáticamente se crearán 2 columnas: created_at y updated_at, las cuales guardarán la fecha de un registro creado y la de un registro actualizado respectivamente. Además, soporta opciones como Data fixtures (Instalación de Datos) que son utilizados para cargar datos de prueba a través de los modelos para poblar la base de datos con estos y realizarle pruebas, migraciones que permiten tener actualizadas las nuevas versiones de las bases de datos, cache mediante de un servidor memcache (Memoria cache) que facilita la rapidez y eficiencia de su trabajo, eventos con una flexible arquitectura de listeners (escuchadores) que no solo posibilita estar a la escucha de los posibles eventos que ocurran sino también alterar la ejecución de un determinado método, paginación e interfaz de línea de comando.

Dentro de los ORM que dispone PHP uno de los más populares es Doctrine. A pesar de lo novedoso que se torna todo el tema de estos framework de persistencia, aun en la actualidad, se dispone de una amigable documentación acerca de su librería, elemento indispensable a la hora evaluar la usabilidad de cualquier técnica o herramienta. Sin embargo, para la comunidad de desarrollo en español, todavía la documentación es escasa, lo cual de cierta manera resulta una limitante.

1.8 Metodología de desarrollo de software

Conjunto formado por procedimientos, herramientas y técnicas con una documentación determinada que permiten estructurar, planificar y controlar el proceso de desarrollo de software. Básicamente constituye la

guía a seguir por los desarrolladores a la hora de implementar un sistema determinado y genera como resultado final al cliente, el software y el soporte documental correspondiente a este.

La selección de una metodología para el desarrollo de un determinado sistema está condicionada por varios factores que definen la más apropiada, entre ellos se encuentran, el tipo de sistema a desarrollar, el personal participante, los recursos disponibles y las necesidades del cliente, por lo que no existe una metodología estándar a seguir, más bien se necesita que este proceso sea adaptable y configurable de acuerdo al proyecto. Existen dos tipos fundamentales de metodologías para tratar este tema, las que siguen los métodos tradicionales, que son generalmente para software de gran envergadura, y las que proponen procesos ágiles para el desarrollo, más usadas en software pequeños y con marcadas diferencias entre sí. Para este caso se centra el estudio de las metodologías ágiles dado el tipo de sistema en cuestión, y poniendo en práctica el modelo de desarrollo utilizado en el proyecto, descrito en el trabajo de diploma de los Ingenieros en Ciencias Informáticas Sergio Hernández Cisneros y Mileidy Magalys Sarduy Pérez. Dicho modelo en resumen, plantea lo siguiente:

"En aras de mejorar la organización, control y rendimiento en el proceso de desarrollo de software tecnológico en la Departamento de Tecnología del Centro de Informatización de la Gestión de Entidades (CEIGE), se ha decidido organizar los procesos que deben desarrollarse en la misma.

Con esta finalidad y luego de un estudio detallado de los modelos de desarrollo de software y estructuras organizativas existentes y basándose en las buenas prácticas de las metodologías ágiles para lograr rapidez y productividad en la construcción de tecnología, se realiza una propuesta de modelo que incluye en una primera parte, los procesos que deben llevarse a cabo y posteriormente basado en dicho modelo la propuesta de estructura organizativa con que debe contar la Subdirección Técnica para que funcione eficientemente y como un todo. Se tienen en cuenta además, los procesos horizontales que deben desarrollarse en la subdirección y que van unidos al proceso productivo como es el caso de la formación académica, la gestión de capital humano, la calidad de los componentes desarrollados y la investigación.

(Hernández Cisneros, y otros, 2009)

1.9 Herramientas, tecnologías y modelo de desarrollo a utilizar

El desarrollo de este sistema estará guiado y dirigido principalmente por el modelo de desarrollo descrito en el trabajo de diploma de los Ingenieros en Ciencias Informáticas Sergio Hernández Cisneros y Mileidy Magalys Sarduy Pérez, basado en el estudio de metodologías ágiles y donde se recogen todos los

artefactos a generar que necesita el proyecto para el desarrollo de un software. Se utilizará el PHP como lenguaje de programación orientada a objetos en la implementación de la lógica del negocio apoyándose en el Zend Framework (framework de PHP seleccionado) y en el IDE de desarrollo Zend Studio. Las vistas serán diseñadas siguiendo los estándares de diseño web propuestos por el proyecto y bajo el uso de las librerías creadas por la UCID (Centro de Desarrollo de Software de las FAR en colaboración con la UCI) y ExtJS de JavaScript, en el IDE de desarrollo Aptana, brindando siempre al usuario un ambiente amigable y de gran usabilidad. Para apoyar el desarrollo de su documentación y modelado se utilizará el Visual Paradigm como potente herramienta CASE. Como control de versiones tanto de la documentación como del código de la solución se utilizará el cliente TortoiseSVN del servidor de Subversion creado en el proyecto. El Mozilla Firefox será el navegador utilizado, ya que cuenta con los complementos necesarios para que el desarrollador facilite su trabajo de implementación. De manera general, son estas las herramientas y tecnologías a utilizar, que son propuestas por la dirección del proyecto ERP para el desarrollo de aplicaciones web, que además de ser muy robustas, eficientes, constituyen en su mayoría aplicaciones de software libres. La solución permitirá resolver el problema del mapeo y persistencia de datos en el proyecto ERP, corrigiendo algunas deficiencias presentadas por el antiguo Mapeador Doctrine Generator.

1.10 Soluciones existentes

Luego de un profundo análisis e investigación a escala global es fácil determinar que hasta el momento el único sistema conocido que existe, capaz de automatizar por completo la generación de ficheros de mapeo a partir de bases de datos relacionales y siguiendo los principios del ORM Doctrine, es el Doctrine Generator, implementado en el proyecto ERP y utilizado actualmente por sus desarrolladores. Es válido señalar que el mundo del software es un proceso cambiante y que está centrando sus cambios en la producción de aplicaciones web, por las capacidades de accesibilidad y distribución que brinda. El proyecto ERP se encuentra trabajando en el diseño e implementación de aplicaciones de este tipo y bajo la utilización de código abierto, más sin embargo, se ve obligado a utilizar la aplicación de desktop Doctrine Generator que además de estar desarrollada bajo la utilización de software propietario (.NET), se aleja de las políticas y características planteadas en los estándares seguidos por el proyecto ERP. Estas necesidades fueron suficientes para centrar los esfuerzos de los desarrolladores del proyecto en la

creación de una nueva versión de dicha aplicación, pero esta vez ajustándose a las políticas definidas por el proyecto.

1.11 Resultados esperados

Se propone para la solución del problema existente, la implementación de una aplicación web que cumpla los estándares y normativas que sigue el proyecto ERP, bajo el uso de tecnologías de código abierto y que sea capaz de solucionar las necesidades, en este caso, de los desarrolladores del ERP, con la posibilidad de ser accesible desde cualquier parte del dominio donde se utilice. Una aplicación que permita, en concreto, acceder a una base de datos relacional seleccionada por el usuario y tomar de esta las tablas, con sus columnas y relaciones, para entonces mapearlas y convertir su contenido en ficheros de código escrito en lenguaje PHP en forma de clases con atributos y objetos. Permite la gestión de las relaciones entre estas clases al igual que las relaciones establecidas entre las tablas de la base de datos, además de agregarles consultas y ver su contenido. Dado el caso de que será una aplicación web, deberá permitir al cliente acceder de alguna forma a los ficheros generados, ya sea descargándolos en su PC u otra vía con el objetivo de que posteriormente trabaje con estos. Se contará con las tecnologías y herramientas seleccionadas y con el modelo de desarrollo a seguir para obtener los mejores resultados.

1.12 Conclusiones parciales

Durante este capítulo se consolidó el basamento teórico con vistas al desarrollo del software mediante un estudio del estado del arte de las principales herramientas y tecnologías que propone el proyecto para crear aplicaciones web. Se profundiza principalmente en el análisis del Doctrine como ORM a utilizar y de la herramienta existente hasta el momento que automatiza el proceso de mapeo, para concluir con los resultados que se esperan alcanzar al final del trabajo. Se define el modelo de desarrollo a seguir, el cual describe cada artefacto a obtener en cada etapa del desarrollo de esta aplicación y se sientan las bases que dan pie a su inicio.

CAPÍTULO 2: DESARROLLO DE LA SOLUCIÓN

2.1 Introducción

En el siguiente capítulo se parte de la fundamentación de la solución propuesta y mediante la aplicación de las dimensiones de desarrollo propuestas en el modelo de desarrollo seleccionado, se describen los

requerimientos funcionales y los escenarios que dan pie a la descripción de los principales procesos del sistema.

Como parte del diseño de la solución se desarrollan los prototipos de interfaz de usuario, junto con la representación de los diagramas de clases y su estructura física en la implementación, así como las vistas lógicas de los procesos, que constituyen los artefactos principales generados para esta etapa.

De la etapa de implementación se muestra una descripción de las clases del sistema, acompañada de las nomenclaturas definidas para los comentariados y para cada tipo de clase, sus variables y funciones. Se define cómo serán tratados los errores para las interfaces y el negocio. Se representa además una vista de despliegue que muestra las necesidades de hardware para la utilización del software.

2.2 Fundamentación de la solución propuesta.

La versión 2.0 en desarrollo del Doctrine Generator viene a concertar algunos detalles funcionales y cambios de tecnologías empleadas en su creación, luego del previo estudio de la versión anterior y con el fin de ajustar dicho software a las necesidades de los desarrolladores del proyecto ERP. Al igual que su antecesor, se basará en la generación de ficheros de mapeo a partir del uso del ORM Doctrine, con el objetivo de automatizar dicho proceso y reducir este trabajo a funcionalidades sencillas y fáciles de ejecutar por los desarrolladores en un entorno amigable y eficaz. El uso de PHP en el desarrollo de la lógica del negocio y del ExtJS en la capa de presentación evidenciará que la nueva solución se convertirá en una importante herramienta dentro del marco de trabajo del ERP, dado que se ajustará a todas las políticas de implementación y desarrollo seguidas por el proyecto.

Doctrine Generator 2.0 se convertirá ya en una aplicación web que se integrará al paquete de herramientas existentes en el proyecto (PACSOFT) y brindará de esta forma servicios a un gran número de usuarios simultáneamente. Por tanto, para acceder a ella será preciso disponer de las condiciones necesarias para utilizar el marco de trabajo sobre el cual funcionaría. A pesar de asumir a los desarrolladores del ERP como sus principales clientes, no será preciso tener grandes conocimientos de programación o de base de datos para trabajar con la herramienta, pues se tratará de forma muy sencilla cada acción con el fin de ejecutar estos procesos de manera inherente a los usuarios.

Al acceder a esta aplicación se mostrará una interfaz principal, de la que se partiría a ejecutar todas las funcionalidades disponibles, aunque sólo será posible, partiendo de esta vista, crear un nuevo proyecto para luego disponer de las demás opciones. El crear un nuevo proyecto se traducirá en realizar una serie

de pasos que permiten establecer todas las condiciones necesarias para generar los ficheros mapeados a partir de la base de datos seleccionada, como por ejemplo, definir el nombre de dicho proyecto, establecer la conexión a la base de datos seleccionada y elegir en esta el esquema y las tablas que se van a mapear. Justo al finalizar se creará un directorio con una estructura de carpetas peculiar en la que se colocarán los ficheros. La raíz del directorio contendrá el nombre del proyecto y su estructura interna quedaría de la siguiente forma:

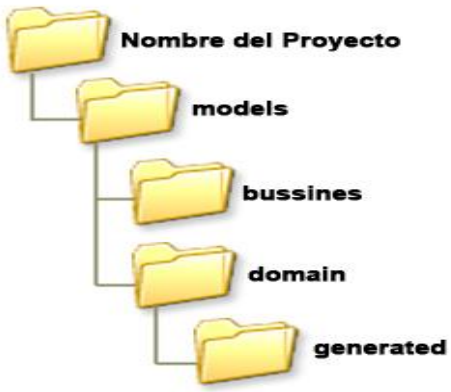

Figura 2. Estructura de carpetas de los ficheros generados.

Dentro de la carpeta que dará nombre al proyecto se encuentra la carpeta models, compuesta por la bussines, domain y generated donde se guardarán los ficheros (*.php) que nombran las tablas mapeadas, tomando el nombre de la tabla con el sufijo Model en el primer caso, el nombre de la tabla para el segundo y el prefijo Base seguido por el nombre de la tabla en el tercero respectivamente.

Una vez terminada esta operación se mostrará en la interfaz principal las tablas del esquema seleccionado e información sobre estas, dando la opción al usuario de editar las relaciones de dependencia existentes para agregar herencias y relaciones de muchos a muchos N-M. A partir de entonces será posible tanto exportar el proyecto como agregarle consultas a los ficheros generados. Las consultas serán fácilmente creadas sólo dando clic y estableciendo los argumentos que tendrá para posteriormente asignársela a alguna clase creada, dando la facilidad también de ver las que ya existen o simplemente la clase en la que se desea agregar alguna mediante un visor de código que mostrará íntegramente el fichero que se seleccione. Por su parte, la acción de exportar el proyecto creará

automáticamente un compactado con todos los archivos que fueron generados y los descargará en la PC del cliente para que pueda utilizarlos.

De manera general esta versión resolverá la persistencia de datos del proyecto completo y las particularidades que la conformarán han sido descritas a grandes rasgos. En los epígrafes posteriores se abordarán todos los temas referentes al desarrollo de los procesos de análisis, diseño e implementación de la herramienta y las actividades que son objetivos en la obtención de esta solución.

2.4 Descripción de los Procesos del Sistema.

Proceso Crear un nuevo proyecto

La figura 3 muestra el diagrama que identifica el proceso Crear un nuevo proyecto, partiendo de un punto inicial en que el usuario entra el nombre de un proyecto y el sistema es capaz de validarlo y crear el directorio del nuevo proyecto a partir de este nombre, posteriormente el usuario introduce los datos para la conexión a la base de datos, estos son verificados por el sistema y a partir de ellos se crea el fichero de conexión correspondiente y se cargan los esquemas de la base de datos, luego se entran los atributos de configuración para el mapeo, estos se validan y conforman el fichero de configuración para así finalizar este proceso.

