UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 15

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIEROS EN CIENCIAS INFORMÁTICAS

TÍTULO: Formalización y estandarización de la documentación técnica de la arquitectura tecnológica del Marco de Trabajo Sauxe versión 2.0

AUTOR: Yadira Piñera Andux

TUTOR: Ing. Marianela Tenrero Cabrera

CO-TUTOR: Ing. Noel Jesús Rivero Pino

Ciudad de la Habana, Junio de 2010

"Año 52 de la Revolución"

.

Declaro que soy la única autora de este trabajo y autorizo a La Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Yadira Piñera Andux

_____

Firma del autor

Ing. Marianela Tenrero Cabrera

_____

Firma del tutor

Ing. Noel Jesús Rivero Pino

_____

Firma del co- tutor

Tutor:

Marianela Tenrero Cabrera, Ingeniera en Ciencias Informáticas, trabajadora civil de la Unidad Militar 1722 y profesora de la facultad 15.

Co-Tutor:

Noel Jesús Rivero Pino, Ingeniero en Ciencias Informáticas, trabajador de la Subdirección de Tecnología del Centro de Informatización para la Gestión de Entidades (CEIGE).

"El éxito parece depender en gran medida de persistir en algo después de que otros han dejado de hacerlo."

William Feather

A mi mamita Enriqueta, mi tía Lucía, mi tía Ines, mis primos Arizbel, Yosbel, Yordanis, mi abuelo Román, mi mamá, por siempre tenerme como un ejemplo en la familia y confiar en mí desde el principio y demostrarme que con un poco de esfuerzo y sacrificio se pueden lograr grandes cosas como es este trabajo de diploma.

A mi novio Yuriangel, por estar conmigo en los momentos alegres y tristes de mi vida, y darme siempre su amor y apoyo incondicional.

A mis amistades del pre que no se olvidaron de mí, aunque tomáramoss caminos diferentes en la vida.

A mis amigos de la universidad que supieron aceptarme con mis defectos y malcriadeces todos estos años que pasamos juntos.

A mis tutores Marianela y Noel que supieron guiarme por el camino correcto hacia la realización completa de este trabajo de diploma. A Mailen Edith por demostrarme que no todo estaba perdido y que siempre se puede más.

Al proyecto ERP fundamentalmente a las personas del Departamento de Tecnología que han puesto su granito de arena de manera incondicional para que este trabajo de diploma se convirtiera en realidad.

A todos los profesores que contribuyeron a mi mejor formación como futura profesional y que de una forma u otra han hecho de mí una mejor persona.

Este trabajo de diploma se lo dedicó primeramente a mi bisabuela Altagracia Pelayo que aunque no pudo verme graduada, siempre confió en mí y tenía la certeza de que yo no la defraudaría y sería una gran profesional. También a mi padrino cuco, mi tía Alina y a Lucy que aunque no estén presentes aquí hoy sé que estarían alegres de verme convertida en toda una profesional.

A mi mamita querida, Enriqueta Piñera Pelayo, que con su amor y su paciencia, supo esperar todo este tiempo de estudio para verme al fin como lo que ella siempre deseo a pesar de las adversidades que se han presentado a lo largo de todos estos años.

La explícita información tecnológica del Marco de Trabajo garantiza el claro entendimiento del sistema por el cliente final, los interesados en utilizar el marco tecnológico, así como para los programadores del sistema Cedrux.

El presente trabajo de diploma pretende hacer un estudio sobre los diferentes Marcos de Trabajo basados en PHP que existen en el mundo, profundizando en ZendFramework por su facilidad para desarrollar aplicaciones web. Se abordan temas relacionados con el Marco de Trabajo: Sauxe, la librería para la capa de presentación: ExtJs, y el Mapeador de Objeto Relacional utilizado para la capa de acceso a datos: Doctrine.

