Universidad de las Ciencias Informáticas
Facultad 2

Título: “Desarrollo de un Marco de Trabajo para Aplicaciones Web Empresariales de las FAR”

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE

INGENIERO INFORMÁTICO

Autores:

Pablo Alejandro Valdespino González.

José Raúl Seoane Rodríguez.

Tutores:

Ing. Julio Cesar García Mosquera.

Ing. Yoandrys Verdecia Álvarez.

CIUDAD DE LA HABANA, JULIO DEL 2008.

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____
Firma del Autor
José Raúl Seoane Rodríguez.

_____
Firma del Autor
Pablo Alejandro Valdespino González.

_____
Firma del Tutor
Julio Cesar García Mosquera.

_____
Firma del Tutor
Yoandrys Verdecia Álvarez.

"Podrán morir las personas, pero jamás sus ideas."

Che

AGRADECIMIENTOS

Agradecemos a todas aquellas personas que de una forma u otra han colaborado con la realización de este trabajo. Principalmente a nuestros tutores que han sabido guiarnos durante todo este tiempo.

A todos, gracias.

DEDICATORIA

A mi madre, a mi abuela y a mi abuelo que me han apoyado en todo momento, por sus esfuerzos para que yo sea alguien en la vida y por su amor y dedicación que me han brindado.

A mi padre que aun lejos no se olvida y me educa desde la distancia.

A todos mis hermanos: Adriana, Adrian, Marielina y Patricia.

A toda mi familia que me ha apoyado en todo momento y que tanto han esperado este momento.

A mi novia Zenia por todo su apoyo incondicional y por ser la rubia mas linda de la nívea.

A Manolo por darme toda la atención y el cariño como un hijo más.

A todos mis amigos por los grandes momentos vividos y su apoyo a lo largo de esta carrera.

Pablo A.

Dedico este trabajo a mis padres Idalis Rodríguez Llanes y José Roberto Seoane Rojas, mis hermanos Idalis Seoane Rodríguez, Roberto Seoane Rodríguez y Yanisley León Camacho por haberme guiado a lo largo de la vida, por haber hecho de mí quien soy y por exigirme a diario ser una mejor persona.

A mis tíos, primos y abuela. A todos los profesores que a lo largo de mi vida como estudiante han aportado un grano de arena en mi formación.

A mis amigos Idalberto, Danny, Líber, René, Reynaldo, Pablo, Marleidy, Beatriz, Zenia, Yuliet, Made y demás amistades quienes me han brindado todo su apoyo incondicional y su amistad en los buenos momentos y en los difíciles.

José Raúl

RESUMEN

Cada vez son más las aplicaciones empresariales web que se desarrollan en el Centro de Compatibilización Integración y Desarrollo de productos informáticos para la Defensa (UCID) lo que conlleva a un difícil análisis de que arquitectura y que componentes utilizar para desarrollar cada una de las aplicaciones.

Como propuesta de solución a este problema, en este trabajo se desarrolla una propuesta de arquitectura de software de dominio específico para desarrollar aplicaciones empresariales web sobre una arquitectura en capas con un conjunto de Librerías la cual se nombra AEWeb. Estas Librerías responden a los requerimientos específicos de estas aplicaciones empresariales web para las FAR, se definió un flujo de trabajo para el desarrollo de las aplicaciones empresariales web utilizando AEWeb y se elaboro una propuesta de ambiente de desarrollo.

PALABRAS CLAVE

Arquitectura de software, patrones, Programación Orientada a Objetos, robustez, escalabilidad, reutilización de código, aplicaciones empresariales.

INTRODUCCIÓN

En el desarrollo del software, un Marco de Trabajo (MT) es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un MT puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y relacionar los diferentes componentes de un proyecto.

Un MT 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 extienden o utilizan las aplicaciones del dominio.

El uso de los MT en el campo de las aplicaciones empresariales web se ha convertido en la principal arma para los desarrolladores de estas desde hace unos años hasta la actualidad, pues hasta entonces se le hacia demasiado tedioso el trabajo teniendo que ocuparse por ejemplo del tema de la seguridad de las aplicaciones, siendo hoy tratado por el propio marco.

Los MT son diseñados con el objetivo de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional. Por ejemplo, un equipo que usa Apache Struts para desarrollar un sitio web de un banco puede enfocarse en cómo los retiros de ahorros van a funcionar en lugar de preocuparse de cómo se controla la navegación entre las páginas en una forma libre de errores. Sin embargo, hay quejas comunes acerca de que el uso de MT añade código innecesario y que la preponderancia de los MT competitivos y complementarios significa que el tiempo que se pasaba programando y diseñando ahora se gasta en aprender a usar MT. Fuera de las aplicaciones en la informática, un MT puede ser considerado como el conjunto de procesos y tecnologías usados para resolver un problema complejo. Es el esqueleto sobre el cual varios objetos son integrados para una solución dada.

En el centro UCID se están desarrollando aplicaciones empresariales web. Para facilitar el trabajo de la misma se decidió llevar a cabo la creación de un MT para las FAR, lo cual nos lleva a la siguiente **Situación Problémica:** Actualmente en el centro UCID se ponen de manifiesto problemas tales como: la poca robustez, escalabilidad y reutilización del código en la creación de las aplicaciones que se llevan a cabo en el mismo, la escasa utilización de patrones de diseño, el poco manejo de las excepciones, el modelo de la persistencia no garantiza la integridad de los datos. Todos estos problemas conllevan a que las aplicaciones que se desarrollan en el centro no tengan la calidad

requerida puesto a que no existe una estructura de trabajo con tecnologías avanzadas regida por una arquitectura que permita desarrollar aplicaciones web de manera eficiente, por lo que la actividad productiva se hace muy tediosa a la hora de manejar los datos recogidos y tanto el esfuerzo como el tiempo empleados son costosos. La calidad es baja. No existe uniformidad entre el trabajo desarrollado por distintas personas. El software resultante tiene numerosos defectos y no satisface las expectativas de los clientes.

De ahí que el **Problema Científico** sea: ¿Cómo lograr mayor reutilización del código, robustez y escalabilidad en la creación de aplicaciones empresariales web de las FAR?

En tanto el **Objeto de Estudio** de la investigación son los MT para aplicaciones empresariales y el **Campo de Acción** es el desarrollo de un MT para aplicaciones empresariales en el UCID.

Idea a Defender

Con el desarrollo de un MT para aplicaciones empresariales en el centro UCID se logra:

- ✓ Elevar la eficiencia en la creación de las aplicaciones empresariales.
- ✓ Mayor reutilización del código, robustez y escalabilidad.

Objetivo General

Desarrollar la propuesta de un MT para la creación de las aplicaciones web empresariales en el UCID.

Objetivos Específicos

- ✓ Identificar la arquitectura idónea para el MT.

- ✓ Describir todos los componentes de las capas de la arquitectura propuesta: acceso a datos, negocio y la capa de presentación, así como la interacción entre estas.

Tareas

- ✓ Estudiar las tendencias arquitectónicas.
- ✓ Analizar la arquitectura ideal para el MT.
- ✓ Describir las capas de la arquitectura.
- ✓ Describir la interacción entre las capas de la arquitectura.
- ✓ Realizar el diagrama de clases de la lógica de negocio del MT.

El presente trabajo esta conformado por 3 capítulos de los cuales se brinda a manera de resumen información acerca de ellos.

Capitulo 1- Fundamentación Teórica: Se hace referencia a temas ligados con las tecnologías existentes, las más utilizadas para el desarrollo de sistemas similares a los que el presente trabajo propone, cuestiones referentes con la teoría para el entendimiento del trabajo y también se exponen los conocimientos de las herramientas empleadas para el desarrollo del presente trabajo así como su descripción, y mencionando sus características de manera tal que se argumente el porque del uso de cada una de ellas.

Capitulo 2- Características del Sistema: Se describen detalladamente las características del sistema, diseñando además cada una de las capas lógicas involucradas en la creación de este sistema y de las aplicaciones que se desarrollen sobre las bases estructurales del mismo conjuntamente con la estructura organizacional que van a contener las mismas y los mecanismos de colaboración entre paquetes dentro de cada aplicación, así como la comunicación entre cada una de las capas que se emplean.

Capitulo 3- AEWeb. Fundamentos para el desarrollo: Se mencionan y describen los módulos que contiene el marco de trabajo los cuales se explican grandes rasgos. Se elabora además un flujo de trabajo para organizar y guiar los procesos de desarrollo de las aplicaciones que de aquí surjan. A manera de propuesta del ambiente de desarrollo se mencionan las herramientas y tecnologías con las cuales se deben desarrollar las aplicaciones y en que lugar de las mismas deben ser utilizadas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

En este capítulo se hará un estudio de las tendencias actuales del desarrollo de Marcos de Trabajo y particularmente de los Marcos de Trabajo para aplicaciones empresariales, además de los tipos de Arquitecturas existentes, decidiendo acerca de las herramientas a utilizar en el desarrollo del software que dará solución al problema.

Se realiza también un análisis de los principales lenguajes de programación orientados al desarrollo de Marcos de Trabajo y sus características fundamentales. De forma exhaustiva se estudian y analizan minuciosamente las tendencias arquitectónicas desde sus inicios hasta la actualidad, viendo así mismo sus particularidades y a partir de ello proponer la más idónea para la creación del Marco de Trabajo que de solución a los problemas anteriormente descritos.

1.2 Análisis de las herramientas de desarrollo.

Las herramientas de desarrollo son la base para implementar los disimiles tipos de software. Las herramientas actuales brindan gran número de facilidades por lo cual agiliza el proceso de creación de los productos. Existen herramientas tanto para modelar bases de datos a partir de diagramas relacionales como herramientas sobre las cuales se implementan aplicaciones a través de lenguajes de programación.

1.2.1 Entornos de desarrollo.

Los entornos de desarrollo integrados son programas compuestos por un conjunto de herramientas para un programador. Son el tipo de herramientas utilizadas para la creación de software mediante los distintos lenguajes de programación, los mismos están formados por un editor de código, un compilador, un depurador y un constructor de interfaz gráfica, además pueden funcionar como aplicaciones por si solas o pueden ser parte de aplicaciones existentes.

1.2.1.1 Zend Studio 6.0

Zend Studio es un editor de texto para páginas PHP que proporciona un gran número de ayudas desde la creación y gestión de proyectos hasta la depuración del código, es propietario pero multiplataforma, compatible con Linux, MAC y Windows. Su versión 6.0 está basada en el IDE Eclipse para código abierto. Incluye todos los componentes necesarios durante el ciclo de vida de una aplicación en PHP. Incluye editor, análisis, depuración, optimizadores de código y herramientas de base de datos. Zend Studio permite agilizar el desarrollo web y simplificar proyectos complejos. (pampua, 2008)

El mismo presenta las características, ventajas y desventajas que a continuación se mencionan:

Características:

- ✓ Excelente completamiento de código.
- ✓ Coloreado en la sintaxis del código,
- ✓ Administración avanzada de proyectos.
- ✓ Múltiples lenguajes.
- ✓ Incorpora el Framework de Zend, PHP Documentor, manual de PHP.
- ✓ Integración con subversión, con los navegadores, integración avanzada con FTP.
- ✓ Soporte para Web Services, PHP4, PHP5 y SQL.

Ventajas:

- ✓ Agiliza el trabajo a los desarrolladores.
- ✓ Cuenta con un buen Depurador, infinitas opciones que permiten un desarrollo profesional de nuestras aplicaciones.

Desventajas:

- ✓ Requiere Licencia de pago.
- ✓ No incluye editor visual HTML.

1.2.2 Herramientas CASE (Computer Aided Software Engineering).

CASE o Ingeniería de Software Asistida por Computadora generalmente puede ser aplicado a cualquier sistema o colección de herramientas que ayudan a automatizar el diseño de software y el proceso de desarrollo. Los compiladores, editores estructurados, sistemas de control de código fuente, y las herramientas de modelado, todos ellos son herramientas CASE en el sentido más estricto.

CASE es una filosofía que se orienta a la mejor comprensión de los modelos de empresa, sus actividades y el desarrollo de los sistemas de información. Esta filosofía involucra además el uso de programas que permiten: construir los modelos que describen la empresa, describir el medio en el que se realizan las actividades, llevar a cabo la planificación, el desarrollo del Sistema Informático, desde la planificación, pasando por el análisis y diseño de sistemas, hasta la generación del código de los programas y la documentación. (danysoft, 2008)

1.2.2.1 Rational Rose.

Es una de las más poderosas herramientas de modelado visual para el análisis y diseño de sistemas basados en objetos. Se utiliza para modelar un sistema antes de proceder a construirlo. Cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases. Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software. Utiliza un proceso de desarrollo iterativo controlado (controlled iterative process development), donde el desarrollo se lleva a cabo en una secuencia de iteraciones. Cada iteración comienza con una primera aproximación del análisis, diseño e implementación para identificar los riesgos del diseño, los cuales se utilizan para conducir la iteración, primero se identifican los riesgos y después se prueba la aplicación para que éstos se hagan mínimos. Cuando la implementación pasa todas las pruebas que se determinan en el proceso, esta se revisa y se le añaden los elementos modificados al modelo de análisis y diseño. Una vez que la actualización del modelo se ha modificado, se realiza la siguiente iteración.

Algunas de sus características son:

- ✓ Tiene un proceso de desarrollo iterativo e incremental.
- ✓ Permite que se trabaje en grupo.
- ✓ Se puede generar código en distintos lenguajes de programación a partir de un diseño en UML.
- ✓ Permite la ingeniería inversa a partir de un código (como son Ada, ANSI C ++, C++, CORBA, Java y Visual Basic), aunque no permite la ingeniería inversa a partir de una Base de Datos.

1.2.2.2 Visual Paradigm Suite 3.1

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Algunas de sus características son:

- ✓ Soporte de UML a partir de la versión 2.1
- ✓ Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- ✓ Modelado colaborativo con CVS (Concurrent Version System) y Subversion.
- ✓ Ingeniería de ida y vuelta
- ✓ Ingeniería inversa - Código a modelo, código a diagrama.
- ✓ Generación de código - Modelo a código, diagrama a código.
- ✓ Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos. (freedownloadmanage, 2007)

- ✓ Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- ✓ Diagramas de flujo de datos
- ✓ Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos existentes a diagramas de Entidad-Relación.

1.2.2.3 Selección de la herramienta case a utilizar.

Teniendo en cuenta que Visual Paradigm es de fácil manejo y brinda un gran número de características que no brindan otras herramientas como son Rational Rose que no permite la ingeniería inversa de Bases de datos ya existentes desde los sistemas gestores de Bases de Datos a diagramas de Entidad-Relación, la generación de las Bases de datos en Visual Paradigm es mucho mas manejable que en Rational Rose, además de que permite hacer modelados colaborativos con sistemas de salvas de versiones, Visual Paradigm brinda o permite hacer un análisis textual que es útil y practico para la técnica de captura de requisitos existentes y la identificación de clases, además de que no requiere de plataforma única como la necesita Rational Rose. Por lo anteriormente explicado se decidió utilizar Visual Paradigm para el desarrollo del sistema.

1.3 Tendencias y tecnologías actuales.

Los marcos de trabajo brindan grandes funcionalidades y posibilidades para el desarrollo de las aplicaciones empresariales, siendo así más fácil la implementación de las mismas.

Hoy en día estos productos independientemente de su arquitectura hacen empleo una arquitectura, lenguajes de programación, marcos de trabajos destinados a resolver un tipo de tareas en particular como la gestión de la interfaz de usuario, lógica de negocio y acceso a datos. Hacen uso también de patrones de diseño y arquitectura, entre ellos Modelo-Vista-Controlador, bajo acoplamiento entre otros.

1.3.1 Principales Lenguajes de programación. Selección del lenguaje a utilizar.

El mundo de la informática, más explícitamente el campo de la programación web cuenta hoy con disímiles lenguajes orientados al desarrollo de este tipo de aplicaciones. Estos lenguajes se clasifican en dos tipos de acuerdo a su función. El primero de estos, llamado Lenguajes del lado del cliente podemos mencionar a: Java Script, HTML y Visual Basic Script destinados a configurar un ambiente dinámico en la lógica de presentación o páginas clientes. El segundo concentra a los lenguajes con tecnologías del lado del servidor y entre ellos se encuentran: Active Server Page (ASP), Practical Extracting and Reporting Language (PERL) y Personal Home Page (PHP)**.** Este grupo esta dirigido específicamente al desarrollo de la lógica del negocio y del acceso a la información ya sea de una base de datos o de un servicio web.

1.3.1.1 Lenguajes de programación del lado del cliente.

Los lenguajes de lado cliente (entre los cuales no sólo se encuentra el HTML sino también el Java y el JavaScript los cuales son simplemente incluidos en el código HTML) son aquellos que pueden ser directamente "digeridos" por el navegador y no necesitan un pretratamiento. (Aníbal, 2007)

1.3.1.1.1 JavaScript.

Es un lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. JavaScript es un lenguaje orientado a objetos propiamente dicho, ya que dispone de herencia, esta se realiza siguiendo el paradigma de programación basada en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM.

Tradicionalmente, se venía utilizando en páginas web HTML, para realizar tareas y operaciones en el marco de la aplicación únicamente cliente, sin acceso a funciones del servidor. Actualmente JavaScript se ejecuta en la interfaz del usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML. (webera, 2006)

1.3.1.1.2 Lenguaje de Marcas de Hipertexto (HTML).

Es el lenguaje de marcas de texto utilizado normalmente en la www (World Wide Web). Fue creado en 1986 a partir de dos herramientas preexistentes: El concepto de Hipertexto (Conocido también como link o ancla) el cual permite conectar dos elementos entre si y el SGML (Lenguaje Estándar de Marcación General) el cual sirve para colocar etiquetas o marcas en un texto que indique como debe verse. HTML no es propiamente un lenguaje de programación como C++, Visual Basic, sino un sistema de etiquetas. Este lenguaje de marcas no presenta ningún compilador, quiere decir que si existe algún error de sintaxis que se presente éste no lo detectará sino que se visualizará en el navegador como éste lo entienda.