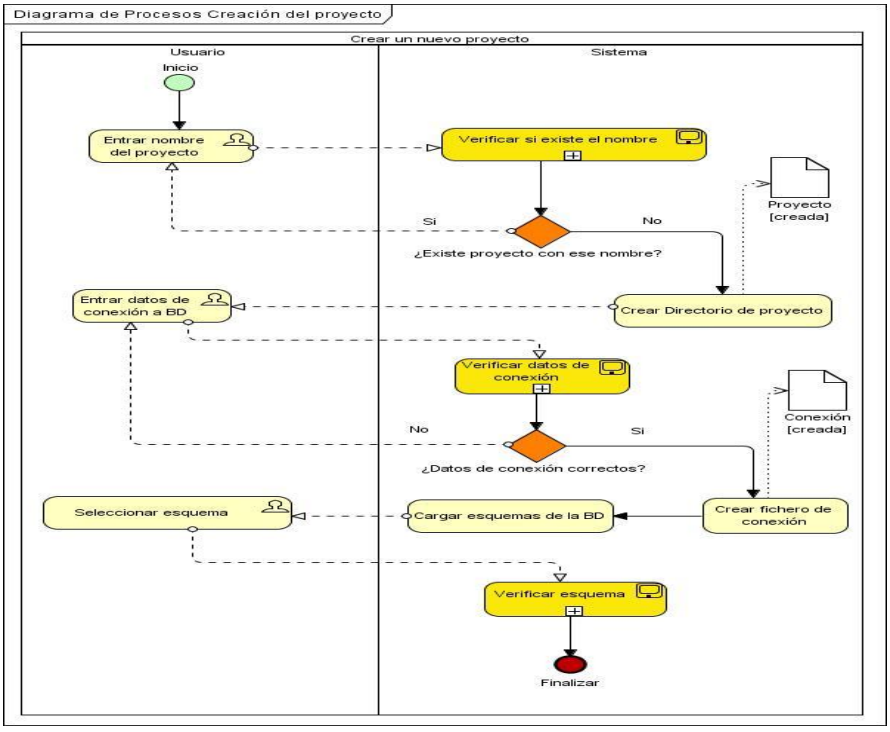

Figura 3. Diagrama de Proceso Crear un nuevo proyecto.

Proceso Construir Consulta

Para realizar el proceso construir consulta el usuario debe seleccionar una tabla, luego si desea ver el código puede seleccionar la opción que le corresponde a esa funcionalidad, para construir la consulta tiene que entrar los datos de la misma, siempre que todos estos datos estén correctos el sistema construye la consulta asignándola a la tabla correspondiente.

Proceso Generar ficheros

El proceso encargado de Generar ficheros de mapeo se inicia con la selección por el usuario de las tablas que se desean mapear y la verificación por el sistema de que la selección es correcta. Luego se generan los ficheros del dominio en su directorio, seguido de los ficheros base y por último los ficheros Model del negocio, para entonces concluir el proceso.

Proceso Gestionar relaciones

Para dar inicio al proceso de Gestionar las relaciones entre las tablas mapeadas, el usuario selecciona el tipo de relación a gestionar, ya sea herencia o muchos a muchos, seguidamente procede a entrar lo datos

de la nueva relación, estos son validados por el sistema y luego es agregada la relación a la tabla mapeada que le corresponde, para de esta forma dar fin al proceso.

Proceso Descargar proyecto

Para realizar el proceso descargar proyecto el usuario solicita la opción que corresponde a esta función, el sistema verifica que exista algún proyecto creado, si no hay creado ningún proyecto se lanza un error, pero si no hay inconvenientes se crea un compactado y luego es descargado al directorio que se seleccionó.

2.4 Análisis de la solución

2.4.1 Descripción de los requerimientos funcionales

Los requisitos funcionales que se asumieron para este sistema garantizan las necesidades de los clientes y constituyen las premisas básicas e inviolables que debe cumplir el sistema en desarrollo. Los requisitos funcionales que se definieron son:

- ✓ Crear un nuevo proyecto.
- ✓ Conectar a la Base de Datos.
- ✓ Construir Consulta.
- ✓ Vista previa.
- ✓ Generar Modelos.
- ✓ Gestionar Relaciones.
- ✓ Descargar Proyecto.

A continuación se muestra la especificación del requisito crear un nuevo proyecto, reflejado en la plantilla utilizada por el proyecto ERP-Cuba para este caso.

Especificación de requisito Crear un nuevo proyecto.

Descripción textual del requisito

Tabla 2. Descripción del requisito Crear un nuevo proyecto.

Precondiciones	N/A
Flujo de eventos	
Flujo básico	
1	Se introduce el nombre del proyecto.
2	Se crea el proyecto.
3	Concluye el requisito.
Pos-condiciones	

1	Se crea un directorio donde se guardará el nuevo proyecto.
Flujos alternativos	
Flujo alternativo a Campos en blanco	
El sistema muestra un mensaje de error.	
Volver al paso 1 del flujo básico.	
Concluye el requisito.	
Pos-condiciones	
1	No se crea el directorio.
Flujo alternativo b Directorio erróneo	
1	El sistema determina que el directorio no es correcto.
2	El usuario define un nuevo directorio.
3	Volver al paso 1 del flujo básico.
Pos-condiciones	
Volver al paso 1 del flujo básico.	
Flujo alternativo c El usuario cancela la acción	
1	Concluye el requisito.
Pos-condiciones	
1	No se establece la conexión.
Pos-condiciones	
1	No se crea el proyecto.
Validaciones	
1	Se validan los datos según lo establecido en el Modelo conceptual CSG-ERP-N-SEG-i2201.
Conceptos	**Nombre proyecto** Visibles en la interfaz: Nombre del proyecto
Asuntos pendientes	Posibles mejoras al requisito.

2.4.2 Descripción de los escenarios.

Con el objetivo de planificar, organizar y dirigir el proceso de desarrollo de este software se partió de un estudio y análisis de los requisitos funcionales asociados al sistema y luego de definirlos como principios inviolables a cumplir, fue fácil detectar los principales escenarios que garantizan la más correcta y eficaz implementación del software. La relación de estos escenarios a los requisitos y la descripción de cada uno de ellos constituyen un factor fundamental en dicho proceso de desarrollo y contribuyen a medir la asequibilidad y ajuste de la herramienta en cuestión a las necesidades planteadas por el proyecto. A continuación se relacionan y describen en su total integridad.

Tabla 3. Descripción de los escenarios.

Escenario	Requisitos Asociados	Descripción
1. **Crear un nuevo proyecto**	Crear un nuevo proyecto Conectar a BD	Es el escenario que permite crear un nuevo proyecto. Para ello, es necesario conectarse a la base de datos seleccionada y así obtener los esquemas que tiene esta.
2. **Construir Consultas**	Construir consultas Visor de código	Es el escenario encargado de gestionar dentro del proyecto todo el trabajo con las consultas que se deseen agregar a los ficheros de las tablas mapeadas, así como la posibilidad de visualizarlas.
3. **Generar Modelos**	Generar Modelos	Es el escenario encargado de gestionar dentro del proyecto la generación de los modelos a partir de las tablas a mapear.
4. **Gestionar relaciones**	Gestionar Relaciones	Escenario que garantiza gestionar las relaciones entre las tablas mapeadas, o sea, crear, eliminar, y editar sus relaciones de dependencia.
5. **Descargar Proyecto**	Descargar Proyecto	Escenario encargado de gestionar la compresión y descarga del proyecto creado en la PC del cliente.

2.5 Diseño de la solución.

2.5.1 Desarrollo de prototipos de interfaz de usuario.

En apoyo al mejor entendimiento y desarrollo de la implementación de esta herramienta se comienza a definir y diseñar las principales vistas que van a interactuar con el usuario final, en busca de aminorar la complejidad y abstracción que puede conllevar la implementación de un software para los desarrolladores y también con el objetivo de evaluar el nivel de satisfacción del cliente y garantizar el entendimiento con este. Estas vistas parten de las necesidades que dan lugar a la creación del software y casi nunca en un principio se ajustan a todos los requisitos que se proponen, es sólo durante su desarrollo que se alcanzan los prototipos finales de los que va a disponer el sistema. Esta herramienta constará de 8 vistas o interfaces fundamentales, en las que se gestionarán todas las funcionalidades que tendrá, descritas en el epígrafe que describe la solución propuesta. A continuación se muestra el prototipo desarrollado para la interfaz principal (figura 4).

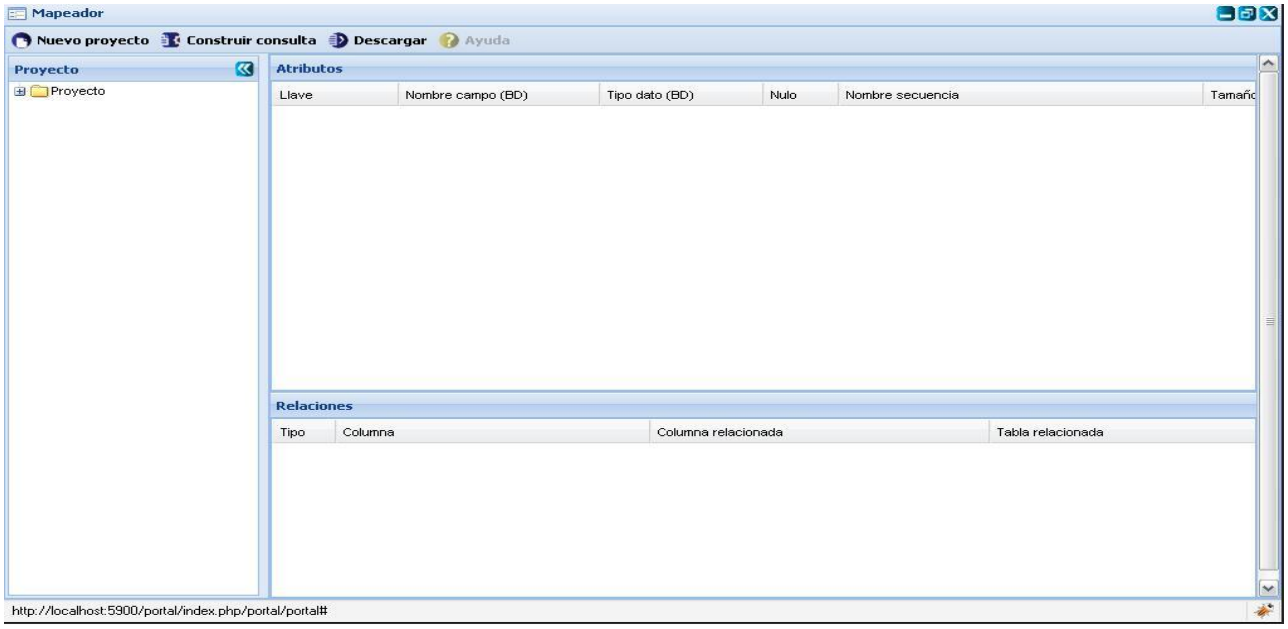

Figura 4. Interfaz principal del sistema.

2.5.2 Diagrama de Clases del sistema.

Diagrama de Clases de Diseño con Estereotipos Web

Este diagrama de modelamiento tiene como objetivo representar la estructura que sigue una aplicación web basándose en los principales componentes que la integran y su relación entre sí. En la figura 5 se describen las interfaces de usuario con sus formularios correspondientes en relación con la página servidora que controla todas las funcionalidades que se realizan, y el conjunto de las principales clases que tributan a la existencia de dichas funcionalidades.

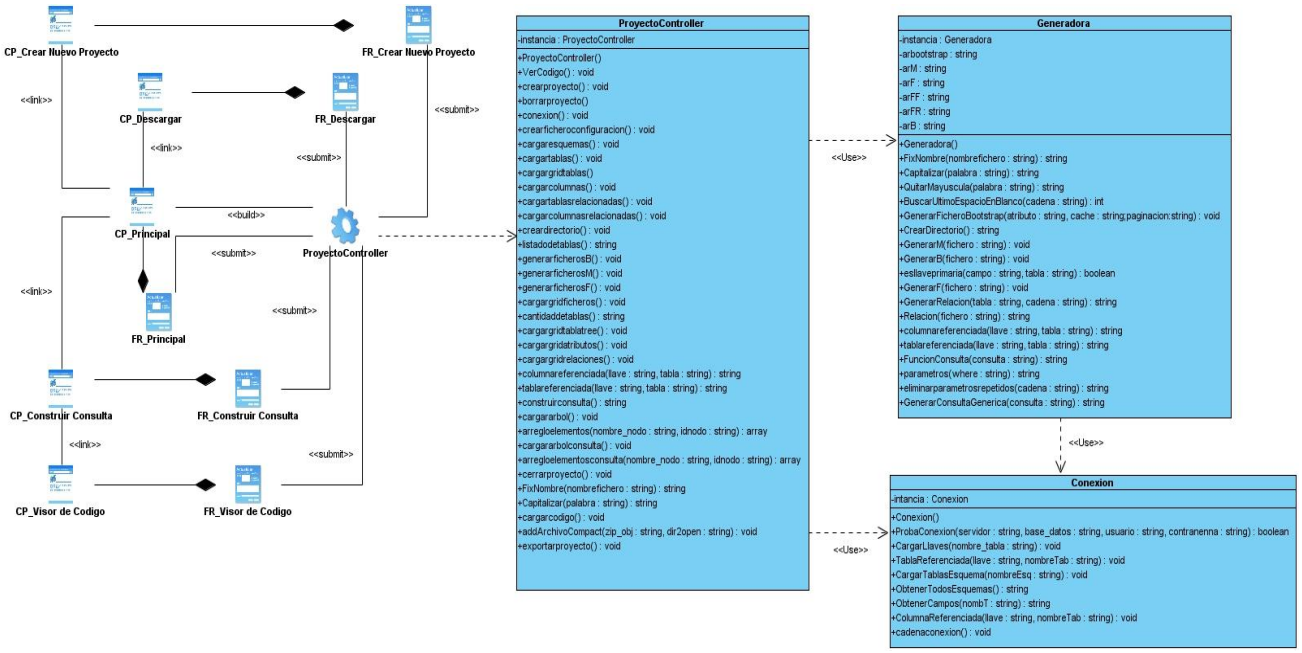

Figura 5. Diagrama de clases de diseño con estereotipos web.

2.5.3 Estructura física de las clases.

En el diagrama de la figura 6 se representan la estructura lógica en que se encuentra organizada la capa del negocio del software, así como las relaciones que existen entre las clases que los componen. Su objetivo principal es de manera general, agrupar los paquetes en que se encuentra dividido el software tanto lógica como físicamente.