Se da a conocer las principales características y actividades del modelo de desarrollo que se utiliza en el Departamento de Tecnología así como el impacto de la documentación de Sauxe tanto en la productividad del programador como en la integridad del sistema y la usabilidad del software. Además, se hace una detallada descripción de los principales escenarios y requisitos arquitectónicos de los 15 componentes de Sauxe en su versión 2.0.

Se muestra la descripción de cada uno de los componentes que integran Sauxe así como el mapa relacional de los mismos. También se aborda de forma general la arquitectura tecnológica del Marco de Trabajo obteniéndose como resultado final una documentación técnica que cumple con las necesidades del proyecto.

Palabras claves: componente, escenarios arquitectónicos, marco de trabajo, requisitos arquitectónicos.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Los sistemas de Planificación de Recursos de la Empresa (ERP), son sistemas de gestión de información que integran y automatizan muchas de las prácticas de negocio asociadas con los aspectos operativos o productivos de una empresa. Se caracterizan por estar compuestos por diferentes partes integradas en una única aplicación. Estas partes son módulos o subsistemas tales como: producción, ventas, compras, logística, contabilidad (de varios tipos), gestión de proyectos, GIS (Sistema de Información Geográfica), inventarios y control de almacenes, pedidos, nóminas, entre otros. Sólo se puede definir un ERP como la integración de todas las partes. Este tiene gran importancia ya que es una vía o alternativa para la mejora e integración de los procesos de negocio de la empresa trayendo consigo la obtención de ganancias. Las razones principales por las cuales las empresas implantan sistemas de este tipo, son: aumentar su competitividad, controlar mejor sus operaciones e integrar su información. (Iberia, 2010)

Para estimar el impacto que puede significar el hecho de contar con un verdadero sistema ERP para la economía cubana es necesario crear un equipo multidisciplinario que pueda disponer de los datos necesarios a nivel nacional, a pesar de esto se pueden identificar algunos elementos que pueden ser ilustrativos:

1. Recursos que se pierden o extravían por no tener un sistema de control efectivo.
2. El tiempo que se emplea en consolidar y relacionar datos manualmente en las organizaciones.
3. Gasto por concepto de toma de decisiones basadas en datos incorrectos o inexactos.
4. Tiempo y recursos que se requieren para hacer un levantamiento nacional de información en un sector determinado de la economía o los servicios.
5. Inversión en pagos de licencias de uso a empresas extranjeras por sistemas ERP que prácticamente nunca funcionan como tal y requieren luego de un largo proceso de “Soporte basado en Parches”.
6. Tiempo y recursos que se invierten en soporte y capacitación de los usuarios.

Este sistema tiene mucha importancia para la economía del país porque permite ahorrar cuantiosos recursos monetarios por concepto de importación de software, ya que de esta manera la nación no tendría que invertir fondos en la compra de licencias, plugins y otros softwares. De esta forma se obtiene la soberanía tecnológica pues se evita el tener que depender de otros países en la elaboración de programas o actualizaciones para poder usarlos.Todo esto trae consigo la uniformidad en el

procesamiento de la información porque todas las empresas cubanas tendrían el mismo sistema contribuyendo así al ordenamiento institucional.

Como los sistemas ERP son muy costosos, se le dio a la Universidad de las Ciencias Informáticas (UCI) en conjunto con la Unidad de Compatibilización, Integración y Desarrollo de Software para la Defensa (UCID) la tarea de desarrollar un ERP que cuente con todos los módulos necesarios para la gestión empresarial del país. La magnitud, complejidad y seguridad de un sistema de este tipo requiere hacer un diseño arquitectónico potente que soporte la gestión y transferencia de información. Para ello se creó un grupo central de Arquitectura que fue definiendo por cada capa, los componentes que formarían parte de la línea base. De estos componentes, algunos fueron reutilizados y otros extendidos de ZendFramework para dar lugar al Marco de Trabajo de Cedrux que adoptó el nombre de Sauxe.