El entorno para trabajar HTML es simplemente un procesador de texto, como el que ofrecen los sistemas operativos Windows (Bloc de notas) o el que ofrece Microsoft Office (Word). El conjunto de etiquetas que se creen, se deben guardar con la extensión .htm o .html Estos documentos pueden ser mostrados por los navegadores de paginas web como Mozilla Firefox o Microsoft Internet Explorer entre otros. (Pablo Ravioli, 2006)

1.3.1.2 Lenguajes de Programación del lado del servidor.

Los lenguajes de lado servidor que son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él. (Aníbal, 2007)

1.3.1.2.1 Active Server Page (ASP).

Es una tecnología del lado servidor de Microsoft para páginas web generadas dinámicamente, que ha sido comercializada como un anexo a Internet Information Server (IIS). La tecnología ASP está estrechamente relacionada con el modelo tecnológico de su fabricante. Intenta ser solución para un modelo de programación rápida ya que programar en ASP es como programar en Visual Basic, por supuesto con muchas limitaciones ya que es una plataforma que no se ha desarrollado como lo esperaba Microsoft.

Lo interesante de este modelo tecnológico es poder utilizar diversos componentes ya desarrollados como algunos controles ActiveX. Otros problemas que han hecho evolucionar esta tecnología es el no disponer de información que oriente a quienes desean aprenderla y resulta muy costosa en tiempo descubrir aquí y allá toda la información para volverla altamente útil. (tringa, 2008)

1.3.1.2.2 Practical Extracting and Report Language (PERL).

Es un lenguaje de programación muy utilizado para la elaboración de aplicaciones CGI, principalmente para realizar consultas a bases de datos como Oracle, SQL-Server, SyBase, o herramientas locales como WAIS. Perl es un lenguaje para manipular textos, archivos y procesos, prorciona una manera fácil y legible para realizar trabajos que normalmente se pueden realizar en C o un shell. Perl nació y se difundió bajo el sistema operativo UNIX, aunque existe para otras plataformas.

(hostingmacro, 2007)

Esta pensado para ser práctico (fácil de usar, eficiente, completo) en lugar de bonito (pequeño, elegante, mínimo). Perl combina algunas de las mejores características de C, sed, awk, y shell, de manera que las personas familiarizadas con estos lenguajes deberían tener pocas dificultades con ´el. La sintaxis de las expresiones corresponden muy cercanamente a la de las expresiones del C. Soporta tanto la programación estructurada como la programación orientada a objetos y la programación funcional, tiene incorporado un poderoso sistema de procesamiento de texto y una enorme colección de módulos disponibles. (Milone, 2001)

1.3.1.2.3 Personal Home Page (PHP).

PHP es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas Web dinámicas, similar al ASP de Microsoft o el JSP de Sun, embebido en páginas HTML y ejecutado en el servidor. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas de sí mismo. La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas. No es un lenguaje de marcas como podría ser HTML, XML o WML. Está más cercano a Java Script o a C. Es un lenguaje de código abierto (Open Source). Está integrado con el servidor Apache como módulo, lo cual lo hace más rápido y eficiente además de reportar menor uso de memoria en el momento de la compilación.

1.3.1.2.4 Fundamentación de la selección del lenguaje a utilizar.

Se decidió utilizar JavaScript y HTML como lenguajes de programación del lado del cliente porque JavaScript hace uso del paradigma de la POO y HTML porque es un lenguaje muy eficiente a la hora de interpretar el código del lenguaje anterior. PHP como lenguaje de programación del lado del servidor en el presente trabajo puesto que además de ser flexible, consume muy pocos recursos principalmente de memoria y es multiplataforma. Mencionando también que cuenta con una amplia librería de funciones que permiten realizar gran número de tareas, ya sea de procesamiento de formularios y de paginación con un alto poder de sencillez. Además PHP es un lenguaje de código abierto.

1.3.2 Tecnología de desarrollo web.

La utilización de las técnicas de desarrollo web hace que las aplicaciones se creen de una manera mucho más interactiva, es decir, estas están orientadas a los usuarios de las aplicaciones.

1.3.2.1 Asyncronic JavaScript Xml (AJAX).

Debido al carácter tan interactivo entre cliente y el servidor se decide utilizar AJAX, acrónimo de Asíncronos JavaScript And XML (JavaScript y XML asíncronos). Es una técnica de desarrollo Web para crear aplicaciones interactivas. Éstas se ejecutan en el navegador del usuario, y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla, lo que significa aumentar la interactividad, velocidad y usabilidad en la misma.

AJAX es una combinación de tres tecnologías ya existentes:

- ✓ XHTML (o HTML) y hojas de estilos en cascada (CSS por sus siglas en inglés) para el diseño que acompaña a la información.

- ✓ Document Object Model (DOM) accedido con un lenguaje de "scripting" por parte del usuario, especialmente implementaciones ECMAScript como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada.
- ✓ El objeto XMLHttpRequest para intercambiar datos asincrónicamente con el servidor Web.
- ✓ XML es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON y hasta EBML.
- ✓ Como el DHTML, LAMP o SPA, AJAX no constituye una tecnología en sí, sino que es un término que engloba a un grupo de éstas que trabajan conjuntamente.

1.4 Formato de intercambio de datos.

Tanto JSON como XML se pueden usar para representar objetos nativos en memoria en un formato de intercambio de datos basado en texto y legible. Además, los dos formatos de intercambio de datos son isomorfos: dado un texto en un formato, se puede generar uno equivalente en el otro.

1.4.1 Extensible Markup Language (XML).

Es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML (Lenguaje de Marcación Generalizado) y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML y SVG.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad

ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

Ventajas:

- ✓ Es extensible, lo que quiere decir que una vez diseñado un lenguaje y puesto en producción, igual es posible extenderlo con la adición de nuevas etiquetas de manera que los antiguos consumidores de la vieja versión todavía puedan entender el nuevo formato.
- ✓ El analizador es un componente estándar, no es necesario crear un analizador específico para cada lenguaje. Esto posibilita el empleo de uno de los tantos disponibles. De esta manera se evitan bugs[1] y se acelera el desarrollo de la aplicación.
- ✓ Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarlo. Mejora la compatibilidad entre aplicaciones.
- ✓ Para los comentarios posee una compatibilidad nativa y generalmente disponible por medio de las API.

Desventajas:

- ✓ Los objetos se deben expresar mediante convenciones, con frecuencia a través del uso combinado de atributos y elementos
- ✓ Los documentos tienden a tener gran tamaño, especialmente cuando se usa para el formato el enfoque centrado en elementos.
- ✓ Generalmente tiende a necesitar del uso de varias tecnologías en colaboración: XPath (Paquete de Lenguaje XML), esquema XML, XSLT (Transformación del Lenguaje Extensible de

[1]bugs: referido a los errores de software.

Hojas de Estilo), espacios de nombres XML, DOM (Modelo de Objetos para la representación de Documentos).

- ✓ No ofrece ninguna noción de tipos de datos. Hay que usar el esquema XML para agregar información de tipos.

1.4.2 JavaScript Object Notation (JSON).

JSON es un formato de intercambio de datos creado a partir de un subconjunto de la notación de literales de objetos en JavaScript. Aunque JavaScript acepta una sintaxis para valores literales muy flexible, es importante tener en cuenta que JSON posee reglas mucho más estrictas. Según el estándar JSON, por ejemplo, el nombre de un miembro de objeto debe ser una cadena JSON válida. En JSON una cadena se debe encerrar entre comillas.

Ventajas:

- ✓ Posee tipos de datos escalares y la capacidad de expresar datos estructurados a través de matrices y objetos.
- ✓ Tiene compatibilidad con objetos nativos.
- ✓ Reconoce el valor **null** de forma nativa.
- ✓ Los conflictos para asignar nombres se evitan generalmente mediante el anidado de objetos o el uso de un prefijo en el nombre de un miembro de objeto (en la práctica, se prefiere lo primero).
- ✓ Para las decisiones de formato proporciona una asignación mucho más directa para los datos de aplicación. La única excepción puede ser la ausencia del literal de fecha/hora.
- ✓ La sintaxis es muy concisa y da como resultado texto con formato en el que la mayor parte del espacio lo consumen los datos representados.

- ✓ No se necesita código de aplicación adicional para analizar texto; se puede usar la función eval de JavaScript.
- ✓ Curva de aprendizaje muy sencilla.

Desventajas:

- ✓ No posee compatibilidad para el uso de comentarios.

1.4.3 Fundamentación del formato de intercambio de datos a utilizar.

Se decidió utilizar JSON como formato de intercambio de datos porque presenta varias ventajas como son la posesión de tipos de datos escalares y la capacidad de expresar datos estructurados a través de matrices y objetos, tiene compatibilidad con objetos nativos, la sintaxis es muy concisa y da como resultado texto con formato en el que la mayor parte del espacio lo consumen los datos representados, por solo mencionar algunas. Estas ventajas dan como resultado el uso más frecuente de JSON y no de XML por parte de los desarrolladores.

1.5 Plataformas de Trabajo existentes.

Los MT ayudan en el desarrollo de software, proporcionan una estructura definida la cual ayuda a crear aplicaciones con mayor rapidez. Ayuda a la hora de realizar el mantenimiento del sitio gracias a la organización durante el desarrollo de la aplicación. Estos son desarrollados con el objetivo de brindarles a los programadores y diseñadores una mejor organización y estructura a sus proyectos. Se utiliza la Programación Orientada a Objetos (POO), permitiendo la reutilización del código.

1.5.1 Plataforma para la gestión de la interfaz de usuario.

1.5.1.1 Ext js 2.0.

Ext JS es una librería de componentes de diseño que proporciona las herramientas necesarias para la creación de aplicaciones Web con excelente calidad gráfica, pues cuenta con una amplia colección de componentes para el diseño de interfaces, ventanas, pestañas, menús, tablas y demás componentes.

Esta brinda soporte para:

- ✓ Construir interfaces gráficas complejas y dinámicas.
- ✓ Comunicar datos de forma asíncrona con el servidor.
- ✓ Manejar datos de distinta índole de una manera simple.

Patrones que utiliza:

- ✓ Observer.
- ✓ Singleton
- ✓ Flyweight
- ✓ Singleton_fr
- ✓ Builder

Ext JS procesa las peticiones de los usuarios de manera asíncrona ya que utiliza la tecnología Ajax.

Actualmente Ext JS es considerado un marco de trabajo independiente; ya que a principios del 2007 se creó una compañía para comercializar y dar soporte al mismo, dicha compañía proporciona los servicios de consultoría necesarios para ayudar a los clientes en el aprovechamiento máximo de las ventajas de Ext JS. Es importante señalar que la Ext JS en sus versiones anteriores tiene la licencia LGPL (Open Source) y a partir de la versión 2.0 que actualmente se encuentra estable cuenta además con la licencia comercial que es obligatoria si se desea obtener algún tipo de soporte.

Crear un sistema serio con esta herramienta se requiere de un uso y conocimientos previos bien prolongados. Se considera que aprender a trabajar con este marco de trabajo es el equivalente en tiempo a aprender a programar en un nuevo lenguaje de programación. Es difícil la creación de un nuevo objeto dentro de esta por parte de quien haga uso de la misma. Se vuelve además inaccesible a los buscadores ya que es en su totalidad desarrollado en java script y por tanto limita su uso a sistemas y no a sitios web. (js, 2007)

1.5.2 Plataformas de acceso a datos.

1.5.2.1 Doctrine ORM.

Doctrine ORM (Object Relational Mapper) en español Objeto Mapeador Relacional que esta diseñado para exportar una base de datos a sus clases correspondientes, es decir que para quien lo utiliza él brinda la posibilidad de trabajar con objetos que tienen métodos y en su capa de abstracción toma este código y lo convierte en sentencias que entiende cualquier sistema gestor de bases de datos (SGBD). Permite trabajar fácilmente con una base de datos. Los ORM permiten olvidarse de las particularidades de cada sistema de base de datos y permiten crear aplicaciones compatibles con cualquiera de las bases de datos más populares de forma sencilla con una Capa de abstracción a datos (DBAL) incorporado. En la figura 2.3.1.1 del capitulo2 se detalla como esta compuesto el mismo.

A continuación se definen las características de este potente ORM:

- ✓ Presenta una habilidad opcional de escribir consultas OO (Orientadas a objeto) o personalizadas (SQL).
- ✓ Tiene un lenguaje propio llamado DQL (Lenguaje de consulta Doctrine) inspirado en HQL (Lenguaje de consulta Hibérnate) posibilitando una alternativa a SQL y manteniendo al máximo una mejor flexibilidad
- ✓ Entre muchas otras cosas da la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos.
- ✓ Presenta una amplia documentación con una detallada API ,donde se describen cada clase, métodos e interacción entre ellos

- ✓ La capa de abstracción a datos DBAL se completa y extiende de la librería nativa de php. Esta puede utilizarse si lo que se necesita es una capa de abstracción de base de datos poderosa encima de PDO.
- ✓ Para el mapeo de objeto relacional pone en practica el patrón Active Record
- ✓ Esta modularizado en muchos paquetes individuales e independientes.

Este ORM esta basado en varios patrones para lograr que sus disímiles acciones se puedan llevar a cabo, estos patrones son los que a continuación se exponen conjuntamente con la acción que realizan cada uno de ellos:

- ✓ Singleton: Para forzar a una única instancia de Doctrine_Manager
- ✓ Composite: Para la configuración nivelada.
- ✓ Factory: Para cargar el manejo de conexiones.
- ✓ Observer: Para escuchar eventos.
- ✓ Flyweight: Para el uso eficaz de validaciones.
- ✓ Iterator: Para iterar a través de los componentes (Tables, Connections, Records etc.).
- ✓ State: Para las conexiones estado-sabias (state-wise connections).
- ✓ Strategy: Para las estrategias de algoritmos.
- ✓ Active Record: Doctrine implementa este patrón para el mapeo.
- ✓ UnitOfWork: Para mantener una lista de objetos afectados en una transacción.
- ✓ Identity Field: Para mantener la identidad entre el registro y fila de la base de datos.
- ✓ Metadata Mapping: Para Doctrine DataDict.

- ✓ Dependent Mapping: Para mapear en general, secuencialmente todos los archivos que extienden de Doctrine_Record.
- ✓ Foreign Key Mapping: Para relaciones (1-1, 1-*,*-*,*-1).
- ✓ Association Table Mapping: Para mapear asociaciones entre tablas (comúnmente relaciones de muchos a muchos).
- ✓ Lazy Load: Para carga persistente de objetos y propiedades de objeto.
- ✓ Query Object: DQL API es actualmente una extensión de la idea básica de Query Object pattern.

(phpDoctrine, 2007)

1.5.3 Plataformas de propósito general.

1.5.3.1 CodeIgniter.

CodeIgniter es un buen MT, utilizado por una gran comunidad de usuarios. Construido para codificadores PHP que necesitan una herramienta de desarrollo fácil para crear aplicaciones web simples y elegantes. Entre sus características podemos encontrar su compatibilidad con PHP4 y PHP5, incorpora el patrón MVC, soporte para múltiples bases de datos, plantillas, validaciones, no requiere instalación, podemos encontrar una librería con un gran número de clases.

1.5.3.2 CakePHP.

CakePHP es un MT similar a CodeIgniter de desarrollo rápido. Es una estructura de librerías y clases para programar aplicaciones web. Su base es el Framework de Ruby on Rails. Brinda la posibilidad de interactuar con las base de datos, usando ActiveRecord. Incorpora el patrón MVC, compatible con PHP4 y PHP5, URLs amigables, Soporta AJAX, incluye tratamiento de excepciones y validación.

1.5.3.3 Symfony.

Diseñado con el objetivo de optimizar la creación de las aplicaciones web, con el uso de sus características. Posee una librería de clases que permiten reducir el tiempo de desarrollo de las aplicaciones. Está desarrollado en PHP5, se puede utilizar en plataformas *nix (Unix, Linux) y Windows. Requiere de una instalación, configuración y líneas de comando, incorpora el patrón MVC, soporta AJAX, plantillas y un gran número de bases de datos.

1.5.3.4 Prado.

Prado está basado en componentes y eventos con el objetivo de acelerar el desarrollo de aplicaciones web usando PHP5. El concepto del desarrollo de aplicaciones en Prado es diferente, se utilizan componentes, eventos y propiedades en vez de procedimientos, URL y parámetros. Este combina especificaciones en un archivo XML, plantillas HTML y una clase PHP. Prado, cuenta con soporte para AJAX, validación, autentificación, plantillas y múltiples bases de datos.

Entre los beneficios que podemos encontrar para el desarrollo de aplicaciones web se encuentran:

- ✓ Reutilización: los códigos y componentes pueden ser reutilizados.
- ✓ Fácil uso: la creación y uso de componentes es fácil.
- ✓ Funcionamiento: utiliza una técnica de caché para asegurar el funcionamiento de aplicaciones basadas en él.
- ✓ Integración: permite la separación del contenido y la presentación.

1.6 Contenedor Web.

1.6.1 Apache Server.

Es un servidor de páginas web de código abierto multiplataforma. Su flexible sistema modular, permite cargar y descargar módulos sin necesidad de tocar el kernel[2]. Dispone de una herramienta (APXS) que facilita la compilación e instalación de estos módulos. Los módulos se cargan en memoria cuando los necesita y se descargan automáticamente cuando dejan de utilizarse.