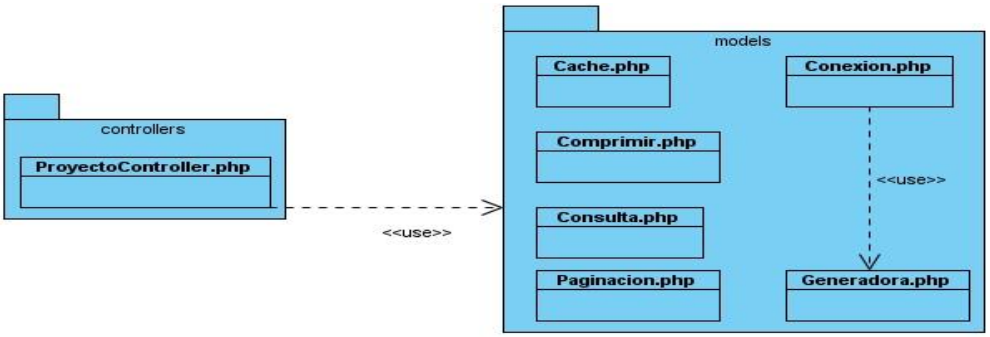

Figura 6. Diagrama de paquetes.

2.5.4 Vista lógica de los procesos

Diagrama de Secuencia

"Un diagrama de secuencia es una forma de diagrama de interacción que muestra los objetos como líneas de vida a lo largo de la página y con sus interacciones en el tiempo representadas como mensajes dibujados como flechas desde la línea de vida origen hasta la línea de vida destino. Los diagramas de secuencia son buenos para mostrar qué objetos se comunican con qué otros objetos y qué mensajes disparan esas comunicaciones. Los diagramas de secuencia no están pensados para mostrar lógicas de procedimientos complejos". (Scribd, 2005)

A continuación se muestra el diagrama de secuencia correspondiente al escenario Crear un nuevo proyecto donde se refleja las interacciones del usuario con las partes del sistema encargadas de realizar esta funcionalidad. En este caso el usuario representa a los desarrolladores que van a utilizar la herramienta y dentro de las partes del sistema se encuentran la interfaz de crear proyecto, la clase controladora que gestiona esta acción y las entidades contenedoras de la información que lo constituye.

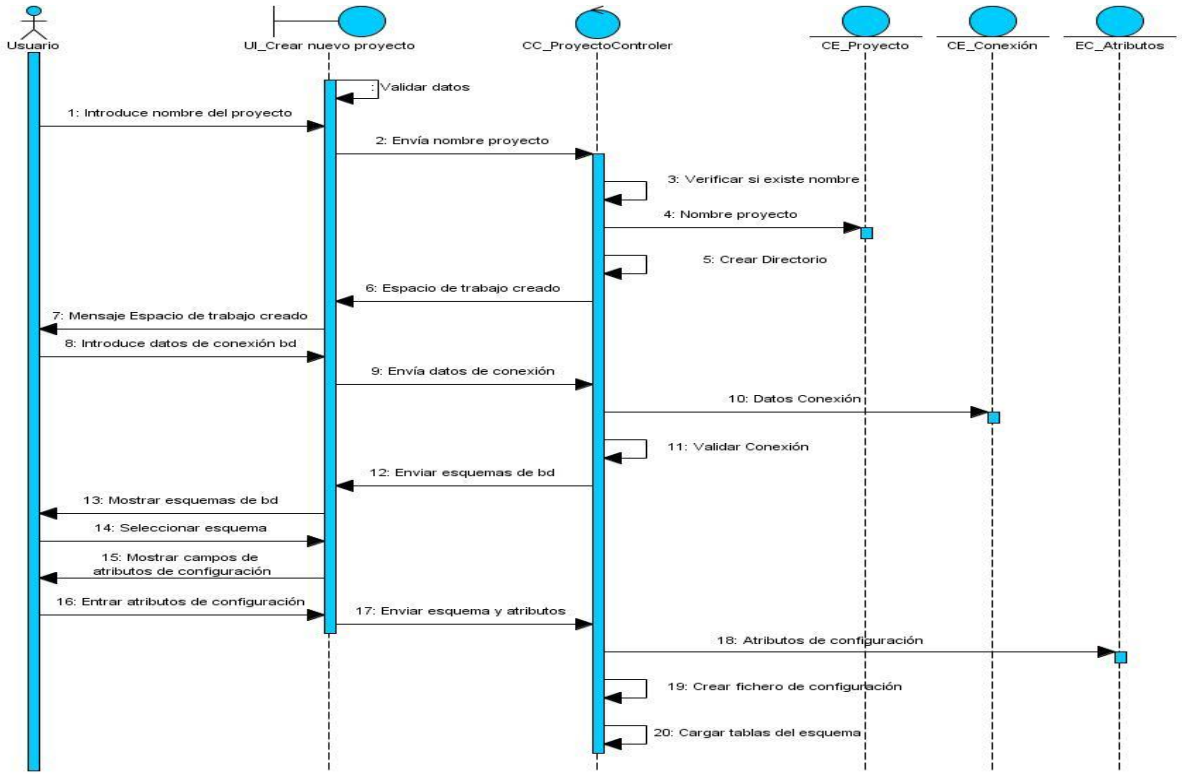

Figura 7. Diagrama de secuencia Crear un nuevo proyecto.

Diagramas de Colaboración

"Los diagramas de interacción que hacen hincapié en la estructura que tienen los objetos participantes en las actividades de interacción, son conocidos como diagramas de colaboración. Estos describen la representación principal de los escenarios arquitectónicos que tenga un sistema, ya que representan y esquematizan las colaboraciones entre los objetos de cada uno de ellos. Como todos los diagramas de interacción (…) constituyen uno de los artefactos más importantes que se generan en el análisis y en el diseño orientado a objetos. El tiempo y el esfuerzo dedicados a su preparación deberían absorber un porcentaje considerable de la actividad total destinada al proyecto. Para mejorar la calidad de su diseño, es posible aplicar patrones, principios y expresiones codificados". (Larman, 2003)

En la figura 8 se muestra el diagrama de colaboración correspondiente a la creación de un nuevo proyecto donde se evidencia la interacción, comunicación, colaboración e intercambio de mensajes que existe entre el cliente con el sistema, en este caso, con la interfaz para crear un nuevo proyecto y la clase controladora nombrada ProyectoControler, encargada de gestionar el cumplimiento de esta funcionalidad.

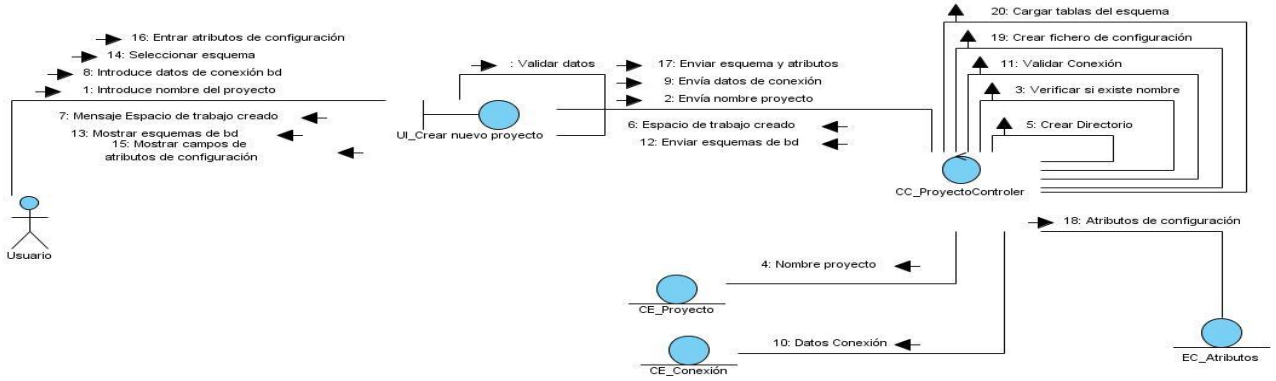

Figura 8. Diagrama de Colaboración de Crear un nuevo proyecto.

2.5.5 Aplicación de patrones.

Implementación de la lógica del negocio

Una vez definidas las interfaces candidatas a la solución final es muy importante la implementación continua de la aplicación por parte de todos los desarrolladores integrados. Su trabajo en conjunto puede brindar muchas facilidades a la hora de integrar las diferentes partes del software y lograr un mejor

entendimiento entre el personal encargado de la lógica de presentación para con los de la lógica del negocio. Luego de disponer de las clases entidades y la controladora del sistema representadas en los diagramas y modelos mostrados anteriormente, y con una clara concepción de los resultados que se esperan alcanzar, se procede a su implementación haciendo uso de los patrones y estándares que permitan obtener un código de alta calidad y usabilidad. En este caso se sigue un estilo Modelo Vista Controlador y se aplica la creación de sólo una instancia para cada clase en todo el sistema (Singleton), características que serán explicadas a continuación.

Patrón Singleton

El Singleton es un patrón de diseño, específicamente de creación. Utilizado para garantizar que solo una clase tenga una instancia y proporcione un punto de acceso global a ella. Su funcionamiento puede resumirse a que oculta el constructor de la clase Singleton, para que los clientes no puedan crear instancias. Declara en esta clase una variable miembro privada que contenga la referencia a la instancia única que se desea gestionar y provee una función o propiedad que brinda acceso a la única instancia gestionada por el patrón.

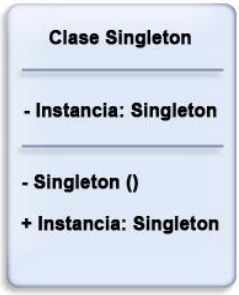

Figura 9. Patrón Singleton.

En el caso de este software el patrón Singleton se utiliza en todas las clases dentro del paquete Model (Cache.php, Comprimir.php, Conexion.php, Consulta.php, Generadora.php y Paginacion.php) haciéndolas actuar como instancias únicas en todos los casos para así poder utilizar cada una de ellas en la clase controladora (ProyectoController.php). En la siguiente figura se muestra un ejemplo de código donde se implementa dicho patrón.

```
15      private static $instancia;
16      public static function getInstance() {
17          if (self::$instancia == null) {
18              self::$instancia = new self ( );
19          }
20          return self::$instancia;
21      }
```

Figura 10. Ejemplo de implementación de Patrón Singleton.

Modelos Vista Controlador

El Modelo Vista Controlador (MVC) es un patrón arquitectónico que tiene como principal característica dividir una aplicación en tres módulos identificables y con funcionalidades bien definidas: el Modelo, las Vistas y el Controlador.

"El modelo es un conjunto de clases que representan la información del mundo real que el sistema debe procesar" (Deacon, 2010)

"Las vistas son el conjunto de clases que se encargan de mostrar al usuario la información contenida en el modelo. Una vista está asociada a un modelo, pudiendo existir varias vistas asociadas al mismo modelo. Una vista obtiene del modelo solamente la información que necesita para desplegar y se actualiza cada vez que el modelo del dominio cambia por medio de notificaciones generadas por el modelo de la aplicación". (Bergin, 2010)

"El controlador es un objeto que se encarga de dirigir el flujo del control de la aplicación debido a mensajes externos, como datos introducidos por el usuario u opciones del menú seleccionadas por él. A partir de estos mensajes, el controlador se encarga de modificar el modelo o de abrir y cerrar vistas. El controlador tiene acceso al modelo y a las vistas, pero las vistas y el modelo no conocen de la existencia del controlador". (Burbeck, 2009).

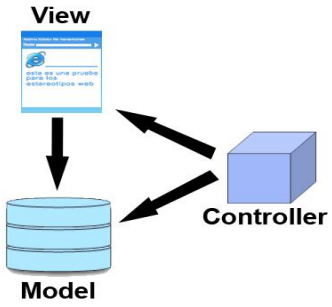

Figura 11. Patrón Modelo Vista-Controlador

Este patrón se utiliza para definir la arquitectura que tendrá el sistema, que constituye una potente guía a seguir en el desarrollo de aplicaciones web y además, por estar implementado dentro de la arquitectura del framework seleccionado para la implementación del negocio, en este caso el Zend Framework de PHP. Esta aplicación se ajustará estrictamente a la descripción de cada una de sus partes, para garantizar el mejor aprovechamiento del patrón y a la vez obtener una herramienta de incuestionable calidad y

usabilidad en el mundo del software actual. Se definen en la parte de las vistas, las 8 fundamentales con las que contará la aplicación, mencionadas en la descripción de la solución propuesta, dentro de los modelos estarán un conjunto de clases que manejarán toda la información necesaria (Cache.php, Comprimir.php, Conexion.php, Consulta.php, Generadora.php y Paginacion.php) que se mostrará en las vistas y serán controladas por una única clase controladora (ProyectoController.php). La estructura que propone este patrón para la lógica del negocio del sistema se ve evidenciada en el diagrama que representa la estructura física de las clases, anteriormente descrita.

2.6 Implementación de la solución.

Clases Controladoras

Se definió para las clases controladoras que después del nombre se les colocará como sufijo la palabra:"Controller". *Ejemplo:* ProyectoController.php

Clases de los modelos

Business (Negocio)

Se definió para las clases que se encuentran dentro de Business que su nombre será tal y como se escriba sin sufijos ni prefijos. *Ejemplo:* Proyecto

Domain (Dominio) y Bases del Dominio (Generated)

En este caso particular no se utilizaron clases del dominio (**Domain**) ni de bases del dominio (**Generated**) puesto que esta es una aplicación sencilla y que no necesita ni siquiera de una base de datos para la persistencia de información, sino que trabaja sobre bases de datos de otros proyectos.

Nomenclatura de las funciones.

Se definió que el nombre de las funciones se escribe con la primera palabra en minúscula, para el caso específico de que sea un nombre compuesto se empleará la notación *CamelCasing*[12], y de forma tal que al leerlo se conozca el objetivo de la misma. *Ejemplo:* eliminarFile(). En caso particular que la función sea una acción de la clase controladora u otra se escribe después del nombre de la función la palabra "Action" como sufijo. *Ejemplo: crearproyectoAction().*

[12] **CamelCase** es un estilo de escritura que se aplica a frases o palabras compuestas. El nombre CamelCase se podría traducir como Mayúsculas/Minúsculas Camello, aunque no es correcto en todos los contextos ya que la palabra inglesa Case no tiene traducción literal. El nombre se debe a que las mayúsculas a lo largo de una palabra en CamelCase se asemejan a las jorobas de un camello.

Nomenclatura de las variables

Se definió que el nombre a emplear para las variables se escribe en minúscula y en caso de ser un nombre compuesto se escriben ambas palabras en minúscula separadas por un guión bajo.