A medida que el sistema Cedrux avanza, mayores son los requerimientos que debe cumplir la arquitectura para soportar y brindar las prestaciones requeridas. Actualmente, Sauxe se encuentra en la versión 2.0 y continúa desarrollando tecnologías robustas y cada vez son más los interesados en aprenderlas y trabajar sobre esta arquitectura.

Por el nivel de integración, asimilación y robustez de este Marco de Trabajo se encuentra generalizado en 15 proyectos de la UCI (Aduana, Akademos2.0, Cedrux, DITRANS, FTC, Generador de reportes, Hoyo, Informatización UCID, LiberGIS, Minería de datos, SEUNE, SIGAC, SIMEM, Sistema de gestión estadística, SRCAMERM), el MINFAR y otras entidades del país como el Polo de Desarrollo de Holguín alcanzando resultados satisfactorios. (Baryolo, y otros, 2009).

Por tanto se hace indispensable poder contar con una documentación precisa, clara, formalizada y entendible para que tanto los desarrolladores de Cedrux así como los clientes o usuarios finales que actualmente utilizan este Marco de Trabajo en diferentes proyectos productivos, puedan interactuar sin dificultades con dicha arquitectura. En este momento no se cuenta con una documentación de este tipo, lo que trae como consecuencia dudas a la hora de reutilizar sus componentes, incremento del tiempo de estudio, mayor necesidad por parte de los desarrolladores de realizar consultas constantes al equipo de arquitectura, lo que puede desencadenar un atraso considerable en los cronogramas de trabajo, además, del descontento por parte de los usuarios finales.

Dada la situación planteada anteriormente el **problema a resolver** queda formulado de la siguiente manera: La documentación existente de la arquitectura tecnológica de Sauxe no permite entender con claridad cómo extender sus componentes para dar solución a nuevos escenarios.

El **objeto de estudio** es la arquitectura tecnológica para aplicaciones web de gestión en entornos multi-entidad.

El **campo de acción** queda enmarcado en la documentación técnica de la arquitectura tecnológica.

El **objetivo general** es estandarizar la documentación técnica de la arquitectura tecnológica de Sauxe versión 2.0.

Objetivos Específicos:

- Recopilar la documentación existente de la arquitectura tecnológica de Sauxe.
- Describir la arquitectura tecnológica de Sauxe.
- Detallar los escenarios arquitectónicos.
- Especificar los requisitos arquitectónicos.
- Describir componentes arquitectónicos.
- Validar la documentación técnica de Sauxe.

Idea a defender:

Si se formaliza y centraliza la documentación técnica de la Arquitectura tecnológica de Sauxe versión 2.0, se logrará entender con claridad cómo extender sus componentes para dar solución a nuevos escenarios.

Estructura de los capítulos de la tesis

La tesis está compuesta por tres capítulos.

Capítulo 1. Fundamentación Teórica.

Capítulo 2. Escenarios y requisitos arquitectónicos.

Capítulo 3. Componentes arquitectónicos.

Resultado esperado

Documentación técnica de la arquitectura tecnológica del Marco de Trabajo Sauxe versión 2.0.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En este capítulo se dan a conocer los conceptos principales para el mejor entendimiento del problema, se realiza un estudio de los diferentes Marcos de Trabajo basados en PHP. Se exponen las principales características y ventajas de Zend. Se abordan temas relacionados con el Marco de Trabajo Sauxe. Además se hace un detallado estudio de la documentación del Marco de Trabajo ZendFramework. Se exponen también, algunas características y actividades del modelo que se utiliza en el Departamento de Tecnología basado en las diferentes metodologías ágiles. Se da a conocer el impacto que tiene la documentación.

1.2. Conceptos generales

Escenarios arquitectónicos

Los escenarios son equivalentes no funcionales de los casos de uso, cuantificables, es decir no existen los diagramas de escenarios. (Reynoso, 2005)

Los escenarios se usan para entender y validar la arquitectura, establecer cierta comunicación entre la arquitectura y aquellos que no tuvieron mucho que ver con su creación, unir las diferentes vistas y entender los límites de la arquitectura. (González, 2001)

Requisitos arquitectónicos

Los requerimientos o requisitos no son más que condiciones o capacidades que tienen que ser alcanzadas o poseídas por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.