El proceso de instalación de Apache variará dependiendo del tipo de sistema operativo que se emplee y de si se tiene o no, privilegios de administrador. Sus principales características se describen a continuación:

- ✓ Trabaja sobre múltiples plataformas (Unix, Linux, MacOSX, Vms, Win32, OS2, etc.).
- ✓ Incluye módulos que se cargan de forma dinámica.
- ✓ Soporta CGI, Perl, PHP.
- ✓ Soporte para Bases de datos.
- ✓ Soporte SSL para transacciones seguras.
- ✓ Incluye soporte para host virtuales.
- ✓ Soporta HTTP 1.1.
- ✓ Código Abierto.
- ✓ Rápido y eficiente.

(abcdatos, 2008)

[2] kernel: Parte fundamental de un programa, por lo general de un sistema operativo, que reside en memoria todo el tiempo y que provee los servicios básicos.

1.7 Patrones de diseño.

Un patrón describe un problema que ocurre frecuentemente en el diseño e implementación del Software y ofrece una solución consistente que puede ser rehusada. Existe un gran número de patrones de diseños agrupados fundamentalmente en tres tipos o categorías: patrones de comportamiento, patrones estructurales y de asignación de responsabilidades. (Larman, 1999)

1.7.1 Singleton.

El patrón Singleton es uno de los más sencillos patrones de diseño, y es útil para limitar el máximo número de instancias de una clase en exactamente solo una. En este caso, si más de un objeto necesita utilizar una instancia de la clase Singleton, esos objetos comparten la misma instancia de la clase Singleton. En un uso más avanzado, este patrón puede ser utilizado también para administrar instancias de una clase.

Ventajas:

- ✓ Consistencia y calidad en el código fuente.
- ✓ Reusabilidad del código fuente.
- ✓ Mejor rendimiento de las aplicaciones.
- ✓ Aplicaciones más fáciles de mantener.

Desventajas:

- ✓ Es engorroso implementar la herencia en un patrón Singleton, ya que solo funcionará si la clase base aún no ha sido instanciada.

(Larman, 1999)

1.7.2 Experto.

Experto es un patrón que se usa más que cualquier otro al asignar responsabilidades y es un principio básico que suele útil en el diseño orientado a objetos. Este asigna una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Ventajas:

- ✓ Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto provee un bajo nivel de acoplamiento, lo que favorece el hecho de tener sistemas más robustos y de fácil mantenimiento.
- ✓ El comportamiento se distribuye entre las clases que cuentan con la información requerida, lo que ayuda a definir clases "sencillas" y más cohesivas, que son más fáciles de comprender y mantener. (Larman, 1999)

1.7.3 Creador.

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento.

Ventajas:

- ✓ Se brinda soporte a un bajo acoplamiento, lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización.
- ✓ Es probable que el acoplamiento no aumente, pues la clase creada tiende a ser visible a la clase creador.

(Larman, 1999)

1.8 Arquitectura de software.

La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad. La Arquitectura de Software establece los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades. (Iósev, 2007)

1.8.1 Cliente- Servidor.

La arquitectura cliente/servidor es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes.

En este modelo las aplicaciones se dividen de forma que el servidor contiene la parte que debe ser compartida por varios usuarios, y en el cliente permanece sólo lo particular de cada usuario.

La parte cliente realiza generalmente funciones como:

- ✓ Manejo de la interfaz de usuario.
- ✓ Captura y validación de los datos de entrada.
- ✓ Generación de consultas e informes sobre las bases de datos.

Por su parte la servidora realiza, entre otras, las siguientes funciones:

- ✓ Gestión de periféricos compartidos.
- ✓ Control de accesos concurrentes a bases de datos compartidas.

- ✓ Enlaces de comunicaciones con otras redes de área local o extensa.

Siempre que un cliente requiere un servicio lo solicita al servidor correspondiente y este le responde proporcionándole la información requerida. Normalmente, pero no necesariamente, el cliente y el servidor están ubicados en distintos procesadores. Los clientes se suelen situar en ordenadores personales y/o estaciones de trabajo y los servidores en procesadores departamentales o de grupo.

Entres sus principales características se pueden destacar:

- ✓ El servidor presenta a todos sus clientes una interfaz única y bien definida.
- ✓ El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- ✓ El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- ✓ Los cambios en el servidor implican pocos o ningún cambio en el cliente.

1.8.2 Modelo-Vista-Controlador (MVC).

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el controlador es el Sistema de Gestión de Base de Datos y el modelo es el modelo de datos.

- **Modelo**: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no permitiendo comprar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o portes en un carrito de la compra.
- **Vista**: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- **Controlador**: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

(bonnerix, 2007)

Ventajas:

- ✓ La separación del modelo de la vista, separar los datos de la representación visual de los mismo.
- ✓ Es mucho más sencillo agregar múltiples representaciones de los mismos datos o información.
- ✓ Facilita agregar nuevos tipos de datos según sea requerido por la aplicación ya que son independientes exclusivamente de otras capas.
- ✓ Crea independencia de funcionamiento.
- ✓ Facilita el mantenimiento en caso de errores.
- ✓ Ofrece maneras más sencillas para probar el correcto funcionamiento del sistema.
- ✓ Permite el escalamiento de la aplicación en caso de ser requerido.

Desventajas:

- ✓ La separación de conceptos en capas agrega complejidad al sistema.
- ✓ La cantidad de archivos a mantener y desarrollar se incrementa considerablemente.
- ✓ La curva de aprendizaje del patrón de diseño es más alta que usando otros modelos más sencillos.

1.8.3 Arquitectura en capas.

La arquitectura de una aplicación es la vista conceptual de la estructura de esta. Toda aplicación contiene código de presentación, código de procesamiento de datos y código de almacenamiento de datos. La arquitectura de las aplicaciones difiere según como esta distribuido este código. En el diseño de sistemas informáticos actual se suele usar las arquitecturas multicapas o Programación por capas. En dichas arquitecturas a cada capa se le confía una misión simple, lo que permite el diseño de arquitecturas escalables.

El diseño más utilizado actualmente es el diseño en tres capas como comúnmente se le suelen llamar:

- ➢ **Capa de presentación:** la capa de presentación representa la parte del sistema con la que interactúa el usuario. En una aplicación Web, un navegador puede utilizarse como cliente del sistema, pero esta no es la única posibilidad, también puede generarse una aplicación que cumpla las funciones de un cliente “ligero” para interactuar con el usuario.

- **Capa de negocio:** el comportamiento de la aplicación es definido por los componentes que modelan la lógica de negocio. Estos componentes reciben las acciones a realizar a través de la capa de presentación, y llevan a cabo las tareas necesarias utilizando la capa de datos para manipular la información del sistema. Tener la lógica de negocio separada del resto del sistema también permite una integración más sencilla y eficaz con sistemas externos.

- **Capa de acceso a datos:** sus funciones incluyen el almacenamiento, la actualización y la consulta de todos los datos contenidos en el sistema. En la práctica, esta capa es esencialmente un servidor de bases de datos aunque podría ser cualquier otra fuente de información.

Ventajas:

- ✓ La ventaja de una arquitectura n-capas comparado con una arquitectura de dos niveles es que separa hacia fuera el proceso, eso ocurre para mejorar el balance de la carga en los diversos servidores; es más escalable.

- ✓ Todos los datos se almacenan en los servidores, así que tienen mayor capacidad del control de la seguridad, el servidor puede controlar el acceso y el recurso al cerciorarse que dejó solamente esos accesos de usuarios permitidos y cambiar datos.

- ✓ Es más flexible que el paradigma del P2P[3] (alternativa para interconectar enormes cantidades de recursos distribuidos de forma descentralizada, es decir, sin ningún sistema de control central) para poner al día los datos u otros recursos.

- ✓ Cualquier elemento de la red Cliente-Servidor puede ser aumentado.

Desventajas:

- ✓ La congestión del tráfico ha sido siempre un problema desde el primer día del nacimiento del paradigma de Cliente-Servidor. Cuando una gran cantidad de clientes envían peticiones al mismo servidor al mismo tiempo, puede ser que cause muchos problemas para el servidor. Mientras más clientes hay, más problemas tiene.

[3] P2P: acrónimo de peer-to-peer, designa la modalidad de compartir información entre iguales, así como el software que facilita tales intercambios.

- ✓ El paradigma de Cliente-Servidor no tiene la buena robustez de una red P2P. Cuando el servidor está caído, las peticiones de los clientes no pueden ser satisfechas.
- ✓ El software y el hardware de un servidor son generalmente muy determinantes. Un hardware regular de un ordenador personal puede no poder servir a cierta cantidad de clientes. Normalmente se necesita software y hardware específico para satisfacer el trabajo.

(slideshare, 2008)

1.8.4 Arquitectura Orientada a Servicios (SOA).

La arquitectura **SOA (Service Oriented Architecture)** propone un modelo mucho más eficiente, en el que el código de la función es independiente de la forma en que se resuelve la integración. La función puede estar hecha en cualquier lenguaje de programación y residir en cualquier tipo de plataforma tecnológica conservándose de esta manera los activos actuales de la empresa en sus sistemas de información. Desde el punto de vista de las aplicaciones externas, la función es una caja negra que recibe unos parámetros de llamada o solicitud de información y responde de una manera que es reconocible según unos estándares. La integración se resuelve mediante una buena definición de los parámetros de llamada a la función y una buena definición de la naturaleza de la respuesta. Esta definición se hace mediante los estándares que engloban los Servicios Web y que describimos brevemente.

- ✓ **XML** (Extensible Markup Language): es un estándar para definir los nombres y propiedades de los ítems de datos que se intercambian entre el servicio solicitante y el servicio proveedor.
- ✓ **SOAP** (Simple Object Access Protocol): es un conjunto de normas que define como deben ser las solicitudes y respuestas entre un usuario Web y un Servicio Web.
- ✓ **WSDL** (Web Services Description Language): describe pormenorizadamente la funcionalidad de un Servicio Web y los protocolos necesarios, tales como SOAP, para interactuar con el mismo.
- ✓ **UDDI** (Universal Description Discovery and Integration): es un registro que permite al proveedor de un Servicio Web, anunciarlo y describirlo para que otros usuarios lo puedan encontrar y hacer uso de él. (neurored, 2008)

1.8.5 Fundamentación de la selección de la arquitectura a utilizar.

Se decidió utilizar una arquitectura en capas ya que es una tecnología de avanzada, se integra a cualquier plataforma sin importar el lenguaje de programación, cumple de manera eficiente con el patrón de bajo acoplamiento y tiene la ventaja de que es flexible (ya que la estación solo necesita transferir parámetros a la capa intermedia) y escalable (ya que realiza un balance de la carga de los servidores). La separación de roles en una arquitectura en capas hace más fácil reemplazar o modificar una capa sin afectar a los módulos restantes. El código de la capa intermedia puede ser reutilizado por múltiples aplicaciones si esta diseñado en formato modular. Con la arquitectura en capas, la interfaz del cliente no es requerida para comprender o comunicarse con el receptor de los datos, por lo tanto, esa estructura de los datos puede ser modificada sin cambiar la interfaz del usuario en la PC.

SOA tiene un modelo mucho más eficiente que los anteriores, puede ser utilizado en cualquiera de las plataformas tecnológicas existentes, las funciones que utilizan estos servicios pueden estar implementadas en cualquier lenguaje de programación pero actualmente no se cuenta con un buen equipamiento de servicios web para hacer una correcta utilización de esta arquitectura, además de que se requiere de mucho tiempo para establecer una arquitectura solida de este tipo, pero no se deja de reconocer que SOA es la arquitectura más completa que existe en la actualidad.

1.9 Arquitectura de Software de Dominio Específico.

Una Arquitectura de Software de Dominio Específico (DSSA) se define como: “un conjunto de componentes de software, especializados para un tipo de tarea (de dominio), para el buen uso generalizado a través de ese dominio, integrados en una estructura normalizada (topología) eficaz para la creación de aplicaciones con éxito “o, alternativamente" un contexto de las pautas del consumo problemático de los elementos, elementos de solución, y de las situaciones que definen mapeos entre ellos”.

El objetivo central de una DSSA es fomentar en gran medida la reutilización de software, enfocándose en los problemas específicos de dominios de software y desarrollando soluciones basadas en componentes para estos dominios.

1.9.1 Definición de Dominio.

Un dominio es un ámbito de interés que se centra en un espacio del problema representando un subconjunto dentro de una o varias ramas. Un ejemplo de dominio puede ser el contexto de las aplicaciones web sobre PHP que utilicen una arquitectura en capas, Zend Framework en la capa de negocio y Doctrine ORM para la gestión de información en una base de datos. Cuando se construye una aplicación, se está desarrollando software para un espacio del problema, en fin, es un conjunto de problemas a ser resueltos dentro de una o varias disciplinas.

DARPA (Agencia de Investigación de Proyectos Avanzados de Defensa) definió cuatro pasos claves para la creación de una DSSA:

- ✓ Un modelo del dominio es desarrollado para establecer una terminología estándar y una semántica común para el dominio del problema.
- ✓ Un conjunto de requerimientos de referencia para todas las aplicaciones dentro del dominio del problema que es desarrollado.
- ✓ Una arquitectura de referencia es definida para una solución genérica, estandarizada y basada en componentes del sistema para anticipar los requerimientos del cliente.
- ✓ Nuevas aplicaciones son desarrolladas para determinar requerimientos específicos de la aplicación, seleccionar o generar componentes particulares y ensamblarlos acorde a la arquitectura de referencia.

(Iósev, 2007)

1.9.2 Elementos de la DSSA.

Una DSSA es un proceso e infraestructura que soporta el desarrollo de un modelo del dominio, requerimientos de referencia y una arquitectura de referencia para una familia de aplicaciones dentro de un dominio del problema. (Iósev, 2007)

1.9.2.1 Modelo del dominio.

El modelo del dominio es una representación de qué ocurre en el dominio, por ejemplo, operaciones de negocio en el dominio. Específicamente, esto describe las funciones a ser ejecutadas, las entidades que las ejecutan y los datos que fluyen entre ellas y si son usados o producidos por las funciones. El objetivo del modelo del dominio es estandarizar la terminología y la semántica del espacio del problema, por ejemplo el conjunto de problemas dentro del dominio que un conjunto de aplicaciones soportarán. (Iósev, 2007)

1.9.2.2 Requerimientos de referencia.

Los requerimientos de referencia son requerimientos estándares que se derivan de las diferentes funcionalidades del conjunto de aplicaciones que pueden ser desarrolladas dentro de un dominio. Los requerimientos funcionales caracterizan la estructura funcional general del espacio del problema en que se trabaja. Muchas veces, a partir de los requerimientos de referencia que son llamados también genéricos se van refinando de acuerdo a los requisitos funcionales planteados por el cliente para la aplicación ya específica.

1.9.2.3 Arquitectura de referencia.

Una arquitectura de referencia es una arquitectura genérica, estándar, que describe a todos los sistemas dentro de un dominio determinado. Esto pudiera especificar tanto la plataforma de hardware o software, y los componentes. (Iósev, 2007)

Los elementos de una arquitectura de referencia son los siguientes:

- ✓ Un modelo de la arquitectura de referencia que describe una configuración basada en componentes de hardware y software sobre un estilo arquitectónico.
- ✓ Un esquema arquitectónico que describe la estructura de alto nivel y restricciones de cada uno de los componentes, incluyendo las principales entidades de datos.
- ✓ Un diagrama de dependencias de componentes describiendo las interacciones a través de los principales componentes.

- ✓ Un conjunto de descripciones de interfaces de componentes describiendo el API de cada uno ellos.
- ✓ Un conjunto de restricciones organizadas por parámetros, rango de valores, valores por defecto, y consecuencia sobre la arquitectura.

(Iósev, 2007)

A continuación se define una arquitectura específica a partir de la arquitectura de referencia anteriormente planteada. Se llama arquitectura específica a la definición de una arquitectura que describe todos los productos que caben dentro del alcance de una línea. Existe una relación estrecha entre la arquitectura específica de la línea de productos y la arquitectura de un producto en particular: todos los componentes definidos para el producto deben ser parte de los componentes definidos dentro de la arquitectura específica.

Los principales elementos de una arquitectura específica según se analizó, son:

- ✓ Propuesta base de diseño de las capas lógicas de la aplicación.
- ✓ Convenciones o estándares de código.
- ✓ Estructura de código y recursos.
- ✓ Mecanismos de colaboración entre los componentes.

1.9.2.4 Propuesta base de diseño de las capas lógicas de la aplicación.

Es la propuesta inicial que se utiliza como una guía para realizar el diseño de las capas lógicas. Estas capas lógicas de la aplicación pueden estar beneficiadas por patrones de diseño. La medida de esta propuesta la dan las necesidades que se vayan detectando en el proceso de desarrollo de las aplicaciones específicas para el dominio que se esté trabajando, que no las resuelva totalmente la propuesta de diseño base.

1.9.2.5 Convenciones o estándares de código.

Plantea las reglas y estructuras en las que se debe implementar el producto (código fuente), la manera en que se deben nombrar tanto las clases como las variables ganando de esta manera en organización, uniformidad y logrando además un mayor entendimiento tanto para quien desarrolle el producto como para quien haga uso del mismo.

1.9.2.6 Estructura de código y recursos.

Elabora una estructura física lo cual consiste en especificar de manera física donde van cada una de las clases de la aplicación, conformando así el armazón inicial sobre el cual se desarrollará dicha aplicación. Este esqueleto a medida en que se vaya desarrollando la aplicación o a consideración del equipo de desarrollo puede irse modificando según las necesidades encontradas.