2.6.1 Normas de los comentariados

En pos de garantizar un código más legible y reutilizable se convierte en una necesidad el comentariado de todo el código que se implemente dentro del desarrollo, con el objetivo de aumentar su mantenibilidad a lo largo del tiempo.

Nomenclatura de los comentariados

Se definió el uso de comentarios claros y precisos que ayudaran a la mejor comprensión del código implementado para que se entiendan sus objetivos específicos.

En las clases

Para implementar una clase se escribe una pequeña descripción donde se expliquen de forma general los propósitos de la misma, así como su autor y sistema al que pertenece. Dicha descripción se escribe de la siguiente forma:

/**
* Nombre de la clase *
* Descripcion *
* @author *
* @package *(módulo)
* @subpackage *(sub módulo)
* @copyright *
* @version (versión - parche)
*/

Quedando en la clase controladora de este sistema de la forma:

```
/**
 * ProyectoController *
 * Descripción: Esta clase maneja todas las funcionalidades que brinda el sistema,
 * mediante su interacción con todas las clases de los modelos.
 * Es la encargada de controlar la logica de este negocio.
 * @author Maikel Y. Sañudo Clemente
 * @package Mapeador
 * @copyright ERP
 * @version 2.0
 */
```

Figura 12. Ejemplo de comentariado en las clases

En las funciones

Antes de la declaración de cada función se establece una pequeña descripción donde se explica el propósito de la misma, su autor, los parámetros que maneja y los valores de retorno. Dicha declaración quedaría como sigue a continuación:

/**

* Nombre de la función *

* Descripcion *

* @author * (en el caso que no sea el mismo autor de la clase)

* @param *(los parámetros que recibe la función con su descripción)

* @throws *(en caso de que lance excepciones)

* @return *(representa los valores de retorno)

**/

Quedando para las funciones de este sistema de la forma:

```
/**
 * Nombre de la función: crearproyectoAction
 * Descripción: Esta función se encarga de crear el directorio con
 * el nombre del proyecto entrado por el usuario.
 * @author Leandro L. Rojas Cuesta
 * @param nombre_proyecto
 * @throws 'Ya existe un proyecto con ese nombre.'
 * @return 'Espacio de trabajo del proyecto creado satisfactoriamente.'
 **/
```

Figura 13. Ejemplo de comentariado en las funciones.

Comentarios por líneas

Se utilizan también los comentarios para líneas de código en específico que ayudan a aclarar la complejidad de algunos algoritmos. Para ellos se utiliza // y auto seguido el comentario que describe la línea de código, tal y como se representa en el siguiente esquema:

```
719  else
720  {
721    $this->addArchivoCompact($zip_obj,$dir2open.'/'.$file); // Ciclo recursivo para añadir un archivo al compactado.
722  }
```

Figura 14. Ejemplo de comentariado por línea de código.

Tratamiento de errores

Los errores se manejan de forma sencilla, tanto en las interfaces de usuario como en la lógica del negocio, para certificar una respuesta a los comportamientos no esperados del sistema o la mala operatividad del usuario.

En el caso de las interfaces se utilizan expresiones regulares en los campos de entrada de textos para garantizar datos correctos y los errores son manejados mediante sentencias de condición if, else y la muestra de mensajes al usuario como se muestra en el siguiente ejemplo de código:

```
184                handler:function(){
185                                    if (descargar)
186                                      {
187                                           ConsultaDQL();
188                                           rootNode1.setText(tree.root.text);
189                                      }
190                                      else
191                                          {
192                                          mostrarMensaje(3,"No se ha creado un proyecto para descargar.");
193                                          }
194                                    }
195            });
```

Figura 15. Tratamiento de errores en las interfaces.

Por su parte, los errores en la lógica de negocio validan nuevamente los datos que reciben de las interfaces y además chequean que el resultado de una función determinada sea el esperado, apoyándose también en las sentencias de condición if, else y el envío de mensajes de error a las interfaces para ser mostradas a los usuarios y el tratamiento del error. Los posibles errores del negocio de este sistema quedan resueltos según se muestra en la figura:

```
40      if (file_exists($dir.'/'.$nombre_proyecto))
41      {
42          if (file_exists($dir.'/'.$nombre_proyecto.'/.bootstrap.php'))
43          {
44              echo ("{codMsg:'3',result:'Ya existe un proyecto con ese nombre.'}");
45          }
46          else
47          {
48              $this->eliminarFile($dir.'/'.$nombre_proyecto);
49              mkdir ( $dir.'/'.$nombre_proyecto, 0755 );
50              echo ("{codMsg:'1',result:'Espacio de trabajo del proyecto creado satisfactoriamente.'}");
51          }
52      }
```

Figura 16. Tratamiento de errores en el negocio.

2.6.2 Uso de Doctrine en la solución.

Para el desarrollo de esta aplicación fue necesario llevar a cabo un estudio profundo sobre el framework de persistencia Doctrine, con el objetivo de conocer sus particularidades y potencialidades a tener en cuenta a la hora de utilizarlo. Centrando dicho estudio en el manual de Doctrine 1.1 se definió las principales funcionalidades a utilizar de este ORM y que constituirán novedades agregadas en la solución propuesta. Las principales características de Doctrine asumidas en la implementación de esta herramienta y el lugar en que se utilizaron dentro de la aplicación se relacionan de esta manera:

Introducción a las Conexiones

Esta función se utilizó para establecer las conexiones a las Bases de Datos y con el fin de obtener los resultados de todas las consultas a los catálogos del PostgreSQL para acceder a sus esquemas, tablas, columnas y relaciones poniendo en práctica las facilidades que brinda Doctrine en este caso.

Lenguaje de Consultas Doctrine

Doctrine Query Language (DQL), su lenguaje de consulta de objetos, se aplicó para ayudar a los usuarios en la recuperación de objetos complejos de las bases de datos. Esta opción siempre debe ser considerada cuando se busca la recuperación de datos relacionales de manera eficiente (por ejemplo, cuando se obtengan los usuarios y sus teléfonos). Este lenguaje posee un sinnúmero de expresiones que garantizan el intercambio de cualquier información por difícil que sea, con la base de datos. A continuación se relacionan solo los ejemplos más utilizados:

1. **SELECT:** para la recuperación de los datos de uno o más componentes. Cada expresión de selección indica una columna o un valor que se desea obtener. Debe haber al menos una expresión de selección (select_expr) en cada instrucción SELECT.
2. **FROM:** cláusula que indica el componente o componentes del que se van recuperar los datos.

3. **WHERE:** cláusula utilizada para indicar la condición o condiciones que los registros deben satisfacer para ser seleccionados. Where_definition es una expresión que se evalúa a verdadero para cada fila a ser seleccionada. La declaración selecciona todas las filas si no hay cláusula WHERE.

Las funcionalidades de Doctrine empleadas son insignificantes ante la gran variedad de opciones que brinda su utilización, por lo que es posible que en nuevas soluciones sean incluidas muchas otras que en este caso han sido omitidas. Sin embargo, la aplicación de las antes mencionadas fue suficiente para satisfacer las necesidades propuestas para el desarrollo de este software.

2.6.3 Vista de despliegue del sistema.

Un diagrama de despliegue es un tipo de diagrama UML con el cual queda representado un modelado del hardware utilizado en la elaboración del software. En la figura 17 se describen los dispositivos necesarios para la utilización del producto, es decir, desde dónde se encuentra instalado, dónde se puede utilizar, hasta dónde se conecta para tomar los datos. Es una vista panorámica que describe los requisitos de hardware y software mínimos, necesarios para que esta herramienta funcione.

Figura 17. Diagrama de Despliegue.

2.7 Pre requisitos o condiciones de uso.

Para poder utilizar esta herramienta será necesario instalar el marco de trabajo del proyecto ERP en un servidor y publicarlo, de forma tal que el usuario pueda acceder a este por la red desde su PC cliente. El usuario debe disponer de un navegador web, preferentemente el Mozilla Firefox y conocer los datos necesarios para la conexión a un servidor de base de datos local o remoto del cual mapeará la información.

Por su parte, para poder instalar el marco de trabajo y publicarlo, será necesario disponer de un servidor web Wamp Server con acceso a un servidor de base de datos de PostgresSQL versión 8.3.2 y el

navegador Mozilla Firefox, así como los manuales necesarios para la configuración de todas estas herramientas.

Solo cuando se cumplan estas condiciones podrá disponerse de la herramienta desarrollada en este trabajo.

2.8 Conclusiones parciales.

La aplicación de la metodología seleccionada, en el capítulo anterior, refleja el análisis, diseño e implementación de la solución desarrollada en este trabajo. Con el fin de resolver los problemas que presentaba el Doctrine Generator se le dio cumplimiento a los nuevos requerimientos funcionales definidos. Se desarrollaron todos los artefactos necesarios en cada etapa, lo cual brinda una descripción de cada uno de ellos, con el objetivo de apoyar, junto a la descripción de los distintos tipos de clases definidas y la nomenclatura de sus variables y atributos, la implementación por parte de los desarrolladores y conformar la documentación del software para el proyecto. Para lograr la reutilización y optimización del código se planteó el uso de patrones arquitectónicos y de diseño, se detalló la forma y el lugar en que fueron utilizados dentro de la solución, en conjunto con la aplicación de las nomenclaturas definidas para el comentariado.

De esta forma se obtuvo una aplicación que plantea diferencias con respecto a la versión anterior en cuanto a la utilización de Doctrine y de las nuevas tecnologías, que va a facilitar el trabajo de los desarrolladores con los datos a persistir.

CAPÍTULO 3: VALIDACIÓN Y PRUEBA DE LA SOLUCIÓN

3.1 Introducción

En el presente capítulo se muestran y aplican algunas métricas utilizadas actualmente para validar la calidad en el diseño de software y se definen cuáles se aplicaron al diseño de la solución propuesta. Estas validaciones proporcionan una medida de cuánto puede ser la calidad del producto desde la visión interna que proporcionan los parámetros que se definen en este, ayudando a comprender todo el proceso técnico que se utiliza para desarrollar un sistema. También se describen las pruebas de aceptación realizadas al software y los resultados que le corresponden a estas, con el objetivo de garantizar el cumplimiento de los requisitos del sistema propuestos.

3.2 Validación de la solución

Métricas de software

En su libro "Ingeniería del software, un enfoque práctico", Pressman plantea:

"El proceso del software y las métricas del producto son una medida cuantitativa que permite a la gente del software tener una visión profunda de la eficacia del proceso del software y de los proyectos que dirigen utilizando el proceso como un marco de trabajo. Se reúnen los datos básicos de calidad y productividad. Estos datos son entonces analizados, comparados con promedios anteriores, y evaluados para determinar las mejoras en la calidad y productividad. Las métricas son también utilizadas para señalar áreas con problemas de manera que se puedan desarrollar los remedios y mejorar el proceso del software". (Pressman, 1997)

Dentro de los principales atributos de calidad que se incluyen en la aplicación de las métricas, se encuentran:

- ***Responsabilidad:*** se le asigna a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- ***Complejidad de implementación:*** grado de dificultad en la implementación de un diseño de clases determinado.
- ***Reutilización:*** nivel de reutilización que tiene una clase o estructura de clase, dentro de un diseño de software determinado.

- ***Acoplamiento:*** valor de dependencia de una clase o estructura de clase con otras. Este atributo está muy ligado al de Reutilización.
- ***Complejidad del mantenimiento:*** categoría de esfuerzo para realizar un arreglo, mejora o rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- ***Cantidad de pruebas:*** número de esfuerzos para realizar las pruebas de calidad (Unidad) del producto (componente, clase, conjunto de clases, etc.) diseñado.

Las clases constituyen la unidad básica y fundamental de un sistema orientado a objetos. Por tanto, y dado el caso de que este software se realizó bajo la programación orientada a objetos (POO), es indiscutible que la validación del mismo se centró en la aplicación de métricas dirigidas a sus clases de forma individual, sus jerarquías y colaboraciones, haciendo uso de los atributos de calidad descritos anteriormente. Dichas métricas se encuentran desarrolladas a continuación:

Tamaño operacional de clase (TOC): está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

Tabla 4.Tamaño operacional de clase (TOC).

Atributo de calidad	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

Tabla 5.Rango de valores para los criterios de evaluación de la métrica Tamaño Operacional de Clase (TOC).

Atributo	Categoría	Criterio
Responsabilidad	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Complejidad implementación	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
	Baja	>2*Promedio

Reutilización	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

3.2.1 Resultados obtenidos al aplicar la métrica de Tamaño Operacional de Clase (TOC)

Al aplicar la métrica TOC se determinó que la aplicación consta de 9 clases y 115 procedimientos, reflejados en la siguiente tabla junto a los atributos de calidad de Responsabilidad, Complejidad de Implementación, Reutilización. Se obtuvo asimismo, un promedio de procedimientos por clase de 13, y los siguientes datos sirvieron para categorizar los atributos por cada clase:

Promedio de Procedimientos por clase= 13.

Responsabilidad: Baja <=13, 13< Media< 26, Alta> 26.

Complejidad: Baja <=13, 13< Media < 26, Alta> 26.

Reutilización: Baja> 26 ,13< Media < 26, Alta<=13.