Arquitectura de software

La arquitectura del software es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación y tiene la responsabilidad de:

- Definir los módulos principales.
- Definir las responsabilidades que tendrá cada uno de estos módulos.
- Definir la interacción que existirá entre dichos módulos.
- Control y flujo de datos.
- Secuenciación de la información.
- Protocolos de interacción y comunicación.

- Ubicación en el hardware.

La arquitectura del software aporta una visión abstracta de alto nivel, posponiendo el detalle de cada uno de los módulos definidos a pasos posteriores del diseño. (Casanovas, 2004)

Metodologías de desarrollo

Conjunto de procedimientos, técnicas, herramientas y soporte documental que deben seguirse para el desarrollo del software. (Universidad Rey Juan Carlos, 2009)

Componente de software

Un componente de software es una unidad de composición con interfaces contractualmente especificadas y explícitas sólo con dependencias dentro de un contexto. Un componente de software puede ser desplegado independientemente y es sujeto a la composición de terceros. (Rojas, y otros, 2004)

Un componente de software en tiempo de ejecución es un paquete dinámicamente vinculado con uno o varios programas manejados como una unidad y que son accedidos mediante interfaces bien documentadas que pueden ser descubiertos en tiempo de ejecución. (Rojas, y otros, 2004)

Aplicaciones web

Aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. Es decir, es una aplicación de software que se codifica en un lenguaje soportado por los navegadores web en la que se confía la ejecución al navegador. Las aplicaciones web son populares debido a la facilidad para actualizar y mantenerlas sin tener que distribuir e instalar software a miles de usuarios. (Hooping.net, 2009)

Marco de Trabajo

Es un conjunto de componentes que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas web. Los objetivos principales que persigue son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones. (Gutiérrez, 2009)

1.2.1. Marcos de Trabajo

Los Marcos de Trabajo son desarrollados con el objetivo de brindarles a los programadores y diseñadores una mejor organización y estructuración a sus proyectos. En sus inicios, para llevar a cabo el desarrollo de Sauxe se hizo necesario realizar un estudio de los Marcos de Trabajo más reconocidos mundialmente por las facilidades que brindan a la hora de optimizar código y estandarizar la programación de una manera eficiente. Como principales y más usados se destacan: Symfony, CakePHP, CodeIgniter, KumbiaPHP y

ZendFramework. De ellos, se partió de ZendFramework para llevar a cabo la extensión de algunos de los componentes de interés para el Departamento de Tecnología, debido a que éste brinda facilidades para desarrollar aplicaciones y servicios web, además por su facilidad a la hora de extender[1] sus componentes.

1.2.2. ZendFramework

Se trata de un Marco de Trabajo de código abierto para el desarrollo de aplicaciones y servicios web con PHP, brinda soluciones para construir sitios web modernos, robustos y seguros. Es una implementación que usa código 100% orientado a objetos. Este Marco de Trabajo está formado por una serie de métodos estáticos y componentes que usarán estos métodos. Los componentes son varios y variados y aunque alguno es posible que no se use nunca, hay otras que puede que se usen hasta la saciedad, por ejemplo el componente para la base de datos. La estructura de los componentes de ZendFramework es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado. A menudo, se refiere a este tipo de diseño como " uso a voluntad". Aunque se pueden utilizar de forma individual, los componentes de la biblioteca estándar de ZendFramework conforman un potente y extensible Marco de Trabajo de aplicaciones web al combinarse. Como características fundamentales ZendFramework tiene:

- Trabaja con Modelo Vista Controlador (MVC)
- El Marco de Zend también incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para consultar su base de datos, sin tener que escribir ninguna consulta SQL.
- Una solución para el acceso a base de datos que balancea el Mapeador de Objeto Relacional con eficiencia y simplicidad.
- Completa documentación y test de alta calidad.
- Soporte avanzado.
- Robustas clases para autenticación y filtrado de entrada.
- Muchas otras clases útiles para hacerlo tan productivo como sea posible (Leopoldo, 2007)