1.9.2.7 Mecanismos de colaboración entre los componentes.

Define las posibles maneras en que interactúan los componentes que integran la aplicación, logrando de esta forma una mayor o menor dependencia entre ellos.

1.9.3 Análisis y ambientes del dominio.

El modelo del dominio es el resultado de un proceso de análisis del dominio, el cual es un aspecto esencial para diseñar los sistemas de software de alta calidad. El análisis del dominio es el proceso de analizar y entender el entorno del usuario final, por ejemplo, qué operaciones de negocio los usuarios ejecutan y en qué contexto. Dentro de la ingeniería de software, el análisis del dominio conlleva a estudiar el conocimiento base necesario para solucionar el problema del diseño del software.

(Iósev, 2007)

El análisis del dominio involucra:

- ✓ Identificar, capturar, organizar, y entender los objetos y las operaciones dentro del dominio.

- ✓ Una descripción de esos objetos y operaciones usando un vocabulario estandarizado.
- ✓ Identificar sistemas y problemas similares dentro del dominio.
- ✓ Determinar qué es reutilizable a través de los sistemas similares.

(Iósev, 2007)

Según Stephen H. Kaiser en su libro “Software Paradigms”, plantea que existen tres ambientes que en el análisis del dominio se deberían tener presentes para definir y desarrollar una aplicación. Ver figura 1.7.1.

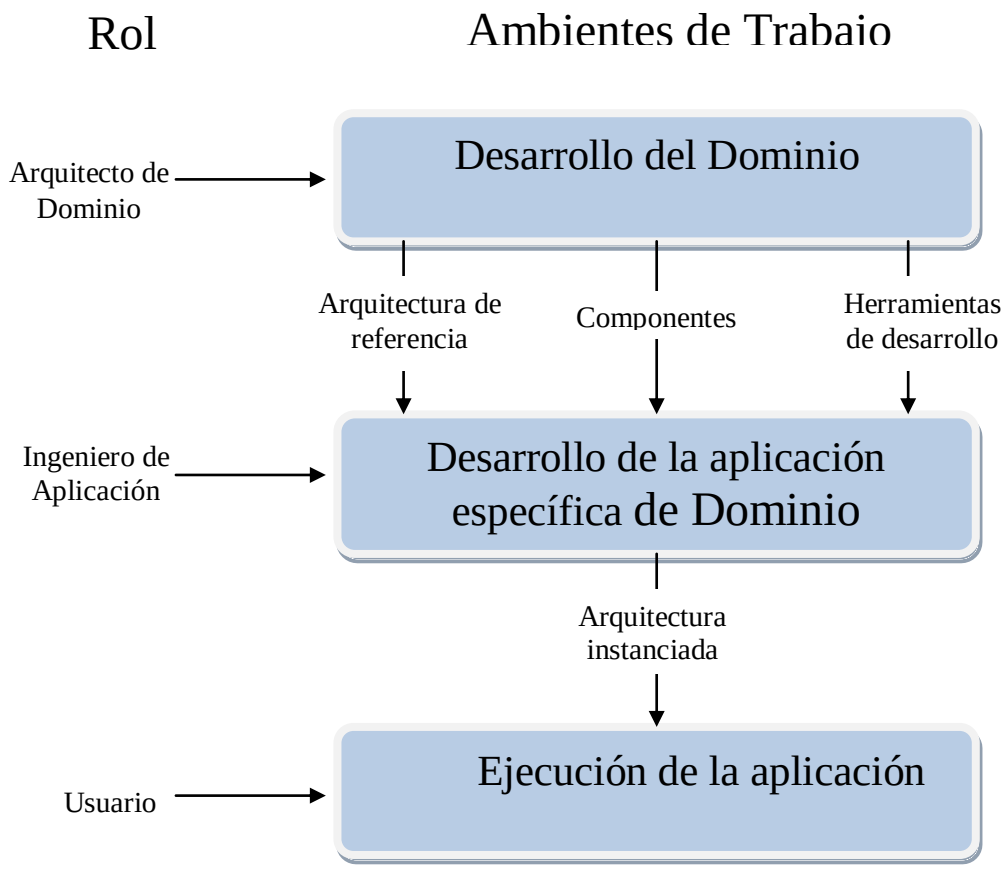


Figura 1.7.1 Los tres Ambientes de Dominio de un DSSA.

1.10 Conclusiones del capítulo.

Luego de un análisis previo de acuerdo a las tecnologías y herramientas existentes para el desarrollo de software ya sean marcos de trabajos o aplicaciones web se decidió utilizar la herramienta Visual Paradigm para el modelado de las clases ya que es multiplataforma y de software libre. Este marco de trabajo se desarrolla sobre PHP ya que es un lenguaje de código abierto y muy flexible. Se utiliza además la arquitectura en capas ya que es una tecnología de avanzada, se integra a cualquier plataforma, cumple de manera eficiente con el patrón de bajo acoplamiento y tiene la ventaja de que es flexible y escalable.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción.

Hasta el momento se han definido las características del ambiente de una DSSA. En este capítulo se utilizarán esas características y definiciones para realizar el ambiente de este DSSA que ha sido llamada AEWeb pues está descrito por el desarrollo de Aplicaciones Empresariales Web. Esta orientará una estructura y organización para el desarrollo de todas las aplicaciones de ese tipo que hagan uso de esta arquitectura.

Aquí se establecerá el dominio de la DSSA y se identificarán los requerimientos de referencia, los cuales son los requerimientos comunes para las aplicaciones desarrolladas sobre esta arquitectura específica. En la figura 2.1.1 se puede ver el ámbito en que esta enmarcado AEWeb dentro del conjunto de ambientes de dominio explicado detalladamente en el capitulo anterior, reflejándose solamente en el primero de los ambientes que es sobre el cual se va a trabajar.

También se presentará la arquitectura específica la cual esta compuesta por cuatro elementos: Definición y diseño de las capas lógicas así como la comunicación entre las mismas, convenciones o estándares de código, estructuras de código, mecanismos de colaboración entre los subsistemas y módulos específicos de las aplicaciones. De forma similar se mencionan las plataformas de trabajo que serán utilizadas en cada capa las cuales serán descritas en el capítulo siguiente.

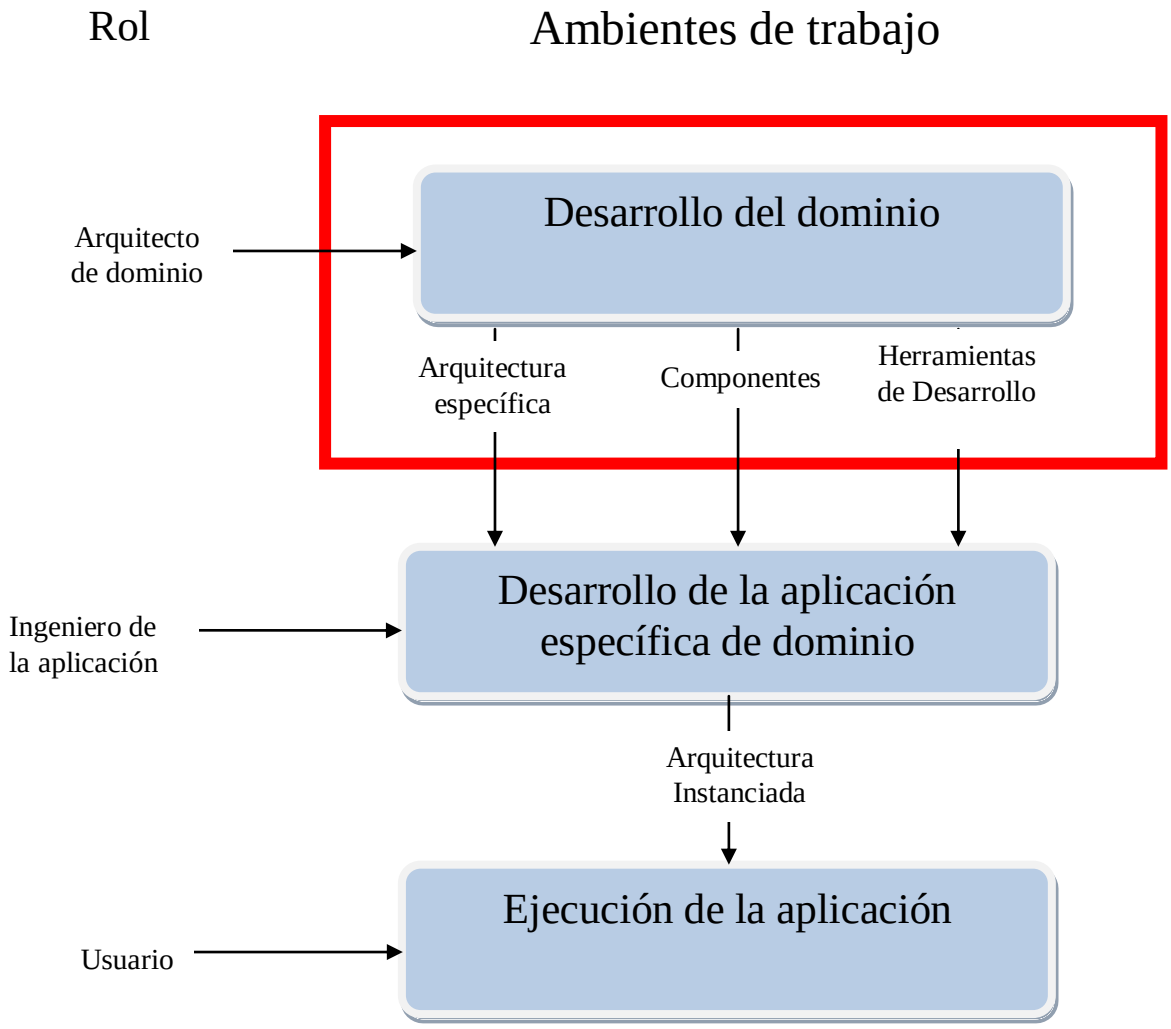


Figura 2.1.1 Ámbito de trabajo de esta arquitectura de referencia dentro del conjunto de ambientes de dominio.

2.2 Definición del Dominio.

A continuación se define el dominio en el que engloba la arquitectura específica de AEWeb a sus aplicaciones analizando e identificando los principales problemas comunes de las mismas.

Esta DSSA encierra a todas las aplicaciones que utilicen una arquitectura en capas, Ext js en la capa lógica de presentación, PHP como lenguaje de programación en la capa de lógica de negocio y

Doctrine para la persistencia de datos. También, para algunos de los requerimientos de referencia los cuales serán planteados en la sección 2.3, que presentan las aplicaciones dentro de este dominio, el Marco de Trabajo da soporte a través de un conjunto de tecnologías agilizando y optimizando el proceso de desarrollo del producto.

2.3 Requerimientos de referencia.

Existe un gran número de aplicaciones empresariales que se desarrollan en el ámbito del dominio anteriormente planteado, en todos los contextos de estas aplicaciones se hace el uso de patrones de diseño y arquitecturas lo cual se convierte en una necesidad común para problemas que en gran medida son similares. Estas necesidades repetidas es a lo que llamamos requerimientos de referencia o comunes para todas estas aplicaciones y tales requerimientos son descritos a continuación:

Almacenamiento en Base de Datos: Para guardar los datos de las aplicaciones en las bases de datos es necesario que sea de forma óptima y para ello se debe utilizar “Doctrine”.

Alto rendimiento de procesamiento de las peticiones del cliente: Las peticiones realizadas por los clientes deben ser respondidas en el menor tiempo posible por que se necesita que la comunicación entre las capas de la aplicación se haga de la manera más eficiente.

Interacción con aplicaciones externas: Es muy común la interacción con sistemas externos por lo que se hace necesario que existan estándares en el uso de las tecnologías necesarias para cumplir este requerimiento.

2.4 Arquitectura especifica.

Este Marco de trabajo presenta una arquitectura específica para realizar el diseño de las capas lógicas mediante una propuesta inicial que da organización a las estructuras de código según las convenciones y estándares de código que se elaboren quedando así conformada una estructura base y a partir de ahí hacer una propuesta de los mecanismos de colaboración entre los componentes que se integren a ella, la cual estará basada en la arquitectura en capas.

2.4.1 Diseño de las capas lógicas.

La arquitectura de las capas lógicas que el Marco de Trabajo Aplicativo propone para las aplicaciones se define en la figura 2.3.1 y a continuación se explican cada una de ellas, exponiendo también la manera en que estas se comunican.

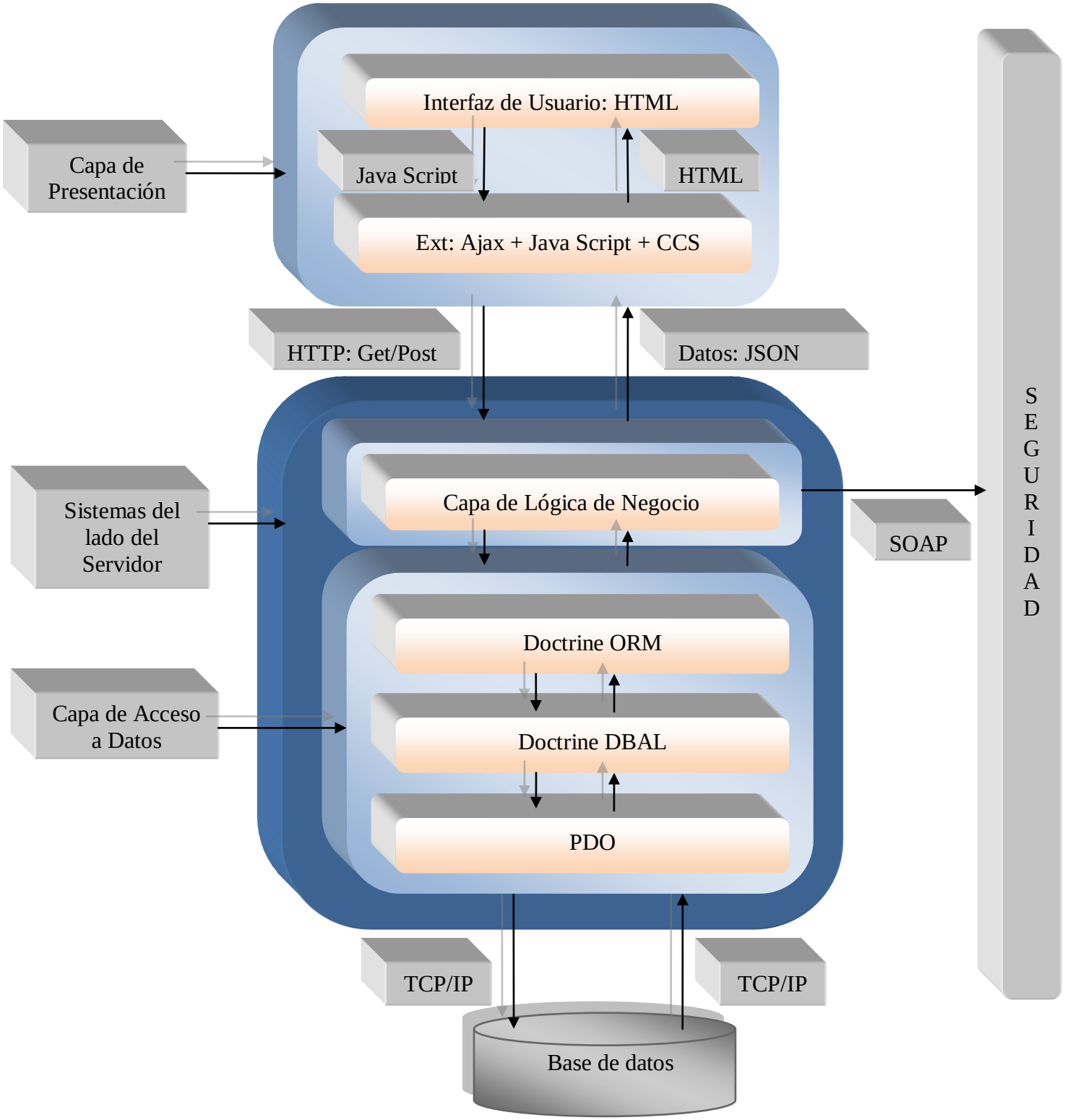


Figura 2.3.1 Arquitectura de las capas lógicas que propone AEWeb.

2.4.1.1 Capa de acceso a datos.

Siguiendo el paradigma de la Programación Orientada a Objetos (POO), el MT hace uso de ella en esta capa mediante Doctrine el cual es un ORM (Object Relational Mapper) en español Objeto Mapeador Relacional que esta diseñado para exportar una base de datos a sus clases correspondientes, es decir que para quien lo utiliza el brinda la posibilidad de trabajar con objetos que tienen métodos y en su capa de abstracción toma este código y lo convierte en sentencias que entiende cualquier sistema gestor de bases de datos(SGBD). Permite trabajar fácilmente con una base de datos. Los ORM permiten olvidarse de las particularidades de cada sistema de base de datos y permiten crear aplicaciones compatibles con cualquiera de las bases de datos más populares de forma sencilla con una Capa de abstracción a datos (DBAL) incorporado. En la figura 2.3.1.1 se detalla como esta compuesto el mismo.