Tabla 6. Resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad de Implementación	Reutilización
Atributo	18	Media	Media	Media
Cache	4	Baja	Baja	Alta
Comprimir	6	Baja	Baja	Alta
Conexion	9	Baja	Baja	Alta
Consulta	10	Baja	Baja	Alta
Generadora	19	Media	Media	Media
Paginacion	6	Baja	Baja	Alta
Proyecto	9	Baja	Baja	Alta
ProyectoController	34	Alta	Alta	Baja

De acuerdo con estos resultados se construyeron las siguientes gráficas para un mejor entendimiento de los valores alcanzados por los atributos evaluados:

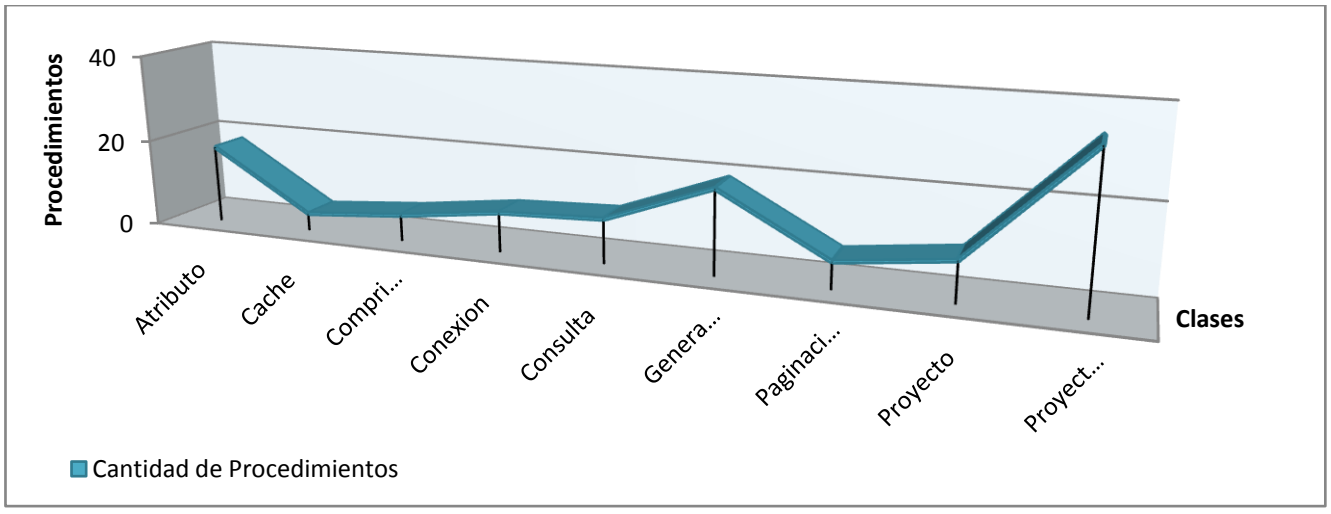

Figura 18. Gráfica que representa los resultados obtenidos después de aplicar la métrica TOC (relación de la cantidad de procedimientos por clases).

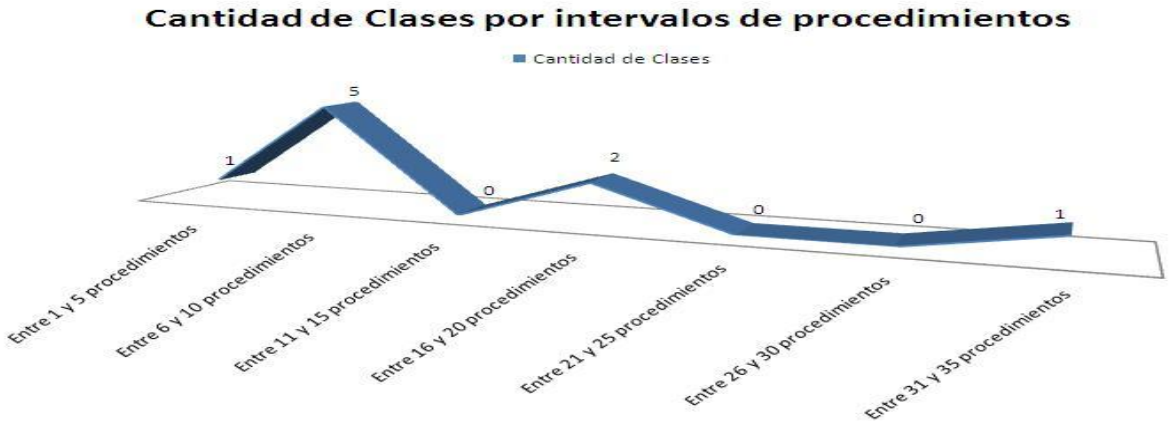

Figura 19. Gráfica que representa la cantidad de clases por intervalos de procedimientos.

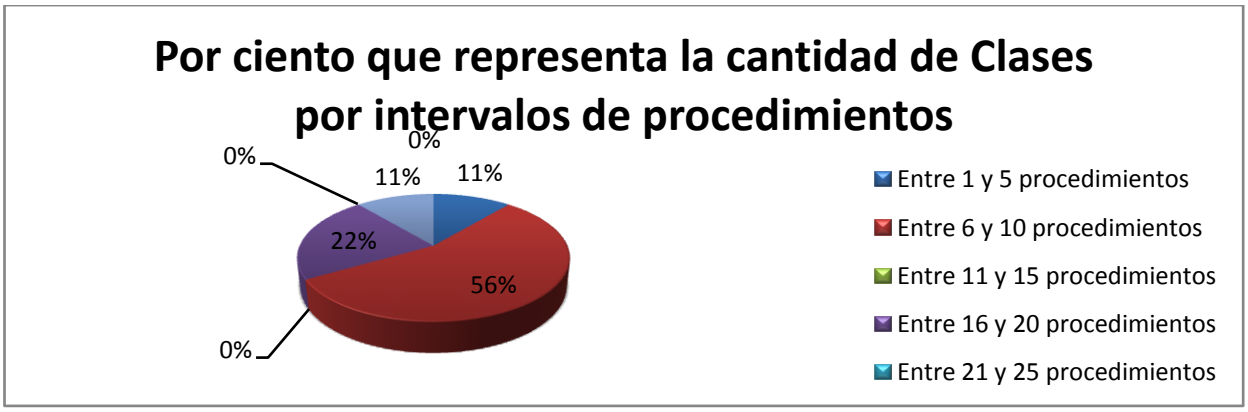

Figura 20. Gráfica que representa los % de la cantidad de clases por intervalos de procedimientos.

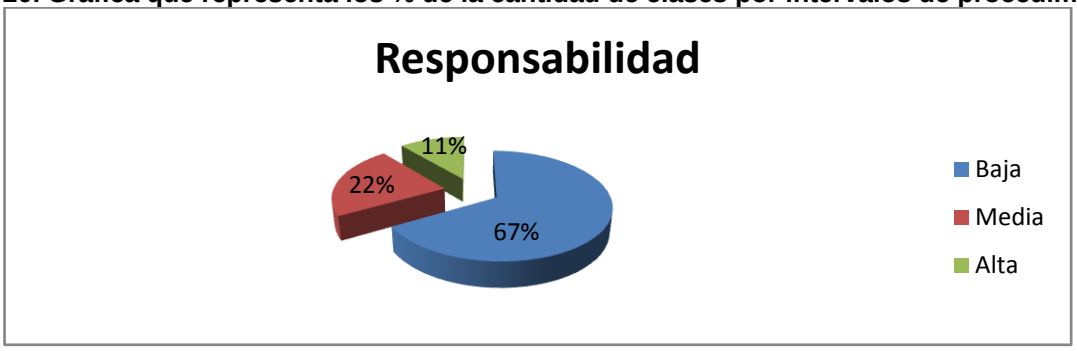

Figura 21. Gráfica que representa los % de clases por categorías del atributo de calidad Responsabilidad obtenidos en la aplicación de la métrica TOC.

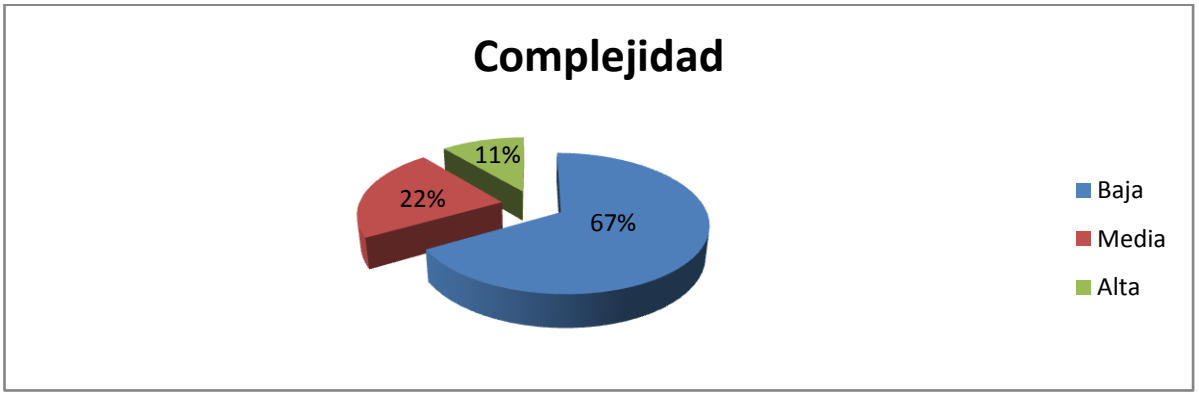

Figura 22. Gráfica que representa los % de clases por categorías del atributo de calidad Complejidad de implementación obtenidos en la aplicación de la métrica TOC.

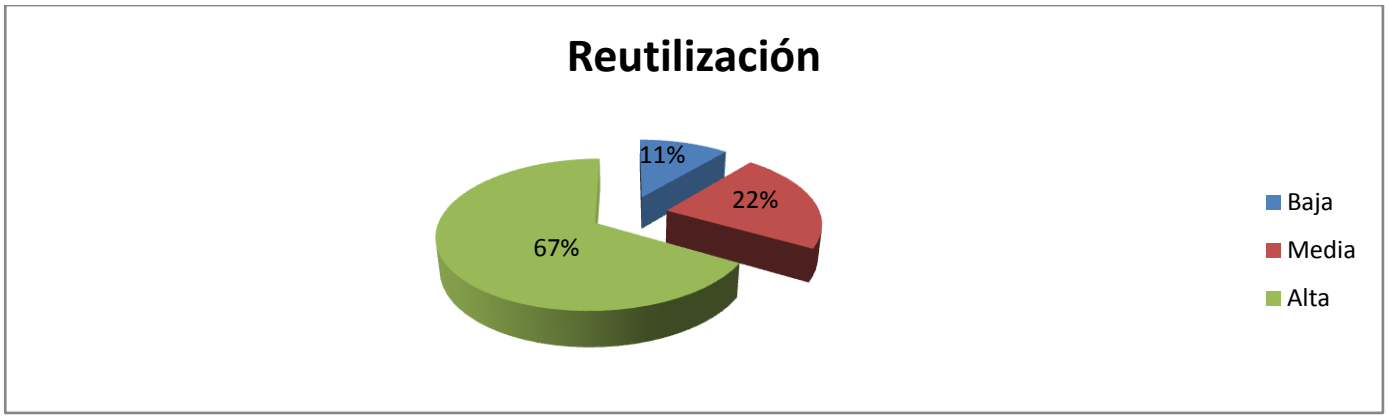

Figura 23. Gráfica que representa los % de clases por categorías del atributo de calidad Reutilización de implementación obtenidos en la aplicación de la métrica TOC.

Tras un análisis de los resultados arrojados por la evaluación del sistema bajo los instrumentos de medición de la métrica TOC, se demuestra que los valores idóneos para cada uno de los atributos de calidad evaluados se alcanzaron, puesto que, como se puede observar, el 67% de las clases del software contienen un número menor que el promedio de procedimientos para una clase, lo cual influye positivamente en el hecho de que predomine una responsabilidad baja de las clases en un 67%, y hace que carezcan de mucha complejidad, por lo que se tornan mucho más reutilizables. Solamente un 11% de las clases tienen pocas posibilidades de reutilización y una gran complejidad de implementación. Estos resultados demuestran la solidez del diseño de la implementación de este sistema, y garantizan que este trabaje de forma rápida y eficiente, permitiendo incluso la reutilización de gran parte del diseño y la futura inserción de nuevas funcionalidades.

Relaciones entre clases (RC): Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

Tabla 7. Tabla Tamaño operacional de clase (TOC).

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.
Complejidad de mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

Tabla 8. Rango de valores para los criterios de evaluación de la métrica Tamaño Operacional de Clase (TOC).

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de mantenimiento	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio
Cantidad de pruebas	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio

3.2.2 Resultados obtenidos al aplicar la métrica de Relaciones entre Clases (RC)

Al aplicar la métrica RC se determinó que las 9 clases existentes constan de 18 dependencias entre ellas, reflejadas en la siguiente tabla junto a los atributos de calidad de Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas. Se obtuvo un promedio de relaciones de dependencia por clase de 2, y los siguientes datos sirvieron para categorizar los atributos por cada clase:

Promedio Cantidad de Relaciones de Uso: 2.

Acoplamiento: Ninguno=0, Bajo=1, Medio=2, Alto>2.

Complejidad Mantenimiento: Baja<=2, 2<Media<4, Alta>4.

Reutilización: Baja>4, 2<Media<4, Alta<=2.

Cantidad de Pruebas: Baja<=2, 2<Media>4, Alta>4.

Tabla 9. Resultados de la evaluación de la métrica RC y su influencia en los atributos de calidad.

Clase	Cantidad de	Acoplamiento	Complejidad de	Reutilización	Cantidad

	Relaciones de Uso		Mantenimiento		de Pruebas
Atributo	1	Bajo	Baja	Alta	Baja
Cache	1	Bajo	Baja	Alta	Baja
Comprimir	1	Bajo	Baja	Alta	Baja
Conexion	2	Medio	Baja	Alta	Baja
Consulta	1	Bajo	Baja	Alta	Baja
Generadora	2	Medio	Baja	Alta	Baja
Paginacion	1	Bajo	Baja	Alta	Baja
Proyecto	1	Bajo	Baja	Alta	Baja
ProyectoController	8	Alto	Alta	Baja	Alta

De acuerdo con estos resultados se construyeron los gráficos que se muestran seguidamente para hacer más viable el entendimiento de los valores alcanzados por los atributos evaluados:

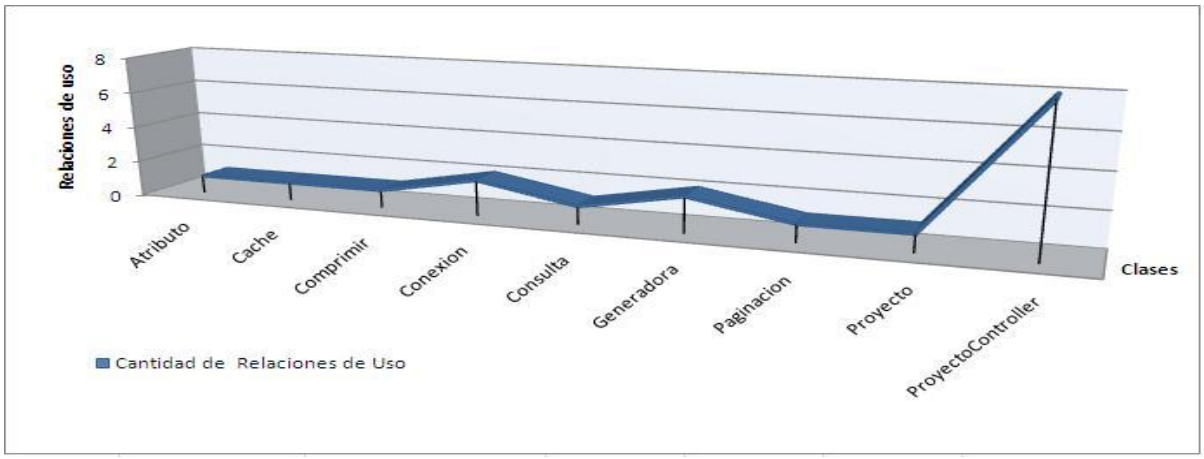

Figura 24. Gráfica que representa los resultados obtenidos después de aplicar la métrica RC (relación de la cantidad de relaciones de uso por clases).