ZendFramework ofrece un gran rendimiento y una robusta implementación MVC, una abstracción de base de datos fácil de usar y un componente de formularios que implementa la prestación de formularios HTML, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos. Otros componentes, como Zend_Auth y Zend_Acl, proveen

autentificación de usuarios y autorización diferentes a las tiendas de certificados comunes. (Diseño web y desarrollo de software, 2009)

Ventajas principales de Zend Framework:

- Estandariza los procesos más frecuentes, dotándolos de gran robustez.
- Facilita el mantenimiento de las aplicaciones.
- Ofrece muchas facilidades para el acceso a recursos avanzados (web services securizados, por ejemplo) que de otro modo resultan bastante más costosos de desarrollar.
- A diferencia de otros Marcos de Trabajo, es posible utilizarlo en modo "desacoplado", es decir, aquellas clases o componentes que sean necesarios en cada proyecto, sin arrastrar todo el Marco de Trabajo detrás para cualquier pequeña necesidad.
- Tiene el respaldo de la propia ZEND, creadora de PHP, lo que asegura su continuidad futura tanto como la del propio lenguaje PHP. (Diseño web y desarrollo de software, 2009)

Los 51 componentes de ZendFramework conforman un potente y extensible Marco de Trabajo de aplicaciones web al combinarse entre sí. De todos estos Sauxe utiliza solamente los siguientes:

1. Zend_Cache
2. Zend_Config
3. Zend_Controller
4. Zend_Loader
5. Zend_Exception
6. Zend_Registry
7. Zend_Session
8. Zend_View

A partir de la extensión de algunos componentes de ZendFramework surge ZendExt, desarrollado por el Departamento de Tecnología y la UCID, con el objetivo de crear un Marco de Trabajo extensible y configurable centrando el desarrollo de las aplicaciones, en la lógica del negocio, en las interfaces de usuario, alejando a los programadores de los detalles arquitectónicos, con soporte para entornos multientidad y para una arquitectura de sistema orientada a componentes. (Baryolo, y otros, 2008)

La unión de ZendExt, Doctrine y Extjs dio lugar al Marco de Trabajo Sauxe.

1.2.3. Marco de Trabajo Sauxe

Sauxe es un Marco de Trabajo que contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. (Baryolo, y otros, 2008)
en Sauxe su versión 2.0 está formado por los componentes: ZendExt_App, ZendExt_Aspect, ZendExt_ADT, ZendExt_Cache, ZendExt_ExpImp, ZendExt_MVC, ZendExt_Exception, ZendExt_FastResponse, ZendExt_GlobalConcept, ZendExt_IoC, ZendExt_Nomencladores, ZendExt_Trace, ZendExt_Validation, ZendExt_Portal, ZendExt_TransactionManager.

1.2.4. Doctrine

Doctrine es un potente y completo sistema ORM (Mapeador de Objeto Relacional) para PHP 5.2.3+ que incorpora una DBL (Capa de Abstracción a Base de Datos). (ServerGrove, 2010)

Sauxe utiliza en la capa de acceso a datos el Lenguaje de Consulta de Datos (DQL) que implementa Doctrine. La documentación de éste tiene todas las características necesarias para ser funcional en casi cualquier proyecto. Entre otros elementos se tiene la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir, convertir clases (convenientemente creadas) a tablas de una base de datos. Por otro lado, como la librería es bastante grande tiene un método para ser 'compilada' al pasar a producción. (Baryolo, y otros, 2008)
Ventajas que facilitan enormemente tareas comunes y de mantenimiento:

1. **Reutilización:** La principal ventaja que aporta un ORM es la reutilización permitiendo llamar a los métodos de un objeto de datos desde distintas partes de la aplicación e incluso desde diferentes aplicaciones.
2. **Encapsulación:** La capa ORM encapsula la lógica de los datos pudiendo hacer cambios que afectan a toda la aplicación únicamente modificando una función.
3. **Portabilidad:** Utilizar una capa de abstracción que permite cambiar en mitad de un proyecto de una base de datos MySQL[2] a una Oracle sin ningún tipo de complicación. Esto es debido a que no se utiliza una sintaxis (MySQL, Oracle[3] o SQLite[4]) para acceder al modelo, sino una sintaxis propia del ORM utilizado que es capaz de traducir a diferentes tipos de bases de datos.
4. **Seguridad:** Los ORM suelen implementar mecanismos de seguridad que protegen nuestra aplicación de los ataques más comunes como una Inyección SQL[5].

5. **Mantenimiento del código:** Gracias al correcto ordenamiento de la capa de datos, modificar y mantener el código es una tarea sencilla (Mata, 2009)

1.2.5. ExtJs

Como se expuso anteriormente Sauxe también contiene ExtJs.

ExtJs es una librería Java Script ligera y de alto rendimiento, compatible con la mayoría de los navegadores que permite crear páginas e interfaces web dinámicas. (Comunidad de desarrollo, 2006-2010)

Sauxe utiliza ExtJs en la capa de presentación, por la gran gama de componentes que se pueden reutilizar para agilizar el proceso de desarrollo y mostrarle al usuario una interfaz más amigable y funcional. (Baryolo, y otros, 2008)

Esta librería incluye:

- Componentes UI (Interfaz de Usuario) del alto performance y personalizables.
- Modelo de componentes extensibles.
- Un API[6] fácil de usar.
- Licencias de códigos abiertos y comerciales. (Corzo, 2008)

Una de las grandes ventajas de utilizar ExtJS es que permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de layouts[7] similar al que provee Java Swing[8], gracias a esto provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno (Firefox, Internet Explorer, Safari[9]). Además, la ventana flotante que provee ExtJS es excelente por la forma en la que funciona. Al moverla o redimensionarla sólo se dibujan los bordes haciendo que el movimiento sea fluido lo cual le da una ventaja tremenda frente a otros. (Corzo, 2008)

Como se ha visto hasta el momento Sauxe es el resultado de una importante mezcla entre: la extensión de algunos componentes de un potente Marco de Trabajo, un sistema ORM y una muy popular librería, dando lugar a un novedoso Marco de Trabajo, con funcionalidades y prestaciones específicas pero también fácilmente extensibles a cualquier aplicación web de gestión.

1.2.6. Documentación de ZendFramework

A la hora de desarrollar la documentación de Sauxe, se realizó un análisis de la documentación de ZendFramework, porque fue el Marco de Trabajo que se tomó para extender sus componentes.

Contiene una documentación abarcadora, explícita y actualizada, cuenta con una “Guía de referencia para programadores” en la que se detallan todos y cada uno de sus componentes, también contiene una lista extensa de ejemplos en el que se especifican diferentes casos que pueden surgir durante el desarrollo. También tiene una “Guía de estudio”, en la cual se explican elementos básicos sobre ZendFramework, así como un conjunto de actividades que permiten ejercitar conocimientos. Ambas guías se encuentran en inglés. Además en la web tiene publicado un sitio en el que aparece la descripción de todos los componentes, algunos de estos han sido traducidos al español. Este sitio, da la posibilidad a todos aquellos usuarios interesados en colaborar con la comunidad Zend, de realizar la traducción de los componentes que deseen y que ésta sea publicada en la web, esta página web se encuentra en la encuentra en la dirección a la cual se hace referencia:(ZendFramework des.com, 2010)

Independientemente de la extensa y abarcadora documentación con la que cuenta Zend, no llega a ser suficiente para Sauxe, debido a que, aun cuando este lo utiliza, tiene extensiones de algunos de sus componentes dando lugar a un nuevo Marco de Trabajo, que cuenta además con Doctrine para la gestión de la capa de acceso a datos y ExtJs en la capa de presentación. Por tanto, surge la necesidad de desarrollar una documentación a la medida que se ajuste a los elementos, particularidades y características propias de este Marco de Trabajo.