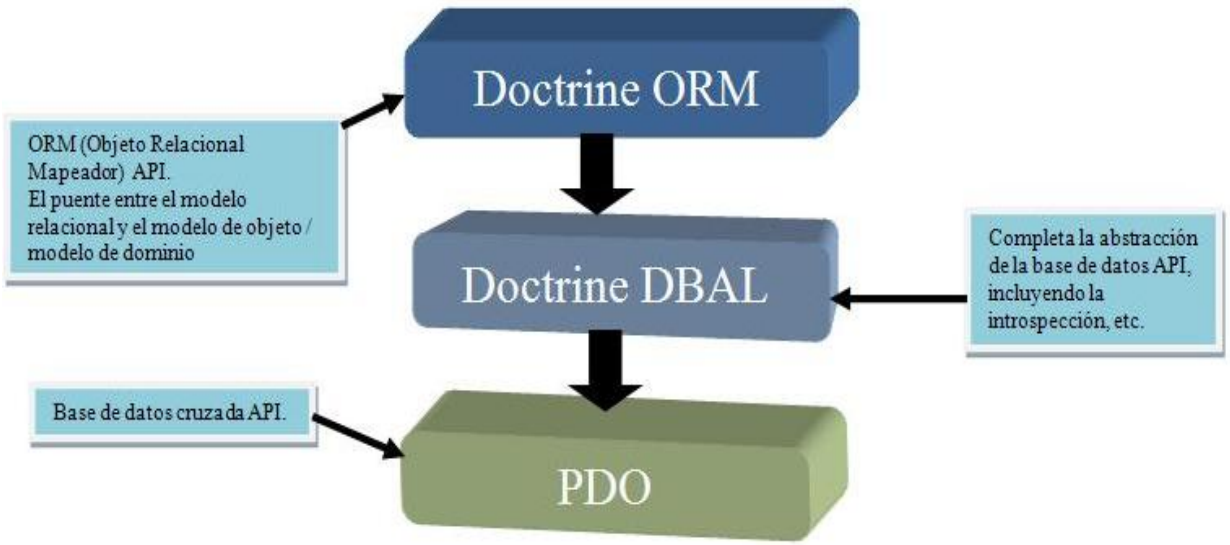


Figura 2.3.2 Estructura de Doctrine ORM.

Seguidamente se expondrán tres tipos de métodos que serán empleados en la capa de acceso a datos en Doctrine:

- ✓ Métodos de persistencia o de salva: Estos métodos hacen persistentes a los objetos que transitan por cada capa de nivel superior.
- ✓ Métodos de eliminación: Estos métodos eliminan los distintos datos persistentes que se encuentran en la mayoría de los casos en las bases de datos.
- ✓ Métodos para generar reportes o hacer conteos: Estos métodos devuelven un conjunto de resultados a partir de operaciones de consulta a la base de datos.

2.4.1.2 Capa de negocio.

En es esta capa es donde reside todo el código en PHP que se encarga de realizar las tareas que dan cumplimiento a los distintos casos de uso, además es donde se implementan las reglas del negocio. Esta capa esta compuesta por los objetos del negocio que no son más que las clases que representan a los conceptos del dominio. Todas estas clases que se implementen deben extender (heredar) de la clase abstracta **lnBase** localizada en la capa de negocio de AEWeb que será explicada en el capítulo 3. Para la creación de los servicios por parte del programador de esta capa dentro de la aplicación deberá extender de la clase **Servicio** también explicada en el siguiente capitulo, y el nombre de las clases de servicios que de aquí hereden deberán coincidir totalmente con el nombre del servicio que se necesite crear.

2.4.1.3 Capa de presentación.

Esta capa no presenta una definición de clases en php y si en java script, se basa en el uso de componentes externos como EXT JS en su versión 2.0 que sigue haciendo el uso de la POO termino que es primordial en este tema logrando proporcionar un ambiente agradable en la interfaz de las aplicaciones que se desarrollen.

Ext JS es una librería de componentes de diseño que proporciona las herramientas necesarias para la creación de aplicaciones Web con excelente calidad gráfica, pues cuenta con una amplia colección de componentes para el diseño de interfaces, ventanas, pestañas, menús, tablas y demás componentes.

Esta brinda soporte para:

- ✓ Construir interfaces gráficas complejas y dinámicas.
- ✓ Comunicar datos de forma asíncrona con el servidor.
- ✓ Manejar datos de distinta índole de una manera simple.

2.4.1.4 Comunicación entre las capas.

AEWeb soporta dos tipos de comunicación entre sus capas. El primero de ellos se hace a través de una **Notación** de **Objetos** de **Java Script** (JSON) que no es más que un formato de texto completamente independiente de cada lenguaje de programación lo cual lo hace idóneo para el envío de datos. JSON presenta dos estructuras, la primera que se conforma por una colección de pares nombre/valor que se le conocen como objetos, estructura o registro, y la segunda se configura por una lista de valores, a menudo llamada arreglo, listas o secuencias.

El segundo se hace mediante una función que reside en la capa de lógica de negocio que recibe los datos devueltos por la capa de acceso a datos y configura en paquetes de tipo **Lenguaje Extensible** de **Marcado** (XML) para ser enviados a la presentación, de esta misma manera funciona la comunicación con JSON. En la figura 2.3.1 puede observarse el esquema completo de comunicación entre las capas de AEWeb.

A continuación se explica como se logra el engranaje entre estas capas y la base de datos. Cuando el cliente hace una petición se hace una llamada a las funciones java script a la lógica de interfaz de usuario dentro de la misma capa de presentación, luego la petición se transfiere mediante el **Protocolo** de **Transferencia** de **Hipertexto** HTTP utilizando el método Post para enviarle los datos a la pagina servidora vinculada a ese caso de uso, la cual procesará la información dentro de la misma capa de negocio o si se necesita hacer la petición a la base de datos recurre a su pagina de persistencia encargada de establecer comunicación con la de la base de datos y obtener la información necesaria, esta conexión tiene lugar mediante el protocolo TCP/IP en ambos sentidos y luego la página servidora que recibe la información con formato pobre en organización, le da estructura con los formatos XML y JSON según se requiera para ser transferidos a la lógica de presentación y ser mostrados al cliente con código html.

►

2.4.2 Convenciones o estándares de código.

Con el fin de obtener un lenguaje entendible, uniforme y común para las aplicaciones que utilicen la arquitectura base definida en AEWeb se han definido un conjunto de estructuras de código y convenciones de nombres para los distintos lenguajes en las capas correspondientes de las aplicaciones. La definición de estas estructuras y convenciones se elaboran de acuerdo al tipo de funciones que tengan cada una de las clases utilizadas independientemente de la ubicación que tengan estas en las capas, al igual que las instancias que se creen de las mismas. Los parámetros de entradas presentan características similares, es decir que también se le asocia un prefijo o un sufijo que permita reconocer su tipo.

2.4.2.1 Clases de la capa de acceso a datos.

- ✓ El nombre de las clases de la capa de acceso a datos que brindan operaciones de trabajo con los mismos se escribe de forma tal que se entienda claramente el fin con el que fue creada y siempre comenzando con la primera letra en mayúscula y para la creación de sus objetos correspondientes se hace de la misma manera pero agregándole una 'o' delante para indicar que es un objeto de esa clase lo que se esta creando.

- ✓ Los atributos de la clase se escriben con la primera letra en minúscula e indicando con su nombre cual es el propósito del mismo, si el nombre es compuesto se utiliza la notación CamelCasing que consiste en escribir las segundas o terceras palabras comenzando con letra mayúscula (Ej. estaNotacion).

- ✓ Las funciones siempre se escriben comenzando con letra minúscula y dejando siempre claro cual es su propósito. Siempre que este nombre sea compuesto se emplea la notación CamelCasing.

- ✓ Las asignaciones a variables se declaran solamente una por cada línea.

- ✓ Las variables de tipo globales los nombres comienzan con la letra 'g' y siempre que sean compuestas de sigue aplicando la notación PascalCasing que no es mas que escribir todos los nombres incluso el primero en letra mayúscula (Ej. gNotacionPascalCasing).

- ✓ No se incluyen espacios en blanco desde la izquierda para la instrucción **Class**.
- ✓ Se emplean líneas en blanco antes de la definición de cada una de las funciones y entre los operadores lógicos y aritméticos logrando así una mayor claridad del código.

2.4.2.2 Clases de la capa de negocio.

- ✓ Las clases que contienen las operaciones del negocio se representa por el nombre con la primera letra en mayúscula de tal manera que con solo leerlo se reconoce el propósito de la misma y para representar su instancia se escribe el mismo nombre pero iniciado con una 'o' delante para aclarar que es un objeto de la misma.
- ✓ El nombre de los atributos de una clase comienza con la primera letra en minúscula, en caso de que sea un nombre compuesto se debe emplear la notación CamelCasing. Si es una sola palabra se mantiene todo en minúscula. Permite con sólo leerlo, reconocer el propósito del mismo dentro de la clase.
- ✓ El nombre de las funciones comienza con la primera letra en minúscula, en caso de un nombre compuesto se empleará también la notación CamelCasing.
- ✓ Las funciones de obtención y actualización de datos siempre se escriben con el prefijo '**get**' y '**set**' respectivamente.
- ✓ Los parámetros de las funciones y métodos son agrupados por el tipo de dato que referencian, es decir colocar primero los string, luego los numéricos y agrupar además con uniformidad según los valores por defecto.
- ✓ Las variables se declaran de acuerdo a su tipo, si son de referencia o no, si son de referencia se comienzan escribiendo con una 'r' delante indicando que son de ese tipo y seguidamente la primera letra del nombre de la variable se escribe con mayúscula y en el caso que no sean de referencia se escribe igual que las demás variables miembros de la clase. Cuando las variables son del tipo compuestas ya sean de referencia o no se aplica entonces la notación CamelCasing.

- ✓ Las constantes de tipo variables se escriben en su totalidad en letra mayúscula y una constante por cada línea.
- ✓ Para hacer asignaciones a variables se declara una por cada línea.
- ✓ Las variables de tipo globales los nombres comienzan con la letra ‘g’ y siempre que sean compuestas de sigue aplicando.
- ✓ En el caso de las variables estáticas se comienza la escritura con la letra ‘e’ y se procede entonces a aplicar la notación PascalCasing si se necesita.
- ✓ No se incluyen espacios en blanco desde la izquierda para las siguientes instrucciones: **Require**, **Include**, **Class**. Se toma como inicio de la página el tag PHP <?php ?>. El comentario inicial de la clase seguirá la misma filosofía de 0 espacio de identación.
- ✓ Se utilizan dos espacios en blanco desde la izquierda antes de escribir las instrucciones: **Function**, **Define**, Inicio y Fin de bloque **{}**. También así para el caso de las instrucciones **If**, **else**, **for**, **while**, **do while**, **switch**, **foreach**.
- ✓ Se utilizan hasta 5 niveles de anidación en las instrucciones de ciclos: **If**, **for**, **while**.
- ✓ Se comentarea la clase o función al inicio de la misma especificando el objetivo de la misma así como los parámetros que utiliza especificando los tipos de dato, y objetivo de estos, así como los atributos de las clases.
- ✓ Se emplean líneas en blanco antes de la definición de cada una de las funciones y entre los operadores lógicos y aritméticos logrando así una mayor claridad del código.

2.4.2.3 Clases de la capa de presentación.

- ✓ Las clases que contienen las operaciones de la presentación se representa por el nombre con la primera letra en mayúscula de tal manera que con solo leerlo se reconoce el propósito de la misma y para representar su instancia se escribe el mismo nombre pero iniciado con una ‘o’ delante para aclarar que es un objeto de la misma.

- ✓ El nombre de los atributos de la clase comenzaran con letra minúscula en caso de que sea una palabra compuesta se debe de emplear la notación CamelCasing. Permite con solo leerlo conocer el propósito del mismo en la clase.

- ✓ El nombre de las funciones comienza con la primera letra en mayúscula, en caso de un nombre compuesto se empleará también la notación CamelCasing.

- ✓ Los parámetros de las funciones y métodos son agrupados por el tipo de dato que referencian, es decir colocar primero los string, luego los numéricos y agrupar además con uniformidad según los valores por defecto.

- ✓ Las variables se declaran de acuerdo a su tipo, si son de referencia o no, si son de referencia se comienzan escribiendo con una 'r' delante indicando que son de ese tipo y seguidamente la primera letra del nombre de la variable se escribe con mayúscula y en el caso que no sean de referencia se escribe igual que las demás variables miembros de la clase. Cuando las variables son del tipo compuestas ya sean de referencia o no se aplica entonces la notación CamelCasing.

- ✓ Las constantes de tipo variables se escriben en su totalidad en letra mayúscula y una constante por cada línea.

- ✓ Para hacer asignaciones a variables se declara una por cada línea.

- ✓ Las variables de tipo globales los nombres comienzan con la letra 'g' y siempre que sean compuestas se sigue aplicando la notación PascalCasing que no es mas que escribir todos los nombres incluso el primero en letra mayúscula (Ej. gNotacionPascalCasing).

- ✓ En el caso de las variables estáticas se comienza la escritura con la letra 'e' y se procede entonces a aplicar la notación PascalCasing si se necesita.

- ✓ Se utilizan dos espacios en blanco desde la izquierda antes de escribir las instrucciones: Function, Define, Inicio y Fin de bloque {}. También así para el caso de las instrucciones If, else, for, while, do while, switch, foreach.

- ✓ Se utilizan hasta 5 niveles de anidación en las instrucciones de ciclos: If, for, while.

- ✓ Se comentarea la clase o función al inicio de la misma especificando el objetivo de la misma así como los parámetros que utiliza especificando los tipos de dato, y objetivo de estos, así como los atributos de las clases.
- ✓ Se emplean líneas en blanco antes de la definición de cada una de las funciones y entre los operadores lógicos y aritméticos logrando así una mayor claridad del código.

2.4.3 Recursos.

- ✓ Las páginas que contienen el código java script tienen la extensión '.js'.
- ✓ Las páginas HTML que son recursos estáticos contienes la extensión '.html'.
- ✓ Las páginas HTML que serán mapeadas por las URLs se crean con la extensión '.htm'.
- ✓ Las páginas donde se inserta el código php contienen la extensión '.php'.
- ✓ Las imágenes que son estáticas y que serán cargadas en la aplicación tienen el formato '.jpg' y las imágenes que se cargan de manera dinámica tienen el formato '.jpeg'.

2.4.4 Estructuras de código.

Considerando que AEWeb contiene a todas las aplicaciones que utilicen PHP5 como lenguaje del lado del servidor, Doctrine ORM para la persistencia de los datos y Ext js 2.0 en la presentación se establece que todas las aplicaciones que hagan uso de esta como arquitectura base tienen que regirse por las especificaciones estándares definidas por los componentes que en ella se utilizan. AEWeb establece una estructura que facilita la organización de las clases de acuerdo a su tipo y los componentes que vayan a ser utilizados en el desarrollo de las aplicaciones.

Estructura de las aplicaciones en unidades organizacionales

Con el propósito de dar organización a todas las aplicaciones o subsistemas que se desarrollen a partir de esta arquitectura base, se crea un paquete con el nombre de cada unidad organizacional. Este

paquete de la aplicación contiene a su vez un paquete para la configuración de conexiones, uno para la declaración e implementación de las clases, contiene además un paquete por cada modulo de la aplicación teniendo en cuenta que estos pueden tener submódulos los cuales también se destinan paquetes y es en estos dos últimos donde se engloban el conjunto de funciones que conllevaran a la solución de cada aplicación que se desarrolle. La aplicación encierra además otro paquete llamado js y aquí se agrupan todas las funciones escritas en el lenguaje java script para la validación de datos. Igual sucede con las imágenes, las css y las plantillas. En las librerías se incluyen los siguientes cinco paquetes: Acceso a Datos, Negocio, Presentación, Seguridad y Auditoría. En las tres primeras capas se guardan las clases y funciones destinadas correspondientemente al desempeño de cada papel y se hace con el objetivo de no ser cambiadas por el programador de la aplicación, simplemente ser llamadas desde las clases definida por este dentro de su aplicación. El uso de las distintas aplicaciones, sus módulos y submódulos se hace de acuerdo a la complejidad que presente la misma teniendo en cuenta los niveles de complejidad definidas en AEWeb.

Clasificación de la complejidad de la estructura organizacional

AEWeb define tres clasificaciones o niveles de complejidad en dependencia de la envergadura estructural del proyecto a realizar: Baja, Media y Alta. Por cada una se define una estructura de paquetes que corresponde a los tipos de sistemas posibles que se necesiten elaborar.

Complejidad Baja

Este tipo complejidad se define para satisfacer la necesidad de crear una aplicación estructuralmente pequeña que cuente a lo sumo con un solo módulo. Este tipo de aplicaciones se conforma del paquete o directorio raíz y el módulo en cuestión.

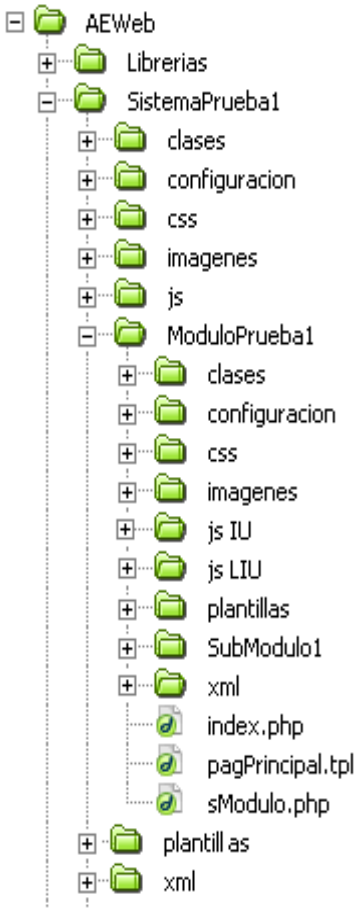


Figura 2.3.4.1 Representación de la complejidad baja.

Complejidad Media

Da respuesta a la estructura de una aplicación que cuenta con un conjunto de módulos dentro de la misma, estructurándose así el paquete raíz que engloba el conjunto de módulos y submódulos por cada uno de ellos.