Figura 25. Gráfica que representa la cantidad de clases por intervalos de relaciones de uso.

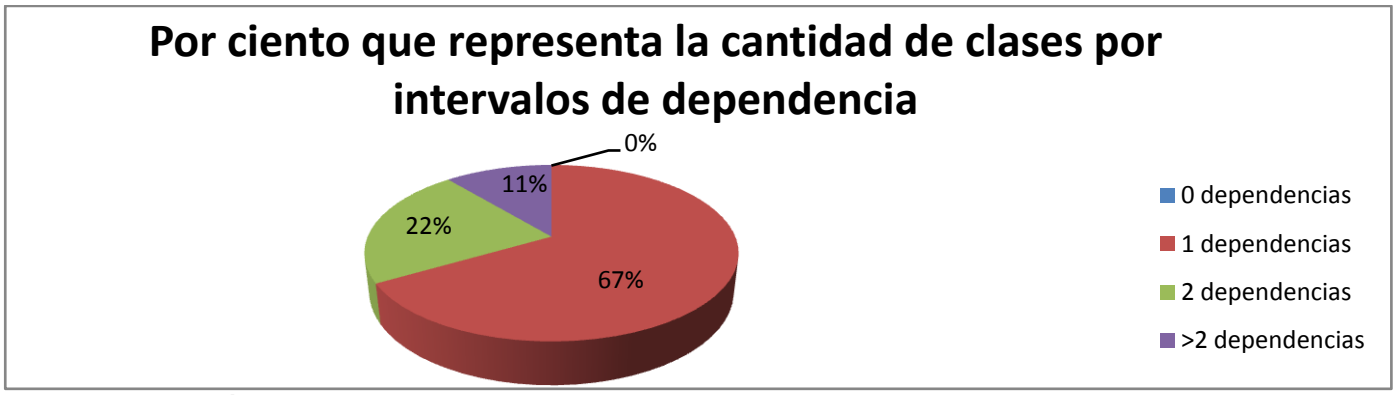

Figura 26. Gráfica que representa los % de la cantidad de clases por intervalos de relaciones de uso.

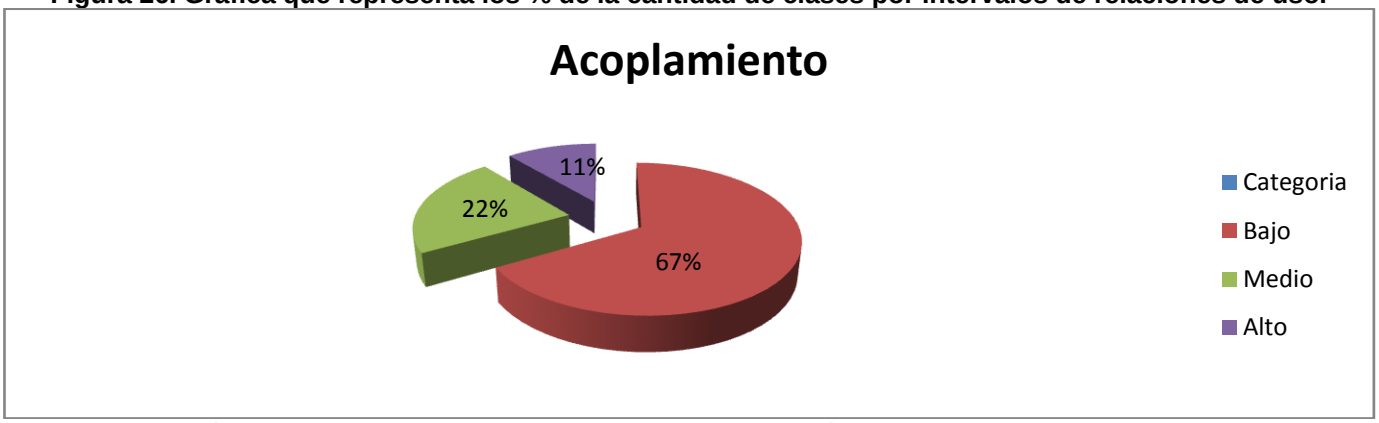

Figura 27. Gráfica que representa los % de clases por categorías del atributo de calidad Acoplamiento obtenidos en la aplicación de la métrica RC.

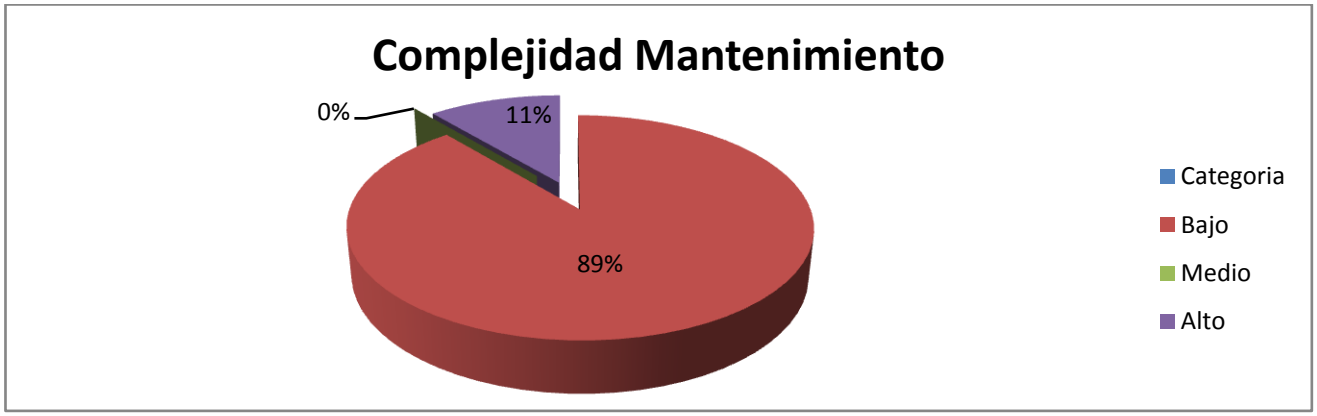

Figura 28.Gráfica que representa los % de clases por categorías del atributo de calidad Complejidad Mantenimiento obtenidos en la aplicación de la métrica RC.

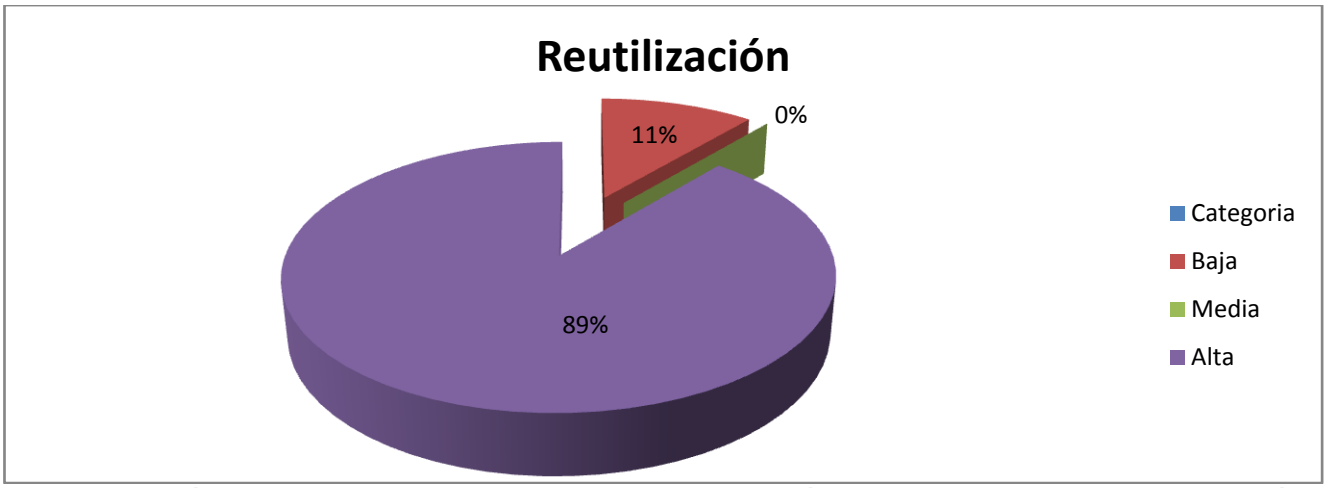

Figura 29. Gráfica que representa los % de clases por categorías del atributo de calidad Reutilización obtenidos en la aplicación de la métrica RC.

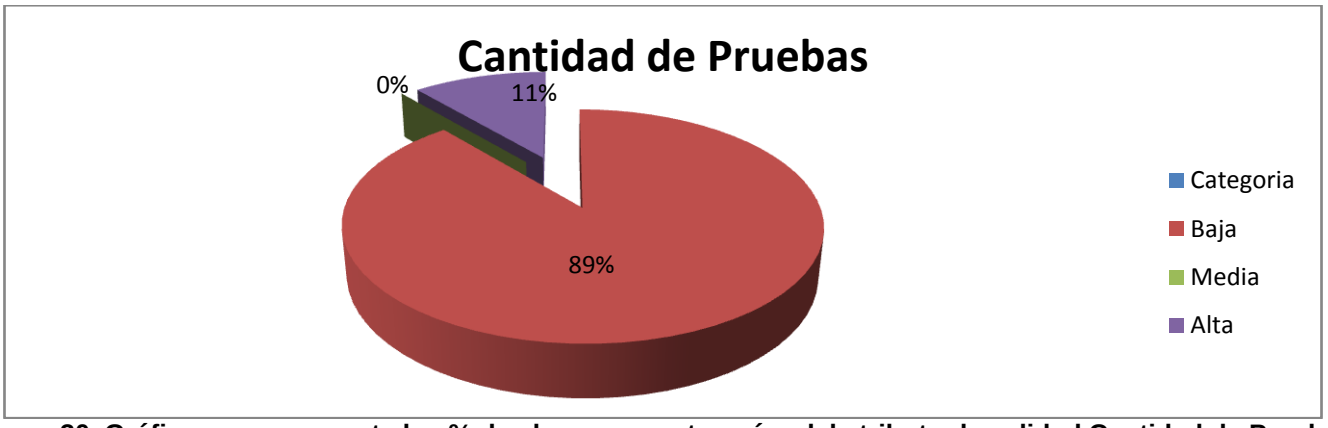

Figura 30. Gráfica que representa los % de clases por categorías del atributo de calidad Cantidad de Pruebas obtenidos en la aplicación de la métrica RC.

La evaluación de los resultados obtenidos durante la aplicación del instrumento de medición de la métrica RC, demuestra que el diseño asumido en la implementación del Doctrine Generator 2.0 tiene una calidad aceptable, puesto que, el 89% de las clases tienen 2 o menos dependencias unas de otras. Por su parte, el nivel de acoplamiento entre las clases se mantiene entre bajo y medio en un 89%, lo cual es un factor aceptable desde el punto de vista de la implementación del software. En cuanto a los atributos de Cantidad de las pruebas de unidad necesarias y de Complejidad para el mantenimiento, se mantienen en niveles bajos de un 89%, mientras que la capacidad de Reutilización del software alcanza niveles de 89%

de forma positiva. Estos resultados de los atributos de calidad se suman a los obtenidos por las pruebas de TOC y demuestran el uso de un buen diseño de software.

3.3 Matriz de cubrimiento o matriz de inferencia de indicadores de calidad

Se le llama matriz de cubrimiento o matriz de inferencia de indicadores de calidad a la representación estructurada de los atributos de calidad y las métricas utilizadas para evaluar la calidad de diseño de un sistema. En este caso se hace para conocer si el resultado obtenido de la relación atributos/métricas para esta solución propuesta son positivos o negativos; de ser 1 son positivos y 0 significa que no son favorables, en caso de que no exista relación se tomará valor nulo (-). Cuando todos los datos sean completados de acuerdo con su valor se realiza un cálculo donde se promedia la sumatoria de los valores obtenidos de un atributo por cada métrica aplicada y la división de dicha sumatoria por la cantidad de métricas evaluadas (solo se promedian las que arrojan un resultado, es decir, las que son nulas no se contemplan). El valor final de esta solución es el que va a tener el atributo dentro de una tabla que medirá si los atributos fueron buenos, regulares o malos. En las tablas y figura que se muestran a continuación se observan los siguientes resultados:

Tabla 10. Resultados obtenidos de la relación Atributos/Métricas aplicados en la solución.

Atributos/Métricas	TOC	RC	Promedio
Responsabilidad	1	-	1
Complejidad de Implementación	1	-	1
Reutilización	1	1	1
Acoplamiento	-	1	1
Complejidad de Mantenimiento	-	1	1
Cantidad de pruebas	-	1	1

Tabla 11. Rango de valores para la evaluación técnica de los atributos de calidad evaluados por cada métrica.

Categoría	Rango de valores
Malo	<= 0.4
Regular	>0.4 y <=0.7
Bueno	>0.7

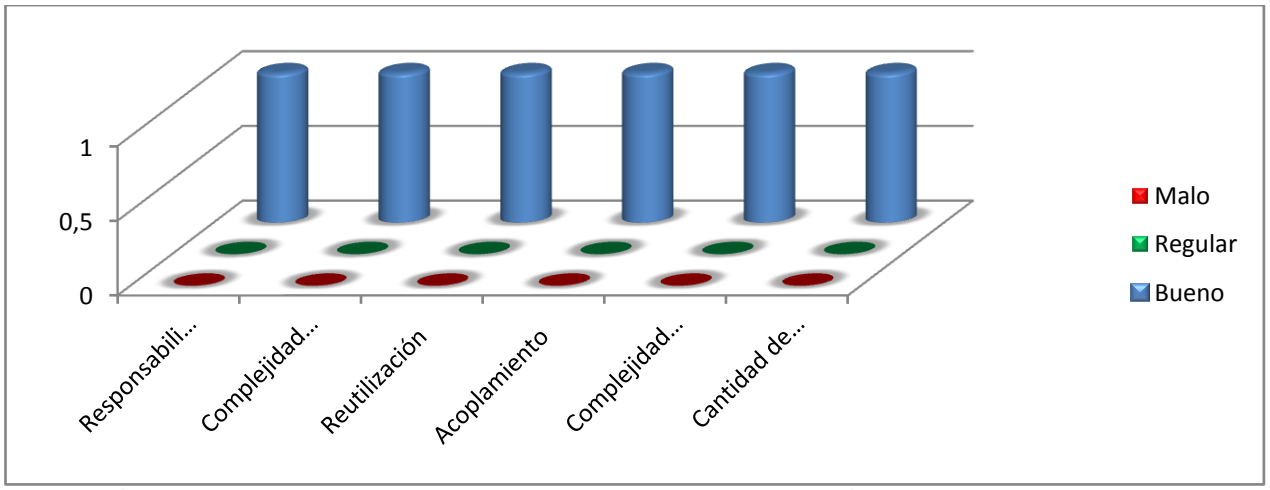

Figura 31. Gráfica que representa los resultados obtenidos de la relación entre los atributos de calidad y las métricas aplicadas.