1.2.7. Modelo de desarrollo

Para llevar a cabo el Modelo de desarrollo que se utiliza actualmente en el Departamento de Tecnología se realizó en el Trabajo de Diploma: “Propuesta de modelo de desarrollo de software tecnológico del Centro de Soluciones de Gestión” un estudio de las diferentes metodologías tanto ágiles como tradicionales más usadas a nivel mundial concluyendo que no existe una metodología lo suficientemente robusta para hacer frente con éxito a cualquier proyecto de desarrollo de software, debido a que cada proyecto es único y tiene sus propias necesidades.

El desarrollo de la documentación se hará utilizando un modelo de desarrollo elaborado en el Departamento de Tecnología, el mismo tiene su basamento en los principios y buenas prácticas de las metodologías ágiles estudiadas para garantizar rapidez y un buen funcionamiento en el desarrollo de los procesos.(Pérez, y otros, 2009)

El modelo provee los procesos, actividades, roles involucrados y artefactos generados durante el proceso de desarrollo tecnológico en la Subdirección Tecnológica del CEIGE. Este está formado por 8 procesos y

una estructura organizativa. Cada proceso es especificado teniendo en cuenta su flujo de actividades, los roles involucrados y artefactos generados en cada actividad. (Pérez, y otros, 2009)

Este modelo de desarrollo tiene las siguientes características:

- Se utilizan solamente los artefactos necesarios para documentar el producto.
- Se basa en la reutilización de componentes.
- Existen áreas dedicadas a tareas específicas y especializadas en temas específicos.
- Se hacen pruebas continuas sobre los compones y/o productos y los cambios se hacen a tiempo.
- Antes de poner un componente en el repositorio se hacen pruebas unitarias y cuando se va a utilizar como parte de otro producto se hacen pruebas de integración, al igual que antes de liberar el producto también. Todo esto demuestra que se está probando en todo el proceso de desarrollo.
- Es un método muy estructurado que funciona bien con personas de poca experiencia.
- Reduce los riesgos ya que:
    - Provee visibilidad sobre el progreso a través de sus nuevas versiones.
    - Provee retroalimentación a través de la funcionalidad mostrada.
    - Permite atacar los mayores riesgos desde el inicio.
- La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software.
- En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.
- Preocupación por el aprendizaje de los desarrolladores. (Pérez, y otros, 2009)

La documentación se guiará de acuerdo a lo que establece el “Proceso de desarrollo de componentes tecnológicos”.

Este proceso inicia cuando se solicita a la Subdirección Técnica del centro iniciar una iteración de desarrollo tecnológico, a partir de una lista de requisitos. (Pérez, y otros, 2009)

Este proceso contiene un total de 8 actividades a seguir, para darle cumplimiento al mismo. Para el desarrollo de la documentación las actividades a tener en cuenta fueron la 1, 5.1 y 6.3 debido a que las otras abordaban otros temas como la construcción del componente, estabilización y rectificación de errores, pruebas unitarias del componente, integración del componente a la solución global, selección y desarrollo de Marcos de Trabajo, librerías, empaquetamiento, selección de la plataforma arquitectónica, descripción y especificación del estilo arquitectónico, pruebas de integración y estabilización del

componente, elaboración de los documentos rectores y de control, elaboración de cursos de capacitación, las cuales no son un objetivo a tener cuenta en el desarrollo de esta solución.

Actividad: Selección de los escenarios tecnológicos.

Descripción: Se realiza una valoración y evaluación de los escenarios tecnológicos de acuerdo al tipo de aplicación que se desea desarrollar y se selecciona la configuración más óptima según las características propias del proyecto, los atributos de calidad que debe alcanzar la aplicación y los recursos materiales y financieros disponibles. (Pérez, y otros, 2009)

Artefacto generado:

- Vista de la Arquitectura tecnológica.