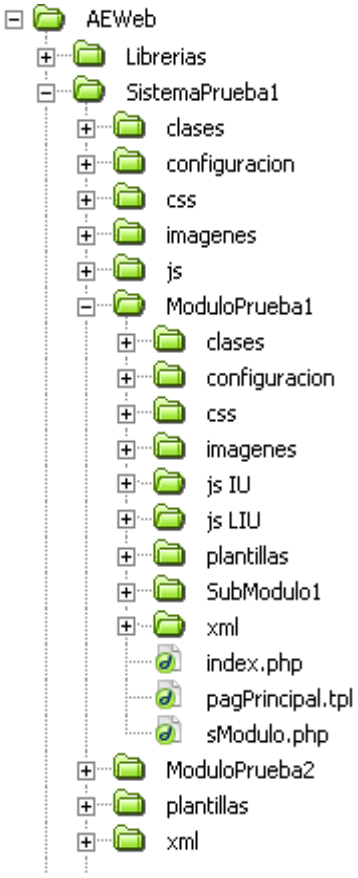


Figura 2.3.4.2 Representación de la complejidad media.

Complejidad Alta

Está definida para la elaboración de dos o más sistemas con módulos dentro de estos y además permite la inclusión de varios niveles organizacionales que son definidos por el equipo de arquitectura. Esta estructura se nutre del paquete raíz seguido por los paquetes de los sistemas que se definan y dentro de estos se encuentran sus módulos y sus submódulos correspondientes.

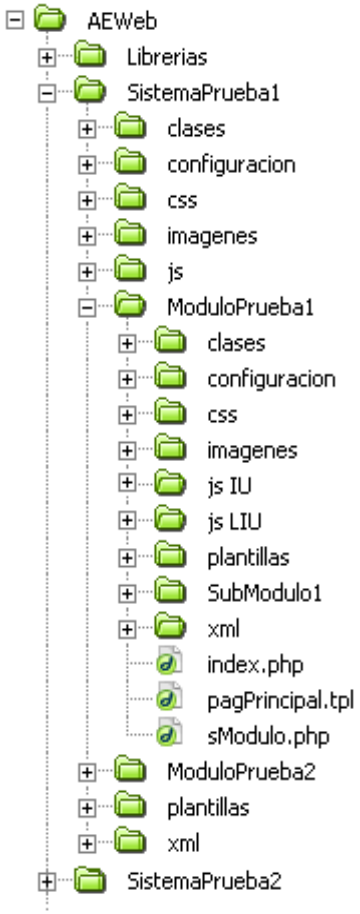


Figura 2.3.4.3 Representación de la complejidad alta.

Estructura más pequeña en la organización por paquetes

La unidad más pequeña en la estructuración de paquetes que se define en AEWeb como referencia es el submódulo. En un submódulo generalmente existen todas las capas lógicas que se definen en la arquitectura base propuesta en AEWeb. Cada componente que se desarrolle en un submódulo tiene un lugar definido en esta estructura de paquete acorde a la complejidad de la aplicación. En la figura 2.3.4.4 se puede observar dicha estructura en concordancia con los niveles de complejidad ya declarados.

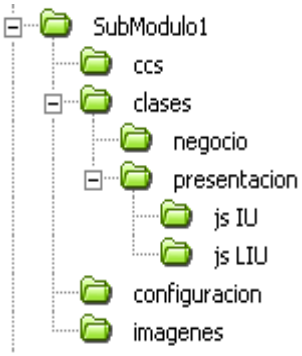


Figura 2.3.4.4 Esquema de representación de la estructura más pequeña en organización de paquetes de AEWeb.

clases: Se localizan todas las clases pertenecientes al submódulo divididas en paquetes.

clases.negocio: Se ubican implementadas las clases referentes a la lógica del negocio del submódulo.

clases.presentacion: Se encuentran ubicadas tanto las funciones de java script para las validaciones de la interfaz de usuario como los objetos que serán presentados en la interfaz del submódulo.

presentacion.js IU: Se localizan todos los objetos que serán mostrados en la interfaz del usuario para ese submódulo.

presentacion.js LIU: Están localizadas todas las funciones de java script para validaciones de los datos entrados por el usuario.

configuracion: Se localizan todos los archivos correspondientes a la configuración y para los ficheros de propiedades del submódulo.

ccs: Se encuentran agrupadas todas la hojas de estilo en cascadas de este submódulo.

imagenes: Se concentran todas las imágenes que serán empleadas en el submódulo.

2.4.5 Mecanismos de colaboración.

Los mecanismos de colaboración son estrategias que se definen en cualquiera de las arquitecturas existentes para establecer comunicación entre los componentes utilizados para la construcción del producto. AEWeb como arquitectura y según las estructuras de código explicadas en la sección 2.3.4 para dividir cada aplicación en estructuras físicas y teniendo en cuenta además los tipos de complejidades que se han definido, también propone sus mecanismos de colaboración.

Como se explicó anteriormente, una aplicación contiene uno o varios módulos y estos a su vez encapsulan submódulos. Debido a la contención de submódulos dentro de sus progenitores se crea una dependencia obligatoria entre estos, es decir que los submódulos dependen probablemente de funcionalidades de sus padres. De manera similar sucede entre módulos que se ubican en el mismo nivel, pero a diferencia de que no se refleja de manera obligatoria, es decir que puede o no suceder.

Dependencia Directa

La dependencia directa se basa en la necesidad que tiene un módulo de utilizar de manera directa una o varias interfaces y funcionalidades que se encuentran implementadas dentro otro módulo. Por ejemplo, si se necesita que en la presentación de varios módulos se cargue información de la base de datos para llenar los combobox pertenecientes a estas, pudiera hacerse un modulo con una funcionalidad que gestione dicha información y luego los módulos necesitados hacen uso de esta mostrándolos en sus respectivos componentes en la capa de presentación. En la figura 2.3.5.1 puede apreciarse con mayor claridad esta dependencia.

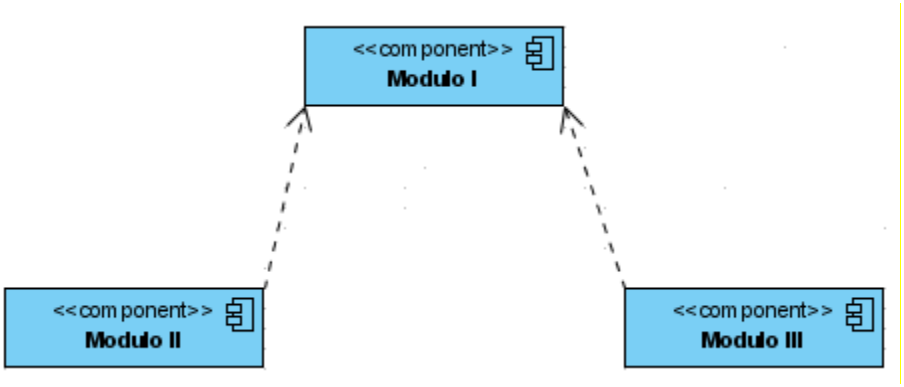


Figura 2.3.5.1 Representación de la dependencia directa entre varios módulos.

Dependencia Indirecta

Este tipo de dependencia establece la existencia de un módulo que dependa de una o varias interfaces y funcionalidades de otro módulo a su mismo nivel dentro de la estructura y que éste a su vez necesite utilizar de otras interfaces y funcionalidades de un tercer modulo, es decir que el primero de ellos depende del último tomando como intermediario al segundo de manera directa. Por ejemplo en la figura 2.3.5.2 los módulos III y IV dependen indirectamente del módulo I a través del módulo II.

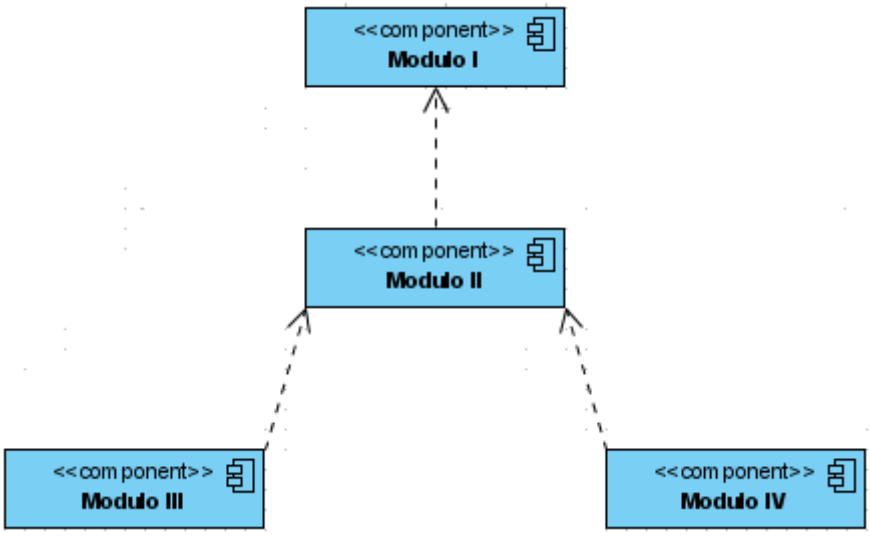


Figura 2.3.5.2 Representación de la dependencia indirecta entre varios módulos.

2.5 Conclusiones.

Con el desarrollo del presente capítulo se definieron los aspectos concretos de la arquitectura de AEWeb como es el dominio en que se enmarca la misma, los requerimientos de referencia del propio dominio, la propuesta de la arquitectura base en la cual se va a trabajar permitiendo así elaborar el diseño de las capas lógicas de las aplicaciones. Además se han establecidos convenciones de nombres y estándares de codificación para lograr una mayor organización y legibilidad tanto en el código fuente como en código de las aplicaciones que surjan a partir de AEWeb y en correspondencia con esta se definieron las diferentes estructuras de código de acuerdo a los tipos de complejidad que pueden presentar las aplicaciones y además se explicaron los mecanismos de colaboración entre las unidades organizacionales implicadas en AEWeb.

CAPÍTULO 3: AEWEB. FUNDAMENTOS PARA DESARROLLAR.

3.1 Introducción.

De manera más detallada se explican las características de AEWeb y se describen los componentes diseñados para dar soporte a los requerimientos de referencia elaborados en el capítulo anterior y se abordan temas como la configuración de AEWeb para cada producto de software que se necesite implementar.

A lo largo del presente capítulo se elabora un flujo de trabajo que orienta de manera minuciosa el desarrollo de las aplicaciones que surjan a partir de AEWeb. También se propone un ambiente de desarrollo en el cual se explican las herramientas y tecnologías a utilizar en el proceso de creación y desarrollo de tales aplicaciones.

3.2 AEWeb como marco de trabajo.

Hasta el momento se identificó la arquitectura de dominio específico idóneo para aplicaciones que cumplen con los requerimientos de referencia y arquitectura específica orientados por los sistemas anteriormente mencionados, que unido a las librerías que se presentarán en este capítulo conforman el marco de trabajo AEWeb. Se vio además la gran ayuda que aporta este tipo de marcos de trabajo a la reducción de una gran cantidad de trabajo, puesto que en ocasiones este grupo de aplicaciones se construyen desde cero, y permitiendo así la ganancia en organización en el proceso de desarrollo de estas y la conclusión del producto final con alto nivel de calidad.

AEWeb como marco de trabajo cumple también con esta tipología, es decir, su arquitectura se enmarca dentro de la llamada DSSA basado en Doctrine ORM para la persistencia de los datos lo cual permite utilizar todas sus características y funcionalidades al igual que la librería Ext js en la capa de presentación para la lógica de interfaz de usuario, y como lenguaje de programación del lado del servidor php 5.

AEWeb se compone de cinco importantes módulos para dar respuesta a los requerimientos planteados en la sección 2.3 del capítulo anterior, el paquete Librerías engloba a estos cinco módulos el primero de estos es el módulo Presentación, Negocio, Acceso a Datos, Seguridad y Auditoría. Los dos últimos

interactúan con aplicaciones exteriores de las que se consume servicios mediante el protocolo estándar SOAP[4].

Figura 3.2.1 Representación de los módulos de AEWeb.

El paquete Librerías funciona como el centro de AEWeb pues dentro de él están contenidos los componentes de la Ext para la presentación, que responde a la gestión de la interfaz gráfica de la aplicación ante el navegador del usuario, además de las validaciones de los datos proporcionados por el mismo, las clases de la lógica de negocio que contiene un conjunto de clases conjuntamente con sus funciones, las cuales van a ser extendidas o heredadas, teniendo estas nuevas sus particularidades que dan respuesta a la misma capa dentro de la aplicación, Doctrine ORM con sus

[4] SOAP: (Objeto Simple de Acceso mediante Protocolo) es un protocolo estándar creado por Microsoft, IBM y otros.

clases y funcionalidades para el acceso a los datos, la clase que controla la seguridad de las aplicaciones y la clase de auditoría. A continuación se describen cada uno de los módulos mencionados anteriormente.

Librerías

Este módulo contiene un conjunto de paquetes de clases desglosados por cada una de las capas, las cuales son observadas a continuación en la figura 3.2.1.

Figura 3.2.2 Representación del módulo Librerías y sus paquetes de clases.

A continuación se mencionan y explican cada unos de los paquetes y sus clases que están contenidos dentro de Librerías.

- ✓ Acceso_Datos: En esta paquete se agrupan todas las clases de Doctrine que tienen que ver con el acceso a datos pero de manera genérica.
- ✓ Negocio: En él se guardan todas las clases genéricas junto con sus funcionalidades que construyen la lógica de negocio.
- ✓ Presentación: Aquí se encuentra el componente de la Ext para las interfaces del usuario y la validación de los datos que el mismo introduzca.

- ✓ El módulo de Seguridad se basa en comprobar la seguridad a nivel de peticiones URLs y métodos. Este módulo se encuentra como aplicación externa y se le da uso por consumo de servicios web desde la capa de negocio por el protocolo SOAP. Este módulo brinda servicios de autenticación, autorización y administración de perfiles.
- ✓ El modulo de Auditoria se basa en auditar la aplicación en el momento deseado y se puede hacer en la capa web, en los métodos de la fachada entre la capa de presentación y la capa de negocio y en cualquier otro evento dentro de la aplicación que se requiera.

Presentación

En esta capa se hace uso de formularios para recoger datos entrados por un usuario y mostrarle información al mismo, estos tipos de formularios son considerados por Ext en todas sus versiones como clases y objeto puesto que hace uso del paradigma de POO. Estas clases contienen atributos, métodos privados y públicos, eventos y propiedades. A continuación se describen cada una de estos componentes.

Checkbox:

Única casilla de verificación sobre el terreno. Puede ser utilizado como un reemplazo directo para los campos tradicionales de casilla.

Figura 3.2.3.1 Vistas de un Checkbox marcado y desmarcado.

ComboBox:

El combobox puede usar cualquier tipo de almacenamiento de datos como su fuente de datos. Esto significa que sus datos pueden ser XML, JSON, arreglos o cualquier otro formato compatible. Puede ser cargado usando Ajax, a través de etiquetas de secuencia de comandos o local. Este combo local utiliza los datos de un array de java script.

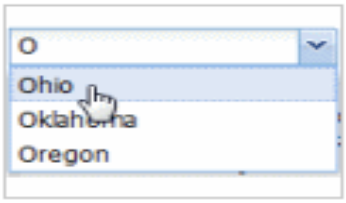


Figura 3.2.3.2 Vista de un combobox con autocompletamiento por palabras.

DateField:

Proporciona un campo de entrada fecha con un componente desplegable y fecha de validación automática.

Figura 3.2.3.3 Vista de la entrada de un dato de fecha a partir de un DateField.

TabPanel:

Se encuentra dentro de una forma y contiene uno o varios campos de entrada referentes a un mismo tema.

Figura 3.2.3.4 Vista de un TabPanel o pestañas.

HtmlEditor:

Proporciona al texto formato html de manera gráfica, es decir el color o el tipo de letra o el formato de la misma que se le pueda dar en este ambiente equivalen a escribir los tag pertenecientes al código html.

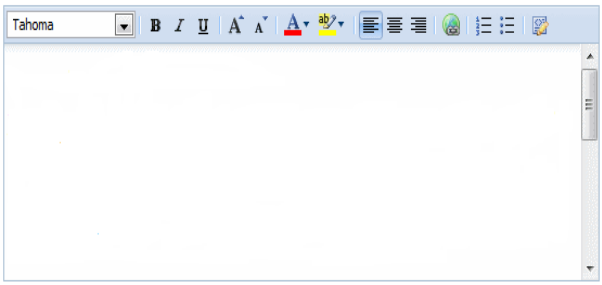


Figura 3.2.3.5 Vista de un editor de HTML.

NumberField:

Este es un tipo de campo de entrada de información de tipo numérico, el mismo tiene validaciones para que no sea entrado otro tipo de datos.

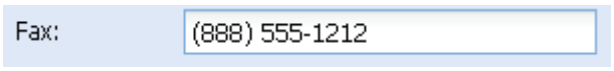


Figura 3.2.3.6 Vista de un NumberField.

Radio:

Proporciona la opción de la toma de una decisión sin posibilidad de tomar ambas a la vez.

Figura 3.2.3.7 Vistas de un Radio en sus dos opciones.

TextArea:

Permite la entrada de un texto de gran tamaño tanto de tipo numérico como alfabético, utilizado frecuentemente para que los usuarios proporcionen opiniones sobre algún tema o una sugerencia que puedan brindar.