Tal y como se evidencia en la figura 31 donde se representa la gráfica que relaciona los atributos de calidad evaluados en las métricas aplicadas al diseño, se puede observar que todos estos atributos son categorizados de buenos dentro de los resultados obtenidos. Por lo que la aplicación de la matriz de cubrimiento sobre las métricas seleccionadas, consolida que el diseño de esta aplicación es aceptable y eficiente, resultados alcanzados hasta el momento en cada una de ellas.

3.4 Pruebas realizadas a la solución

"El desarrollo de sistemas de software implica una serie de actividades de producción en las que las posibilidades de que aparezca el fallo humano son enormes. Los errores pueden empezar a darse desde el primer momento del proceso, en el que los objetivos pueden estar especificados de forma errónea o imperfecta, así como en posteriores pasos de diseño y desarrollo. Debido a la imposibilidad humana de trabajar y comunicarse de forma perfecta, el desarrollo de software ha de ir acompañado de una actividad que garantice la calidad". (Pressman, 1997)

Las actividades que garantizan la calidad y el correcto funcionamiento de un software determinado no son más que las pruebas que se le realizan a este. A continuación se reflejan algunas características que se asumen para las pruebas a realizar:

- El objetivo principal de las pruebas será validar la corrección de cualquier artefacto que se vaya a probar. Las pruebas realmente exitosas son aquellas que encuentran defectos.
- Es posible probar todos los artefactos, no solamente el código fuente. Como mínimo se deben revisar los modelos y documentos para de esta forma encontrar y corregir los defectos antes que lleguen al código.
- En la medida en que pueda ser más riesgoso algún artefacto o sistema en general, será más necesario que este sea revisado y probado.
- Es posible creer que la aplicación funciona correctamente, pero solo cuando se muestren los resultados de las pruebas se podrá estar seguro de la veracidad de este hecho.

3.4.1 Pruebas de Caja Blanca

Según plantea Pressman: *"La prueba de caja blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que:*

1- Garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo.

2- Ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.

3- Ejecuten todos los bucles en sus límites y con sus límites operacionales;

4- *Ejerciten las estructuras internas de datos para asegurar su validez."* (Pressman, 1997)

Pruebas del Camino Básico

Por su parte Pressman define dentro de las pruebas de caja blanca que: *"La prueba del camino básico es una técnica de prueba de caja blanca propuesta inicialmente por Tom McCabe [MCC76]. El método del camino básico permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa."* (Pressman, 1997)

Toda sentencia de código tiene una representación en forma de nodo o conjunto de ellos, dentro del grafo de flujo que describe un algoritmo, y los caminos que se pueden definir a través de las aristas que unen los nodos en este grafo representan los posibles caminos básicos de un algoritmo.

Un grafo de flujo está formado por 4 componentes fundamentales. Estos son:

- **Nodo:** son los círculos representados en el grafo de flujo, el cual representa una o más secuencias del procedimiento, donde un nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.
- **Aristas:** son constituidas por las flechas del grafo, son iguales a las representadas en un diagrama de flujo y constituyen el flujo de control del procedimiento. Las aristas terminan en un nodo, aun cuando el nodo no representa la sentencia de un procedimiento.
- **Regiones:** áreas delimitadas por las aristas y nodos donde se incluye el área exterior del grafo, como una región más. Se enumeran siendo la cantidad de regiones equivalente a la cantidad de caminos independientes del conjunto básico de un procedimiento.
- **Nodos predicados**: cuando en una condición aparecen uno o más operadores lógicos (AND, OR, XOR,...) se crea un nodo distinto por cada una de las condiciones simples. Cada nodo generado de esta forma se denomina nodo predicado.

En aras de lograr la aprobación y verificación del código implementado del sistema se aplicaron, entre otras, pruebas del camino básico a sus principales algoritmos no triviales, para los que se diseñaron diferentes casos de prueba que abarcaran todos los posibles caminos por donde estos algoritmos podían arribar a una solución. Tomando como ejemplo un fragmento de código que compone la solución y que representa uno de estos algoritmos no triviales se obtuvieron los siguientes resultados.

Según el código correspondiente a la función de cargaresquemasAction() se enumeran las diferentes sentencias de acuerdo con los criterios definidos para formar el grafo correspondiente.

```
177
178     function cargaresquemasAction()
179     {
180         $nombre_proyecto = $_SESSION['nombre_proyecto']; (1)
181         $conexion= Conexion::getInstance(); (1)
182         $listadodeesquemasbd= $conexion->ObtenerTodosEsquemas(); (1)
183         foreach ($listadodeesquemasbd as $esquema) (2)
184         {
185             $listadodeesquemas[]=$esquema['table_schema']; (3)
186         } (4)
187         for($i=0; $i<count($listadodeesquemas); $i++) (5)
188         {
189             if(strcmp($listadodeesquemas[$i],"") != 0 ) (6)
190             $listadodeesquemas['esquemas'][$i] = array('idesquema'=>$i+1,'nombreesquema'=>$listadodeesquemas[$i]); (7)
191         } (8)
192         echo json_encode($listadodeesquemas); (9)
193     } (9)
194
```

Figura 32. Representación del algoritmo no trivial correspondiente a la función cargaresquemasAction().

A partir de las sentencias de código enumeradas se crea el grafo de flujo asociado a estas, el cual queda como sigue.

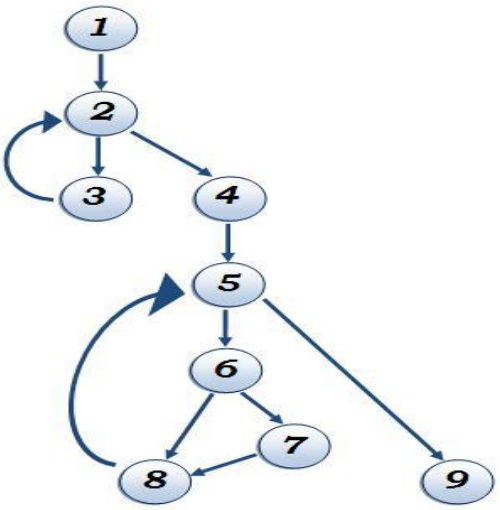

Figura 33. Grafo de flujo asociado la función cargaresquemasAction().

La complejidad ciclomática de este grafo es calculada aplicando las 3 fórmulas existentes para el análisis de la complejidad de algoritmos, garantizando que los resultados obtenidos en cada una de ellas sean los mismos para que el cálculo de la complejidad sea correcto. Estas fórmulas son:

1. V (G) = (A (Aristas) – N (Nodos)) + 2.
2. V (G) = P (Nodos Predicados) + 1.
3. V (G) = R.

Una vez aplicadas estas fórmulas al grafo de flujo de la figura anterior se obtienen los siguientes resultados:

Aplicando la fórmula 1:

V (G) = (11 – 9) + 2

V (G) = 4.

Aplicando la fórmula 2:

V (G) = 3 + 1

V (G) = 4

Aplicando la fórmula 3:

V (G) = 4

Los resultados obtenidos demuestran que la complejidad ciclomática del código es de 4, lo que significa que existen 4 posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento seleccionado.

Los caminos básicos definidos son extraídos del grafo y a partir de ellos se crean los casos de prueba correspondientes.

Camino Básico #1:

(1 – 2 – 4 – 5 – 9)

Camino Básico #2:

(1 – 2 – 3 – 2 – 4 – 5 – 9)

Camino Básico #3:

(1 – 2 – 3 – 2 – 4 – 5 – 6 – 8 – 5 – 9)

Camino Básico #4:

(1 – 2 – 3 – 2 – 4 – 5 – 6 – 7 – 8 – 5 – 9)

Para realizar los casos de prueba de cada camino básico es preciso cumplir con las siguientes exigencias:

- Descripción: se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.
- Condición de ejecución: se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.
- Resultados Esperados: se expone resultado que se espera que devuelva el procedimiento.
- Evaluación de los resultados: se exhibe la evaluación que dio el resultado final del procedimiento.

Caso de prueba para el camino básico # 1.

Descripción: se quiere determinar si dado un arreglo vacío de una consulta de base de datos que carga los esquemas se puede cargar algún esquema, los datos cumplirán con los siguientes requisitos: el arreglo $listadodeesquemasbd está vacío.

Condición de ejecución:

- Arreglo $listadodeesquemasbd estará vacío. ($listadodeesquemasbd = null)

Resultados esperados: al ejecutar el procedimiento no se cargará ningún esquema en el combobox de esquemas.

Evaluación de los resultados obtenidos: los resultados de la realización del caso de prueba de caja blanca para el caso de cargar los esquemas de una base de datos, fueron satisfactorios al lograr que no se cargara ningún esquema a partir de un arreglo vacío.

Caso de prueba para el camino básico # 2.

Descripción: se quiere determinar si dado un arreglo de una consulta de base de datos que carga los esquemas se puede cargar algún esquema, los datos cumplirán con los siguientes requisitos: el arreglo $listadodeesquemas está vacío.

Condición de ejecución:

- Arreglo $listadodeesquemasbd = "SELECT DISTINCT table_schema FROM information_schema.tables WHERE table_schema <> 'information_schema' and table_schema <> 'pg_catalog' ORDER BY table_schema"
- Arreglo $listadodeesquemas estará vacío porque en el resultado de la consulta no existe ningún esquema. ($listadodeesquemas = null).

Resultados esperados: al ejecutar el procedimiento no se cargará ningún esquema en el combobox de esquemas.

Evaluación de los resultados obtenidos: los resultados de la realización del caso de prueba de caja blanca para el caso de cargar los esquemas de una base de datos, fueron satisfactorios al lograr que no se cargara ningún esquema a partir de un arreglo de una consulta de base de datos sin esquemas.

Caso de prueba para el camino básico # 3.

Descripción: se quiere determinar si dado un arreglo de una consulta de base de datos que carga los esquemas se puede cargar algún esquema, los datos cumplirán con los siguientes requisitos: el arreglo $listadodeesquemas contiene nombres de esquemas en blanco.

Condición de ejecución:

- Arreglo $listadodeesquemasbd = "SELECT DISTINCT table_schema FROM information_schema.tables WHERE table_schema <> 'information_schema' and table_schema <> 'pg_catalog' ORDER BY table_schema"
- Arreglo $listadodeesquemas estará formado por valores en blanco. ($listadodeesquemas [1] = “ ”;)

Resultados esperados: al ejecutar el procedimiento no se cargará ningún esquema en el combobox de esquemas.

Evaluación de los resultados obtenidos: los resultados de la realización del caso de prueba de caja blanca para el caso de cargar los esquemas de una base de datos, fueron satisfactorios al lograr que no se cargara ningún esquema a partir de un arreglo de esquemas con valores en blanco.

Caso de prueba para el camino básico # 4.

Descripción: se quiere determinar si dado un arreglo de una consulta de base de datos que carga los esquemas se puede cargar algún esquema, los datos cumplirán con los siguientes requisitos: el arreglo $listadodeesquemas contiene el nombre de esquema “mod_traza”.

Condición de ejecución:

- Arreglo $listadodeesquemas = "SELECT DISTINCT table_schema FROM information_schema.tables WHERE table_schema <> 'information_schema' and table_schema <> 'pg_catalog' ORDER BY table_schema"
- Arreglo $listadodeesquemas posee valores válidos. ($listadodeesquemas [1] = “mod_traza”;)

Resultados esperados: al ejecutar el procedimiento se cargará el comobox de esquemas con el esquema “mod_traza”.

Evaluación de los resultados obtenidos: los resultados de la realización del caso de prueba de caja blanca para el caso de cargar los esquemas de una base de datos, fueron satisfactorios al lograr que se cargara el esquema “mod_traza” en el combobox de los esquemas, a partir de los datos entrados.

3.4.2 Pruebas de Usuario

Las pruebas realizadas por el usuario constituyen pruebas de caja negra que se centran en el cumplimiento de los requisitos definidos para el sistema una vez concluido. Por lo tanto, estas pruebas constituyen la validación de la aplicación por parte del usuario final y según Pressman plantean lo siguiente:

"La validación del software se consigue mediante una serie de pruebas de caja negra que demuestran la conformidad con los requisitos. Un plan de prueba traza la clase de pruebas que se han de llevar a cabo, y un procedimiento de prueba define los casos de prueba específicos en un intento por descubrir errores de acuerdo con los requisitos. Tanto el plan como el procedimiento estarán diseñados para asegurar que se satisfacen todos los requisitos funcionales, que se alcanzan todos los requisitos de rendimiento, que la documentación es correcta e inteligible y que se alcanzan otros requisitos (por ejemplo, portabilidad, compatibilidad, recuperación de errores, facilidad de mantenimiento)". (Pressman, 1997)

Dentro de las pruebas de aceptación no formales, realizadas en software a la medida del cliente y muy convenientes cuando se desarrollan aplicaciones que van a ser utilizadas por muchos clientes, con el objetivo de descubrir errores que parezca que sólo el usuario final puede descubrir, se encuentran las pruebas alfa y beta. Estas pruebas pueden aplicarse durante semanas o meses y van desde un informal "paso de prueba" hasta la ejecución continua de una serie de pruebas bien planificadas.

Prueba Alfa

"La prueba alfa se lleva a cabo, por un cliente, en el lugar de desarrollo. Se usa el software de forma natural con el desarrollador como observador del usuario y registrando los errores y los problemas de uso. Las pruebas alfa se llevan a cabo en un entorno controlado." (Pressman, 1997)

Prueba Beta

"La prueba beta se lleva a cabo por los usuarios finales del software en los lugares de trabajo de los clientes. A diferencia de la prueba alfa, el desarrollador no está presente normalmente. Así, la prueba beta es una aplicación "en vivo" del software en un entorno que no puede ser controlado por el desarrollador. El cliente registra todos los problemas (reales o imaginarios) que encuentra durante la prueba beta e informa a intervalos regulares al desarrollador." (Pressman, 1997)

Dada la condición de que las aplicaciones web son utilizadas por un gran número de usuarios simultáneamente y el hecho de que la versión 2.0 del Doctrine Generator descrita en este trabajo sea una aplicación de este tipo, junto a las necesidades de demostrar el cumplimiento de los requisitos propuestos

del sistema y con ello la aprobación de los desarrolladores del proyecto que en este caso constituyen el cliente final; se decidió aplicar las pruebas de tipo alfa y beta una vez obtenida la solución.

En este caso las pruebas alfas fueron realizadas por varios de los desarrolladores del proyecto en conjunto con el equipo de trabajo que implementó la aplicación en sus puestos de trabajo. Los errores que se detectaron constituyeron cambios sencillos en la implementación de algunas funcionalidades y en el diseño de algunas vistas. Una vez terminados contribuyeron a refinar el software y a acercarlo un tanto más a las necesidades del proyecto.

Por su parte, las pruebas betas estuvieron a cargo del equipo de trabajo del Departamento de Calidad encargado de llevar el control de la gestión de la calidad de software en el proyecto, el cual se ocupa de probar, mediante la aplicación de diferentes casos de prueba, todas las aplicaciones que se desarrollen en el proyecto, en un ambiente semejante al que tendrá la aplicación cuando esté desplegada y en explotación. Los resultados de este equipo se reflejan en documentos que contienen las no conformidades detectadas que atentan contra los requerimientos que debe cumplir el sistema. Este proceso se torna iterativo hasta tanto el Departamento de Calidad deje de generar no conformidades. La liberación por parte del equipo de Calidad del software constituye en el proyecto, el resultado más consistente de que el software se encuentra en completa culminación, independientemente de las demás validaciones y pruebas que ejecute el equipo de desarrollo. La liberación del Doctrine Generator 2.0 tras 3 iteraciones de prueba por el departamento de Calidad, evidencian con gran certeza que el software está en su total refinación.

3.4.3 Casos de Prueba

El resultado obtenido en las pruebas beta del software se debe en gran medida a la aplicación por parte del Departamento de Calidad, de los casos de prueba que se definieron para cada uno de los requisitos del sistema. Su objetivo principal es demostrar la reacción que corresponderá por parte del sistema luego de realizar alguna acción en el mismo. Para un mejor entendimiento de las respuestas o posibles funcionalidades que brindará el sistema, según la necesidad del usuario. A continuación se muestran las condiciones de ejecución para el caso de prueba correspondiente al requisito Crear un nuevo proyecto, donde se recoge la descripción de las variables y los juegos de datos a probar.

CP #1: Crear un nuevo proyecto

1. **Requisito a probar**

Tabla 12. Requisitos a probar del CP Crear un nuevo proyecto.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Crear Nuevo Proyecto	Se crea un nuevo proyecto	EP 1.1: Crear Proyecto	• Seleccionar del menú la opción **Crear Proyecto**. • Se insertan todos los datos del proyecto. • Se presiona el botón **Siguiente.**
		EP 1.2: Crear proyecto dejando campos requeridos en blanco.	• Seleccionar del menú la opción **Crear Proyecto.** • Se insertan todos los datos del proyecto dejando campos en blanco. • Se presiona el botón **Siguiente.**
		EP 1.3: Cancelar	• Seleccionar del menú la opción **Crear Proyecto.** • Se insertan todos los datos del proyecto. • Se presiona el botón **Cancelar.**

2. Descripción de variable

Tabla 13. Descripción de variable del CP Crear un nuevo proyecto.

No	Nombre de campo	Tipo	Válido	Inválido	Inválido	Inválido	Inválido

1	Nombre	Campo de texto	Combinación de letras, números y otros caracteres.	-	-	-	-

3. Juegos de datos a probar

Tabla 14. Juegos de datos a probar del CP Crear un nuevo proyecto.

Id del escenario	Escenario	Variable 1(Nombre)	Respuesta del sistema	Resultado de la prueba
EP 1.1	Crear Proyecto	V (finanzas)	Se crea una carpeta en el directorio raíz del servidor con el nombre del proyecto	
EP 1.2	Crear proyecto dejando campos requeridos en blanco.	I (Vacío)	Se muestra el campo de texto en rojo que indica que no se puede dejar el campo en blanco.	
EP 1.3	Cancelar	NA	Se cancela la operación y se cierra la ventana Nuevo	

3.5 Conclusiones parciales

En este capítulo se aplicaron métricas dirigidas a evaluar la calidad del diseño del software, basadas en diferentes atributos de calidad que avalaron sus resultados. Con la métrica Tamaño Operacional de Clase se evidencia que la implementación no es complicada, que la mayoría de las clases tiene una baja responsabilidad y que a su vez son muy reutilizables. Por su parte la de Relaciones entre Clases demuestra que la implementación del sistema tiene una calidad aceptable, con niveles bajos de acoplamiento, de cantidad de pruebas de unidad y de la complejidad para el mantenimiento, por lo que se demuestra el uso de un buen diseño de software. La matriz de cubrimiento o matriz de inferencia de los

atributos de calidad permitió valorar el impacto que tuvieron los atributos que se midieron en las métricas aplicadas en el diseño de la solución propuesta. Por lo que se puede concluir que el diseño propuesto no presenta complejidad, es fácil de implementar y dar mantenimiento, tiene una alta cohesión y es fácil de probar.

Para garantizar el buen funcionamiento del código implementado, así como su eficiencia y solidez se realizaron las pruebas del camino básico como parte de las pruebas de caja blanca, y para demostrar el cumplimiento de los requisitos definidos, se aplicaron pruebas de caja negra que fueron realizadas por el usuario, dentro de las que se incluyen la alfa y la beta, estas últimas guiándose por los casos de prueba que se definieron para cada uno de los requisitos del sistema. Todo esto permitió verificar que las funcionalidades implementadas responden a las necesidades y propósitos de los clientes.

CONCLUSIONES GENERALES

Al desarrollar la versión 2.0 del Doctrine Generator utilizando las herramientas y tecnologías propuestas, se logró que esta formara parte del conjunto de herramientas del marco de trabajo del proyecto ERP y por ende, se incluyó dentro de su paquete de instalación. Todo esto fue posible gracias al estudio del estado del arte realizado sobre la versión existente y sobre las herramientas, tecnologías y técnicas aplicadas como parte de la fundamentación teórica del sistema a desarrollar, lo cual sentó las condiciones para su posterior análisis, diseño e implementación.

Los resultados obtenidos fueron analizados mediante la validación de su diseño y una etapa de pruebas que aseveró el cumplimiento de los requerimientos propuestos.

Esta nueva versión de la herramienta existente se encuentra en funcionamiento y goza de muy buena aceptación entre los desarrolladores del proyecto, que además, son sus clientes finales.

RECOMENDACIONES

Se recomienda para el desarrollo de futuras versiones o la adición de nuevas funcionalidades del Doctrine Generator 2.0:

- Hacer un uso más profundo de las facilidades que brinda el framework de Doctrine, con el fin de garantizar las conexiones y mapeos con otros gestores de bases de datos relacionales.
- Extender a otros lenguajes de programación la generación de ficheros de mapeo, además del PHP.
- Lograr el proceso inverso al que realiza esta herramienta, o sea, la generación de bases de datos relacionales a partir de los ficheros mapeados. Esta opción junto a la de conectar a diferentes gestores permiten la migración entre bases de datos.
- Hacer uso de la clase Validate del framework utilizado para fortalecer el tratamiento de errores de la solución.

BIBLIOGRAFIA CITADA

Bergin, Joseph. 2010. *Building Graphical User Interfaces with the MVC pattern. [Online] 2 4, 2010. http://csis.pace.edu/ bergin/mvc/mvcgui.html.*

Bonanata, M. 2003. Programación y algoritmos.

Burbeck, Steve. 2009. *Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC). [Online] 10 3, 2009. http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html.*

Cadavid, Andrés Navarro. 2008. *Sistema de investigación de la Universidad Icesi. Sistema de investigación de la Universidad Icesi. [En línea] [Citado el: 15 de 12 de 2009.] http://www.icesi.edu.co/investigaciones/proyecto.do?id=38 .*

Castillo, Juan Marmol. 2009. *Universidad de Informática de Murcia. Universidad de Informática de Murcia. [Online] [Cited: 11 20, 2009.] http://dis.um.es/~jmolina/Persistencia%20de%20Objeto%20JDO.pdf.*

Commons, Creative. 2009. *SOA Agenda. SOA Agenda. [En línea] http://soaagenda.com/journal/articulos/que-son-los-frameworks/.*

Deacon, John. 2010. *Model-View-Controller (MVC) Architecture. [Online] 1 20, 2010. http://www.jdl.co.uk/briefings/mvc.pdf.*

Larman, Craig. 2003. *UML y Patrones: Introducción al análisis y programación orientada a objetos.*

Martinez, Rafael. 2009. *Portal de Postgres en Español. Portal de Postgres en Español. [En línea] 2009. [Citado el: 29 de 11 de 2009.] http://www.postgresql-es.org/sobre_postgresql.*

O'Reilly. 2003. *Learning Python, Fourth Edition .*

PostgreSQL Global Development Group . 1996. *Sitio Oficial de Postgres. Sitio Oficial de Postgres. [En línea] 1996. [Citado el: 3 de 11 de 2009.] http://www.postgresql.org/docs/8.2/static/catalogs.html.*

Pressman, Roger S. 1997. *Ingeniería del Software, un enfoque práctico.* 1997.

Hernández Cisneros, Sergio and Sarduy Pérez, Mileidy Magalys. 2009. *Propuesta de modelo de desarrollo de software tecnológico del Centro de Soluciones de Gestión.* 2009.

Scribd 2005*. (http://www.scribd.com/doc/15493687/DIAGRAMAS-DE-SECUENCIA) (accessed 3 15, 2010).*

SENCIOLABS. 2009. *Doctrine ORM for PHP. 2009.*

UCID. 2008. *ERP-ARQ Estandar para el diseño de interfaces v1.1. La Habana : s.n., 2008.*

Universidad de la Republica de Uruguay. 2009, noviembre 19. *Retrieved diciembre 3, 2009, from Universidad de la Republica de Uruguay: http://www.iie.fing.edu.uy/ense/asign/tap/obrar09/...comp/.../VOCABULARIO.doc*

Universidad Nacional Autónoma de México. 2008. *Retrieved noviembre 27, 2009, from UNAM: http://fcasua.contad.unam.mx/apuntes/interiores/docs/2005/administracion/5/1553.pdf*

Universidad Tecnica Particular de Loja. **2009.** *Retrieved diciembre 15, 2009, from Universidad Tecnica Particular de Loja: http://blogs.utpl.edu.ec/disenowebymultimedia/page/6/*

Zend Technologies Inc. 2005. *Manual Zend Framework Español. Manual Zend Framework Español. [En línea] 2005. http://manual.zfdes.com/es/introduction.overview.html.*

BIBLIOGRAFÍA GENERAL

1. ***Zaninotto, Francois and Potencier, Fabien.*** *Guía Definitiva. 2003.*

2. ***Wage, Jonathan H.*** *What's new in Doctrine. 2009.*

3. ***Vesterinen, Konsta.*** *Doctrine Manual. 2008.*

4. ***Paré, Rafael Camps, et al.*** *Curso de Bases de Datos. 2005.*

5. ***PostgreSQL, Equipo de Desarrollo.*** *Tutorial de PostgreSQL. 1999.*

6. ***PostgreSQL, Equipo de Desarrolladores.*** *Manual del usuario de PostgreSQL. 2000.*

7. ***Pérez, Javier Eguíluz.*** *Introducción a XHTML. 1999.*

8. ***Gonzalez Castellanos, Ma. Argenis and Rojas Pabon, Wilson.*** *Comparación entre sistemas de gestión de bases de datos (SGBD) bajo licenciamiento libre y comercial. 2005.*

9. ***Ltd, Zend Technologies.*** *Sitio Zend. [Online] 2009. [Cited: octubre 2, 2009.] http://www.zend.com/en/products/studio/.*

10. *—. Sitio Zend Framework. Sitio Zend Framework. [Online] 2006. [Cited: octubre 14, 2009.] http://framework.zend.com/.*

11. ***Paradigm, Visual.*** *Sitio Visual Paradigm. [Online] 1999. [Cited: noviembre 18, 2009.] http://www.visual-paradigm.com/product/vpuml/.*

12. ***Mokhtarzada, Haroon and Mokhtarzada, Zeki.*** *Sitio Sistemas Web. [Online] 2010. [Cited: febrero 19, 2010.] http://darksystems.webs.com/vsystem.htm.*

13. *Grupo de Soluciones GCINNOVA. [Online] 2007. [Cited: febrero 17, 2010.] http://www.rational.com.ar/herramientas/roseenterprise.html.*

14. ***project, The Propel.*** *Sitio Propel. [Online] 2006. [Cited: octubre 25, 2009.] http://propel.phpdb.org/trac/.*

15. ***PHP, Grupo de.*** *Sitio PHP. [Online] 2001. [Cited: noviembre 21, 2009.] http://www.php.net/.*

16. ***Foundation, Mozilla.*** *Sitio Mozilla. [Online] 2009. [Cited: noviembre 15, 2009.] http://www.mozilla.com/es-ES/.*

17. ***Newton, Mark, et al.*** *Sitio Hibernate. [Online] 2003. [Cited: noviembre 18, 2009.] https://www.hibernate.org/.*

18. ***Ext, LLC.*** *Sitio ExtJS. [Online] 2006. [Cited: octubre 12, 2009.] http://extjs.es/.*

19. ***ACM.*** *Sitio acm. [Online] 2010. [Cited: marzo 5, 2010.] http://www.acm.org/crossroads/espanol/xrds7-4/frameworks.html.*

20. ***Redaccenir, S.L.*** *Portal de Programas. [Online] 2003. [Cited: enero 21, 2010.] http://www.portalprogramas.com/ayuda/c19/programas-online.*

21. ***Attribution, Creative Commons.*** *Doctrine ORM for PHP Doctrine 1.1. 2009-06-29.*