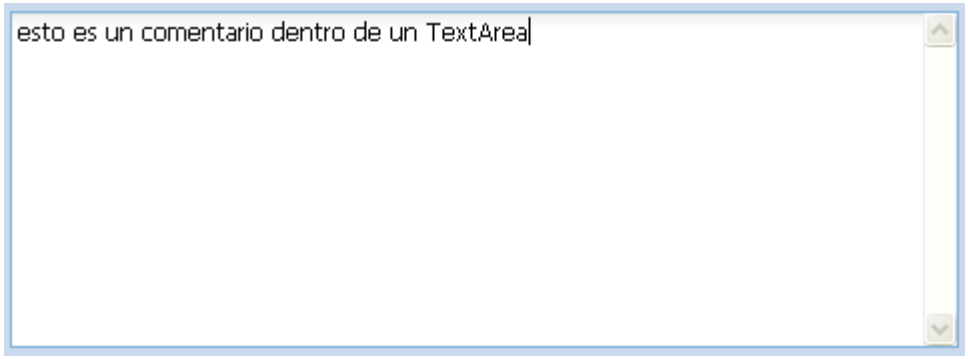


Figura 3.2.3.8 Vista de un TextArea.

TextField:

Este es un tipo de campo de entrada de información con caracteres de tipo alfabético, el mismo tiene validaciones para que no sea entrado otro tipo de datos.

Figura 3.2.3.9 Vista de un TextField para texto del tipo alfabético.

TimeField:

Proporciona un campo de entrada de la hora contando con un componente desplegable.

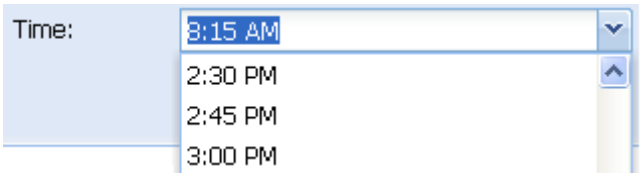


Figura 3.2.3.10 Vista de un TimeField con su componente desplegable.

GridPanel:

Este puede ser de varios tipos, puede ser de solo lectura o editable, básico o avanzado y tiene las opciones de cargar datos XML o de un arreglo.

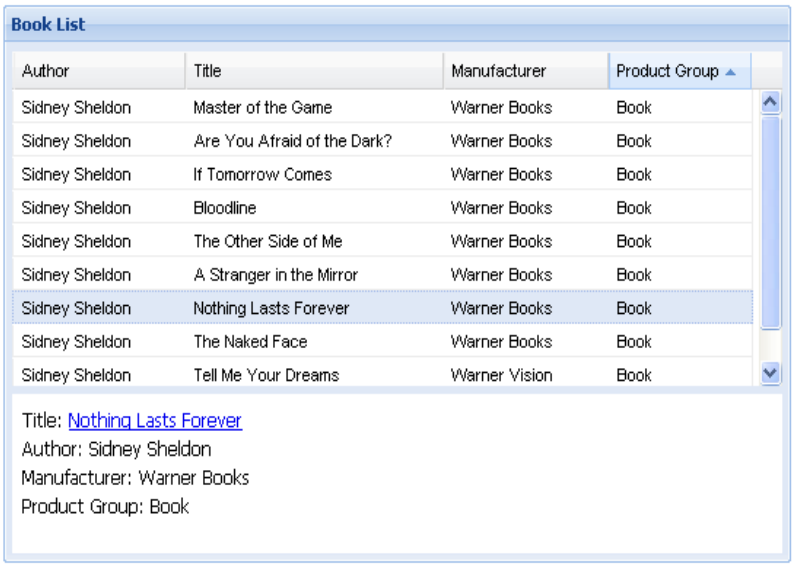


Figura 3.2.3. Vista de un grid cargado de datos XML.

Negocio

El conjunto de clases que aquí se encierran están implementadas de manera básica o general, es decir que el programador de cualquier aplicación empresarial que haga uso de esta capa nunca va a poder trabajar dentro de estas clases sino que crea un conjunto de nuevas clases particulares de él y estas heredan de algunas de las que se encuentran ya definidas en esta capa. A continuación se mencionan y describen cada una de ellas.

- ✓ **abstract class** lnBase: es la clase de la cual van a heredar las clases de lógica negocio de cada aplicación y contiene el siguiente grupo de funcionalidades. Para esta clase no se define ningún atributo.
  - **Métodos**
    - **public function** __construct (): este es el constructor de la clase el cual se inicializa nulo y no se le pasa ningún parámetro.

- **public function** get ($nombre): se le proporciona como parámetro el nombre de la variable enviada por el método GET deseada si esta está definida la retorna si no devuelve null.
- **public function** post ($nombre): se le proporciona como parámetro el nombre de la variable enviada por el método POST deseada si esta está definida la retorna si no devuelve null.
- **public function** request ($nombre): se le proporciona como parámetro el nombre de la variable enviada por el método REQUEST deseada si esta está definida la retorna si no devuelve null.
- **public function** file ($nombre): se le pasa como parámetro el nombre del archivo, verifica que el mismo este definido, en caso positivo lo retorna y encaso negativo retorna null.
- **public function** getXML ($valores): se le pasa como parámetro un resultado obtenido de la base de datos y esta se encarga de formatear los datos a un estándar xml.
- **public function** getJSON ($valores): se le pasa como parámetro un resultado obtenido de la base de datos y esta se encarga de formatear los datos a un estándar json.
- **public function** retornar ($valores, $modo = 'XML'): se le pasa como parámetros los valores a retornar y el modo en que se quiere retornar (xml o json) tomando el modo xml por defecto, esta función formatea el resultado y lo retorna.
- **abstract function** control ():

✓ **class** archivo: es la encargada del trabajo con los archivos que se suban al servidor, esta clase contiene un conjunto de atributos y métodos que seguidamente se describen.

- **Atributos**
  - **public** nombre: guarda el nombre original del archivo.
  - **public** identificador: almacena el identificador temporal del archivo.

- **public** destino: directorio por defecto donde se desea almacenar el archivo.
- **public** extensionesPermitidas: almacena una cadena de caracteres separada por coma con las extensiones permitidas para un archivo determinado.
- **public** crearDirectorio: variable booleana que nos va a decir si se va a crear una carpeta para guardar el archivo o no.
- **public** tamanno: guarda el tamaño original del archivo.
- **public** tamannoPermitido: contiene el tamaño máximo que debe tener un archivo.

- **Métodos**
  - **public function** __construct ($aDestino='', $aArregloExtensiones='', aIdentificador='', $aNombre='', $aCrearDir='', $aTamano='', $aTamanoPermitido=''): esta función representa al constructor de la clase, al cual se le pasan los parámetros con los que van a ser inicializados los atributos anteriormente descritos.
  - **public function** setNombre ($aNombre): función para cambiar el nombre del archivo y se le pasa como parámetro el nuevo nombre.
  - **public function** getNombre ():
  - **public function** permitirCambiarNombre (): esta función retorna verdadero si es posible cambiar el nombre al archivo. No se le proporciona ningún parámetro.
  - **public function** cambiarNombre ($aNombre): encargada de cambiar el nombre a un archivo y se le pasa como parámetro el nuevo nombre.
  - **public function** permitirSubir (): encargada de valorar si es posible subir o no un archivo al servidor.
  - **public function** subir (): es la encargada de subir un archivo al servidor.

- **public function** chequearNombre ($aNombre): se le pasa como parámetro el nombre del archivo y verifica que su formato sea correcto.
- **public function** chequearDirectorio (): verifica que un directorio exista y además tenga permiso de escritura.
- **public function** eliminarArchivo (): se encarga de eliminar del servidor un archivo determinado.

✓ **class** servicio: es la clase de la cual van a heredar todas aquellas que brindaran servicios web en la capa de negocio de la aplicación. A continuación se explican sus métodos y funcionalidades.

- **Atributos**
  - **public** $nombre: almacena el nombre del servicio que se va a crear el cual tiene que coincidir con el nombre de la clase.
  - **public** $dirWsdl: contiene la dirección donde estará el wsdl del servicio.
  - **public** $clienteUddi: guarda un objeto soapclient del servicio uddi, este atributo tiene valor null por defecto.
- **Métodos**
  - **public function** __construct (): que se va a encargar de dado el nombre del servicio obtener sus datos a través del servicio uddi.
  - **public function** crearServicio (): es el método que se encarga como tal de publicar la clase como un servicio.

- ✓ **class** clienteServicio: clase utilizada para consumir de los servicios.

  - **Atributos**

    - **public** $clienteUddi: guarda un objeto soapclient del servicio uddi, este atributo tiene valor null por defecto.

    - **public** $cliente: va a ser una instancia soapclient del servicio solicitado y su valor por defecto es null.

    - **public** $objResultado: almacena el resultado del consumo de un servicio determinado y tiene como valor por defecto null.

    - **public** $nombreServicio: el nombre del servicio que se quiere consumir y además tiene null como valor por defecto.

  - **Métodos**

    - **public function** __ construct ($nombreServicio): se le pasa el nombre del servicio que se desea consumir y este se encarga de obtener los datos de dicho servicio a través del uddi.

    - **public function** __call (): encargada de ejecutar una función determinada de un servicio.

    - **public function** existeFuncion ($nombreFuncion): verifica si el servicio solicitado tiene la función deseada.

    - **public function** validarParametros ($nombreFuncion, $arrayValores): valida si los parámetros pasados a la función son correctos.

- ✓ **class** session: Clase encargada de manejar todo lo referente a las variables de sesión en las aplicaciones.

- **Métodos**

  - **public function** AlmacenarValores (): Almacena todos los valores iniciales que hacen falta cuando se inicia una aplicación.

  - **public function** Cerrar_Sesion (): Destruye la sesión.

  - **public function** Existe_Variable_Sesion ($nombre): Verifica si existe una variable de sesión dado el nombre de la misma.

  - **public function** Borrar_Variable_Session ($nombre): Elimina una variable de sesión dado el nombre de la misma.

  - **public function** Get_Valor_Variable_Sesion ($nombre): Devuelve el valor de una variable de sesión dado su nombre.

  - **public function** Crear_Variable_Session ($nombre, $valor): Crea una variable de sesión pasándole el nombre y el valor de la sesión.

✓ **class** AEWeb: Clase principal del Marco de trabajo, es la que se encarga del funcionamiento general de cada aplicación, su funcionamiento es similar a un controlador frontal.

- **Atributos**

  - **public** $auditoria= new Auditoria (): objeto de la clase Auditoria.

  - **public** $session= new Session (): objeto de la clase Session.

  - **public** $servicio= new Servicio (): objeto de la clase Servicio.

  - **public** $archivo= new Archivo (): objeto de la clase Archivo.

  - **public** $seguridad= new Seguridad (): objeto de la clase Seguridad.

  - **public** $arreglo_objetos= array (): Arreglo de los objetos que van a ser utilizados en cada una delas funcionalidades de esta clase.

- **Métodos**

  - **public function** definirConstantes (): Este método e el encargado de definir todas las constantes utilizadas en la aplicación.

  - **public function** __autoload ($nombreClase): Este método es el encargado de cargar e inicializar todas las clases del marco de trabajo.

  - **public function** cargarConfiguracion (): Encargado de cargar toda la configuración de la aplicación.

  - **public function** controlAcceso (): Este método verifica en cada momento que se pueda acceder al recurso solicitado.

Acceso_Datos

La clase **Doctrine_Record** y **Doctrine_Table** son la base entre los componentes en los cuales se sustenta la capa de negocio de cada aplicación utilizando Doctrine. A continuación se describen cada una de ellas.

Doctrine_Record es uno de los componentes más esenciales de Doctrine. Las clases que heredan de Doctrine_Record se denominan componentes y debe haber al menos una por cada tabla existente en la base de datos. La clase es una envoltura de base de datos donde se definen las filas y columnas de cada tabla junto con las relaciones específicas que establecen con otros componentes.

Doctrine_Table contiene el esquema de información que se especifica por el componente de Doctrine_Record dado. Por ejemplo, si se tiene una clase Usuario que extiende de Doctrine_Record, cada esquema de definición es convocado por un único objeto tabla que contiene la información para su uso posterior. Doctrine_Table es registrado por **Doctrine_Connection** lo que significa que se puede recuperarlas tablas de la conexión llamando a la función **getTable ()** con el apropiado nombre del componente.

Seguridad: En este paquete se localiza la clase seguridad que se encarga comprobar la seguridad de las aplicaciones constantemente a la aplicación externa de seguridad, es decir que esta clase recopila los datos necesarios para hacer una prueba y mediante el consumo de servicios a través del protocolo SOAP verifica la petición.

- ✓ **class** seguridad: clase encargada de manejar la seguridad y comprobar la misma con la aplicación de seguridad mediante servicios.
  - **Métodos**
    - **public function** autenticar ($usuario, $contrasena): método encargado de autenticar, se le pasa como parámetros el usuario y la contraseña para verificar que sean correctas.
    - **public function** obtenerFuncionalidades ($sistema, $certificado): devuelve las funcionalidades de un usuario en un sistema dados el sistema y el certificado
    - **public function** verificarAcceso ($recurso, $certificado): verifica si un usuario tiene acceso a un recurso determinado dados el recurso y el certificado.
    - **public function** verificarCertificado ($certificado): verifica si un certificado es valido pasándolo como parámetro el propio certificado.

Auditoria:

- ✓ **class** auditoria: es la clase que se encarga de interactuar con los servicios de auditoria.
  - **Métodos**
    - **public function** Regis_Audit_Aplic ($evento,$sistema,$arr,$usuario,$IP): Método encargado de auditar un evento en un sistema específico.
    - **public function** CambiarObservaciones ($re,$ar): Método encargado de darle valores reales a las variables que llegan en las observaciones de la auditoria.

El diagrama de clases que se construye de AEWeb, específicamente del paquete de Librerías se muestra en la figura 1.1 del anexo.

3.3 Flujo de trabajo empleando AEWeb.

AEWeb propone un flujo de trabajo para el desarrollo de las aplicaciones empresariales, el mismo guía y organiza las principales tareas en el desarrollo de los tres componentes o capas a partir de la arquitectura específica elaborada en el capitulo2. Para la elaboración de las capas lógicas se definen tres roles de manera que exista un gran nivel de organización en la creación del producto final. AEWeb brinda las funcionalidades necesarias para que las capas lógicas puedan ser elaboradas paralelamente y se consiga la mayor interacción posible entre los desarrolladores de cada una de ellas.

En la figura 3.3.1 se muestra un esquema con las tareas que define AEWeb.

A continuación se describen los tres roles que intervienen en el desarrollo de las aplicaciones.

Se define un rol correspondiente a cada una de las capas lógicas de la aplicación: Programador de presentación (PP), programador de lógica de negocio (PLN), programador de acceso a datos (PAD) e integrador de componentes (IC).

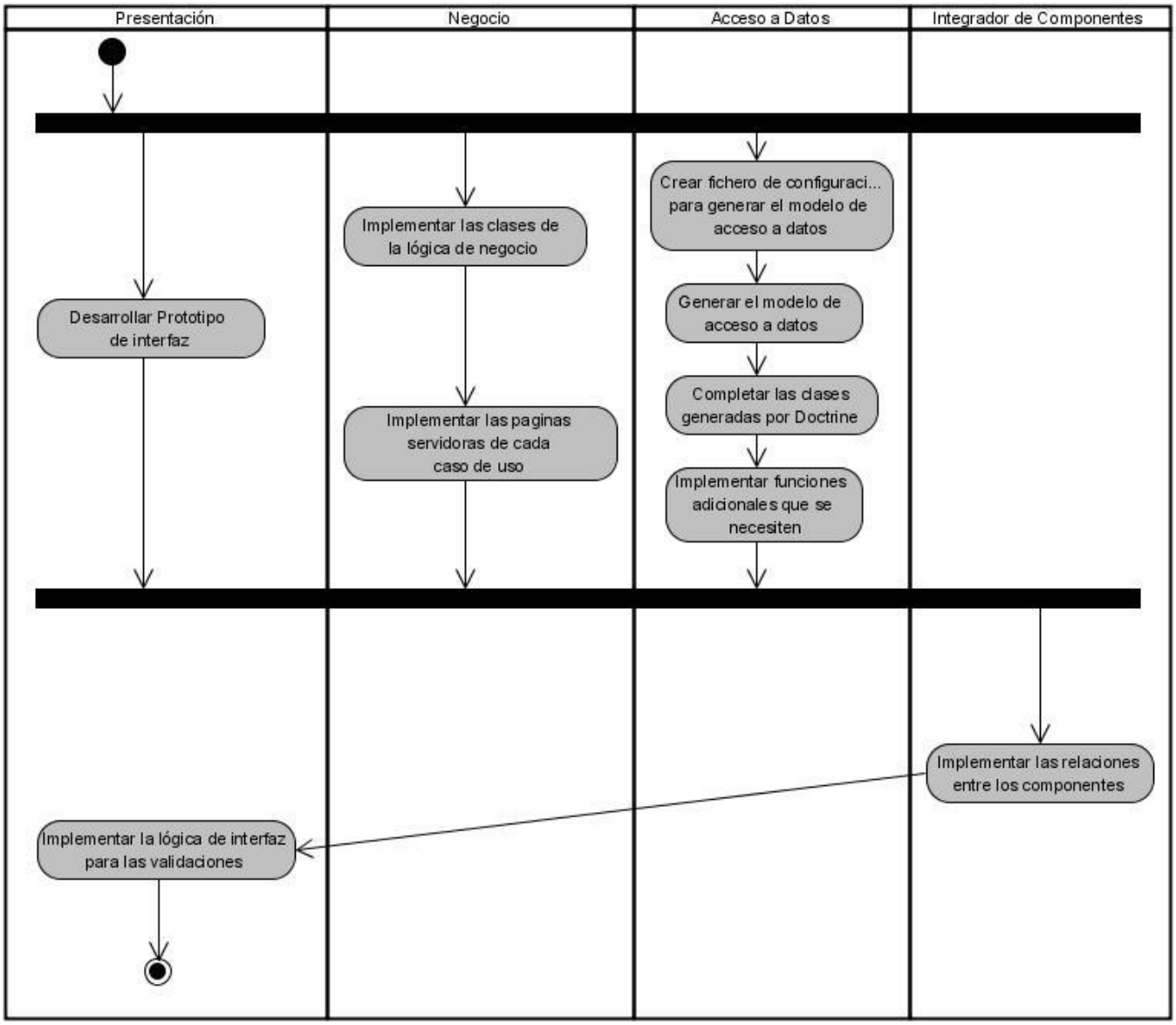


Figura 3.3.1 Esquema del flujo de trabajo.

Seguidamente se describen las actividades referentes a cada componente:

- ✓ Presentación

  - Desarrollar prototipo de interfaz: El PP elabora la estructura de las páginas de la interfaz de usuario sin definir ninguna lógica, es decir sin funcionalidad alguna.

- Implementar la lógica de interfaz para las validaciones: Una vez que el IC defina la relación entre los componentes el PP puede entonces implementar la lógica de la interfaz de usuario realizando las validaciones a los campos de datos.

✓ Negocio

- Implementar las clases de la lógica de negocio: El PLN define e implementa cada una de las clases involucradas en las reglas del negocio.

- Implementar las páginas servidoras de cada caso de uso: Luego de haber implementado las clases de la lógica de negocio, el PLN implementa las páginas servidoras correspondientes a cada caso de uso.

✓ Acceso a Datos

- Crear fichero de configuración para generar el modelo de acceso a datos: El PAD crea el fichero de configuración que no es mas que una pagina diseñada para ello y consiste en pasar los datos de la conexión, la dirección donde debe generarse el modelo de datos, el nombre de la base de datos, la tecnología de la base de datos, entre otros.

- Generar el modelo de acceso a datos: Una vez que el PAD introduce los datos referentes a la conexión y demás parámetros, procede entonces a iniciar la aplicación por el navegador y es en ese momento donde se genera el modelo de acceso a datos en la dirección antes indicada.

- Completar las clases generadas por Doctrine: Una vez que Doctrine genera las clases del modelo de acceso a datos el PAD debe completar las clases estableciendo por ejemplo las relaciones entre las demás clases.

- Implementar funciones adicionales que se necesiten: El PAD puede implementar sus funciones propias que le sirvan de ayuda para trabajar con los datos del modelo, esta tarea no se considera como obligatoria, esta en dependencia de la necesidad que tenga el PAD.

✓ Integrador de Componentes

- Implementar las relaciones entre los componentes: Luego de que el PP implementa la interfaz, el PLN termine de implementar las clases de la lógica de negocio y las paginas

servidoras por cada caso de uso y el PAD cree el fichero de generación del modelo de datos, genere el propio modelo, complete las clases generadas por Doctrine e implemente sus funciones adicionales, entonces el IC está en condiciones de implementar las relaciones entre los componentes de cada capa.

3.4 Propuesta de ambiente de desarrollo.

Para realizar la definición de la propuesta de ambiente de desarrollo para la creación de las aplicaciones web empresariales a partir de las bases de AEWeb, se analizaron en el capitulo 1 el conjunto de tecnologías y herramientas a utilizar así como sus características. A lo largo de esta sección se explicará en que parte del desarrollo de AEWeb se hace uso de estas herramientas y tecnologías. Para ello y con el objetivo de alcanzar una mayor organización de esta sección se dividió el ambiente en dos grupos: **herramientas de desarrollo** y **componentes externos**.

3.4.1 Herramientas de desarrollo.

- ✓ **Ambiente de desarrollo integrado:** Se propone como ambiente de desarrollo integrado a editor de código Zend Studio y se emplea para desarrollar la capa de negocio y la capa de acceso a datos de acuerdo a sus características y a las ventajas que el mismo proporciona.
- ✓ **Contenedor Web:** La propuesta del contenedor web para desarrollar las aplicaciones es el Apache Server de acuerdo a las características mencionadas en el análisis de las herramientas y tecnologías a utilizar ver Capitulo1.
- ✓ **Sistemas Gestores de Bases de Datos:** Se propone PostgreSQL para la realización de todas las aplicaciones que aquí se desarrollen ya que es una herramienta de software libre y por la demás características planteadas en el Capitulo 1.
- ✓ **Herramientas de modelado:** ER/Studio se propone par la elaboración de los diagramas de entidades y sus relaciones. Visual Paradigm Suite se propone como herramienta para elaborar los diagramas de clases, diagramas de casos de uso, diagramas de componentes y demás diagramas de toda la aplicación.

3.4.2 Componentes externos.

- ✓ **Componente para la capa de presentación:** Ext JS 2.0 se propone como componente externo puesto que brinda una nivel avanzado de ambiente grafico, validación de datos y alto rendimiento con el uso de Ajax y teniendo en cuenta las característica planteadas en el Capitulo1.

- ✓ **Componente para la capa de acceso a datos:** Doctrine ORM es propuesto como componente en la capa de acceso a datos teniendo en cuenta que brinda la posibilidad de hacer consultas siguiendo el paradigma de la POO y consultas en cualquier lenguaje de base de datos y contando además con sus características y beneficios que tiene el uso de este componente explicados con detalles en el capitulo1.

3.5 Conclusiones del capitulo.

En el desarrollo del presente capitulo se explicaron detalladamente las características de AEWeb como marco de trabajo describiendo además sus principales módulos a grandes rasgos y haciendo énfasis en el modulo de librerías desglosando cada uno de sus paquetes de manera detallada. Se elaboró además un diagrama del flujo de trabajos que define y da organización a las tareas dentro del proceso de desarrollo de las aplicaciones. Se realizo también una propuesta del ambiente de desarrollo donde se mencionaron cada una de las herramientas y tecnologías que se deben utilizar para la creación de las aplicaciones sobre las bases de AEWeb.

CONCLUSIONES

Con la utilización de AEWeb se logra un desarrollo de aplicaciones empresariales web con calidad al aplicar un desarrollo regido por estándares y patrones de diseño en el desarrollo de las aplicaciones sobre una arquitectura en capas.

Se logra mayor robustez, escalabilidad y reutilización del código para el desarrollo organizado y eficiente de las aplicaciones en UCID.

Se propone un flujo de trabajo general de AEWeb para guiar el desarrollo de las aplicaciones empresariales web y el beneficio de programar en las capas lógicas de forma paralela.

Se propone un ambiente de desarrollo basado en las herramientas y tecnologías con el propósito de estandarizar el uso de ellas en el desarrollo de las aplicaciones empresariales.

RECOMENDACIONES

Se recomienda:

- ✓ Continuar con el desarrollo de AEWeb ya que es una propuesta bien definida y que le puede dar solución a los problemas en la creación de aplicaciones empresariales web en UCID.
- ✓ Implementar la propuesta de AEWeb integrando Zend Framework en la capa de negocio ya que es un componente bien probado.

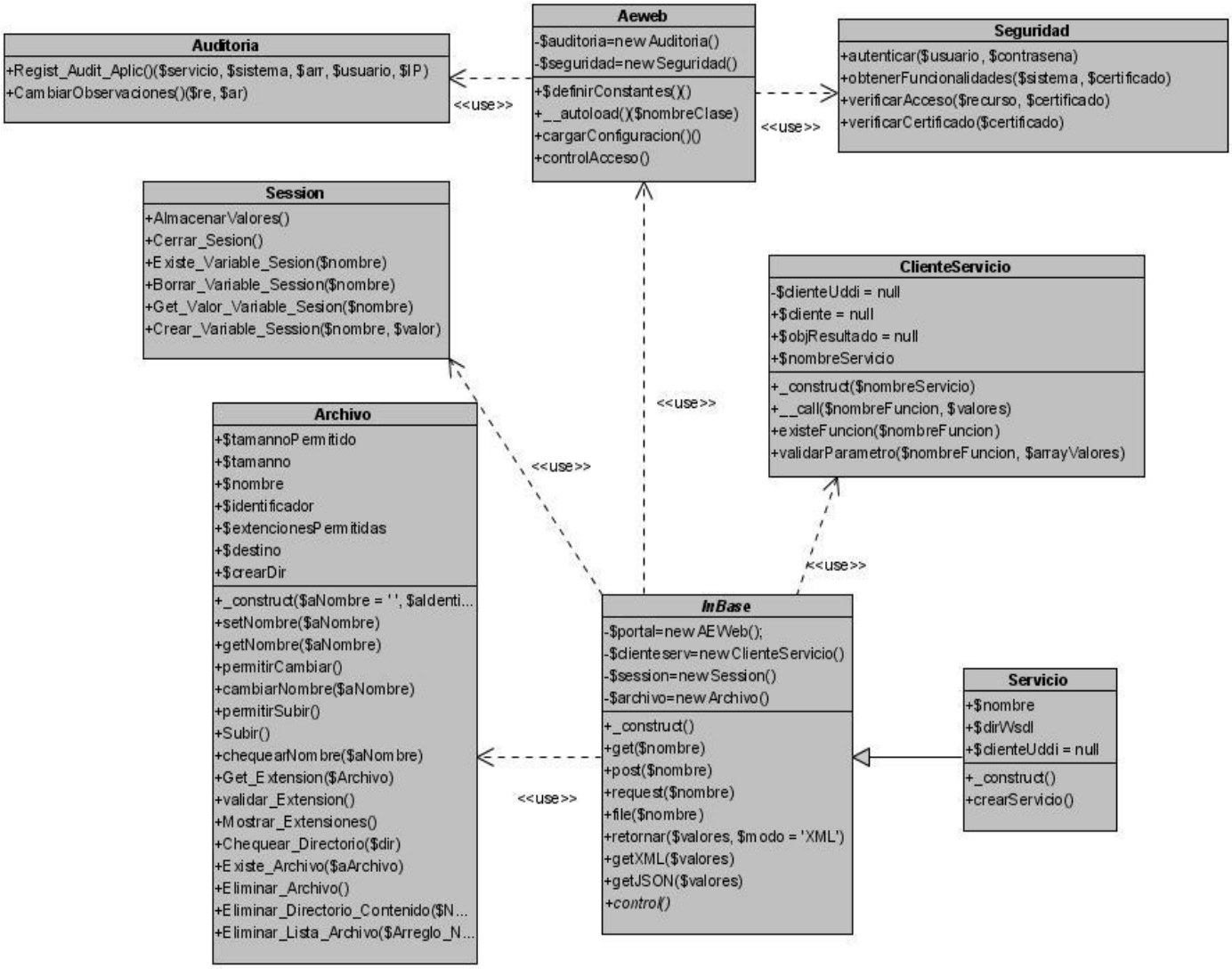
BIBLIOGRAFÍA

abcdatos. 2008. abcdatos. *abcdatos.* [Online] 2008. [Cited: abril 22, 2008.] http://www.abcdatos.com/webmasters/programa/z2818.html.
2008. adictosaltrabajo. *adictosaltrabajo.* [Online] 2008. [Cited: mayo 30, 2008.] http://adictosaltrabajo.com.
Aníbal. 2007. adelat. *adelat.* [Online] 2007. [Cited: diciembre 21, 2007.] http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html.
Aruquipa Chambi Marcelo G., Márquez Granado Edwin P. 2007. *Desarrollo de Software Basado en Componentes.* [pdf] La Paz, Bolivia : s.n., 2007.
Aruquipa Chambi Marcelo, Chavez Durán Marcelo. 2008. *Patrones de Creación.* [pdf] La Paz, Bolivia : s.n., 2008.
danysoft. 2008. danysoft. *danysoft.* [Online] 2008. [Cited: junio 1, 2008.] http://www.danysoft.info/free/model2.pdf..
2007. desarrollo en php. *desarrollo en php.* [Online] 2007. [Cited: abril 17, 2008.] http://blog.enicaragua.org.ni/roller/php/entry/crea_tu_radioblog_sin_php.
freedownloadmanage. 2007. sitio de descarga de software. *sitio de descarga de software.* [Online] 2007. http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.
2008. gestiopolis. *gestiopolis.* [Online] 2008. [Cited: enero 16, 2008.] http://www.gestiopolis.com/administracion-estrategia/estrategia/presente-y-futuro-de-los-sistemas-de-informacion.htm.
Grimán, Pérez de Ovalles,Mendoza. 2007. *Mecanismos Arquitectónicos.* [pdf] Caracas : s.n., 2007.
Guerrero. 2008. dcc. *dcc.* [Online] 2008. [Cited: junio 10, 2008.] http://www.dcc.uchile.cl/%7Eluguerre/cc40b/clase10.html.
hostingmacro. 2007. hostingmacro. *hostingmacro.* [Online] 2007. http://hostingmacro.com/glosario.html.
Iósev, Pimentel. 2007. *Arquitectura base sobre la Web.* [pdf] Ciudad de La Habana : s.n., 2007.
Javier Eguíluz Pérez. 2008. *introducción a AJAX.* 2008.
js, Ext. 2007. ext js. *ext js.* [Online] 2007. http://ext js.com.
json. 2008. json. *json.* [Online] 2008. http://www.json.org/json-es.html.
KORTH, Henry F. y SILBERSCHATZ, Abraham. 1993. *Fundamentos de bases de datos.* [pdf] Madrid : s.n., 1993.
Larman, Craig. 1999. *UML y patrones.* [pdf] 1999.
López, Pablo López. 2006. *Sistema para la gestión web del catálogo de productos de una empresa.* [pdf] Naucalpan de Juhez : s.n., 2006.
2008. maestrosdelweb. *maestrosdelweb.* [Online] 2008. [Cited: enero 17, 2008.] http://maestrosdelweb.com.
2008. manual de java. *manual de java.* [Online] 2008. [Cited: marzo 5, 2008.] http://manual-java.com/codigos-java/utilizando-patron-singleton.html.
Mercedes Fierro. 2008. mercedes. *mercedes.* [Online] 2008. [Cited: enero 16, 2008.] http://mercedesfierroc.googlepages.com/RATIONALROSE.doc.

MIGUEL, Adoración de y PIATTINI, Mario G. 1997. *Fundamentos y modelos de bases de datos.* [pdf] Madrid : s.n., 1997.
Milone, Azar, Rufiner. 2001. *Desarrollo de una supercomputadora.* [pdf] 2001.
2008. monografias. *monografias.* [Online] 2008. [Cited: enero 22, 2008.] http://monografias.com/modelodebase.html.
2006. msdn. *msdn.* [Online] 2006. [Cited: enero 15, 2008.] http://www.microsoft.com/spanish/msdn.
neurored. 2008. neurored. *neurored.* [Online] 2008. [Cited: enero 17, 2008.] http://www.neurored.com/documentos.asp.
Pablo Ravioli. 2006. monografias. *monografias.* [Online] 2006. http://www.monografias.com/trabajos7/html/html.shtml.
pampua. 2008. pampua. *pampua.* [Online] 2008. http://pampua.com/blog/index.php?date=20080206.
2007. phpDoctrine. *phpDoctrine.* [Online] 2007. [Cited: mayo 7, 2008.] http://phpdoctrine.org.
phpDoctrine. 2007. phpdoctrine. *phpdoctrine.* [Online] 2007. [Cited: mayo 7, 2008.] http://phpdoctrine.org.
2008. portalacm. *portalacm.* [Online] 2008. [Cited: enero 15, 2008.] http://portal.acm.org.
Raul Rodas Hinostroza. 2007. linuxcentro. *linuxcentro.* [Online] febrero 22, 2007. [Cited: junio 2, 2008.] http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP.
Sabja, Soto, Antillanca. 2007. *Método para transformar el diseño de una aplicación basada en la arquitectura MVC++ a una aplicación Web.* [pdf] 2007.
slideshare. 2008. slideshare. *slideshare.* [Online] 2008. [Cited: abril 12, 2008.] http://www.slideshare.net/Decimo/arquitectura-3-capas.
2007. snooks. *snooks.* [Online] marzo 18, 2007. http://snook.ca/archives/php/codeigniter_vs_cakephp/.
tringa. 2008. tringa. *tringa.* [Online] 2008. http://taringa.net/posts/info/1088860/Lo-que-tenes-que-saber-de-_ASP-(Para-Principiantes-en-Progra.html.
VALLE, JOSE GUILLERMO. 2005. monografias. *monografias.* [Online] 2005. [Cited: abril 12, 2008.] http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml#intro.
webera. 2006. webera. *webera.* [Online] 2006. http://www.htmlpoint.com/javascript/corso/js_02.htm.

ANEXOS

1.1 Diagrama de clases del paquete de Librerías.



GLOSARIO

1. **Capa de presentación**: Obligatoriamente es la capa donde residen las páginas que les mostrarán a los usuarios que interactúen con la aplicación los formularios de entrada de datos y le presentará la información procesada a manera de respuesta a una petición previamente realizada. Básicamente es donde debe residir todo el código del lenguaje de programación que entiende el navegador de cada cliente.

2. **Capa de negocio**: Es donde necesariamente se identifican las clases que procesaran los datos introducidos por los usuarios y la información obtenida de la base de datos, para darle respuesta a las peticiones de dicho usuario. En esta es donde además recae la responsabilidad de delimitar las transacciones y proveer un punto de entrada a las operaciones que se realizan sobre las aplicaciones.

3. **Capa de acceso a datos**: Dentro de esta capa residen las clases independientes de la tecnología de los sistemas gestores de bases de datos que se utilicen, a través de las cuales se realizan las operaciones sobre la información almacenada, es decir en la base de datos. Estas clases están implementadas sobre una técnica de acceso a datos que hace uso del Objeto Mapeador Relacional (ORM) Doctrine.