  Objetivo: Formalizar el escenario tecnológico a utilizar.

Actividad: Diseño del componente.

Descripción: Se realiza el diseño de cada uno de los componentes definidos, para ello se puntualizan las entradas, las salidas y una breve descripción de lo que va a hacer el mismo, además se especifican y describen los diagramas de clases, diagramas de secuencia, modelo de datos y se incluyen en el documento de especificación técnica de componentes, así como los patrones a utilizar. (Pérez, y otros, 2009)

Artefacto:

- Documento de especificación técnica de componentes.

  Objetivo: Documentar el diseño para cada componente definido y los patrones que fueron utilizados para el mismo.

Actividad: Desarrollo de la documentación del Marco de Trabajo.

Descripción: Desarrollar toda la documentación necesaria para utilizar e implantar el Marco de Trabajo. (Pérez, y otros, 2009)

Artefacto:

- Caso de estudio.

  Objetivo: Guiar a los desarrolladores en el uso del Marco de Trabajo.

1.2.8. Impactos de la documentación

El desarrollo de la documentación del Marco de Trabajo traerá consigo un impacto considerable tanto en la productividad del programador como en la integridad del sistema así como en la usabilidad del software.

1.2.8.1. Impacto en la productividad del programador

La existencia de una documentación detallada, constituirá para los desarrolladores una guía referencial que les permitirá realizar las tareas acordes a su rol, tomando de ésta los conocimientos que le son necesarios para el desarrollo e implantación de los componentes. Esto permite reducir el número de consultas a los arquitectos, ahorro de tiempo y mayor uniformidad en el proceso de programación.

1.2.8.2. Impacto en la integridad del sistema

La documentación debe estar centralizada y completa, de esta forma se evita tener que realizar encuentros innecesarios y talleres. Esto contribuye a que el sistema sea cada vez más robusto, tenga mayor rendimiento y cumpla con todos los requerimientos especificados por el cliente.

1.2.8.3. Impacto en la usabilidad del software

Contar con una documentación fiable, y sólida en su contenido, es esencial para llevar a cabo cualquier actividad de desarrollo. Es fundamental para el logro en tiempo y forma de cualquier tarea productiva; constituye una fuente de motivación para los responsables de efectuar el desarrollo de determinado producto. Esto asegurará mayores resultados en la utilización del proyecto, mayor auge en la venta del producto así como la satisfacción del equipo de desarrollo y usuarios finales.

1.3. Herramienta utilizada

Hay varias herramientas creadas para el desarrollo de la ingeniería de software. Estas existen con el fin de desarrollar programas, utilizando técnicas de diseño y metodologías bien definidas, soportadas por herramientas automatizadas. (Giraldo, y otros, 2005)

Visual Paradigm

Visual Paradigm es una herramienta CASE[10] que sirve para realizar modelado UML[11] siguiendo el estándar UML 2.1.Tiene licencia gratuita y comercial. Esta herramienta tiene características gráficas muy cómodas que facilitan la realización de los diagramas de modelado que sigue el estándar de UML, los cuales son:

Diagramas de clase, casos de uso, comunicación, secuencia, estado, actividad, componentes, entre otros. (Altamirano, 2009)

Entre sus características están:

- Producto de calidad.
- Soporta aplicaciones web.
- Las imágenes y reportes generados, no son de muy buena calidad.
- Varios idiomas.
- Generación de código para Java y exportación como HTML.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones. (Giraldo, y otros, 2005)
- Integración con diversas IDE's[12] como son: NetBeans[13] , Eclipse[14] (de IBM[15]).
- Ingeniería inversa para Java, .NET[16], XML.
- Exportación de imágenes jpg y png. (Altamirano, 2009)

Visual Paradigm ofrece:

- Disponibilidad en múltiples plataformas.
- Entorno de creación de diagramas para UML 2.1
- Disponibilidad de integrarse en los principales IDE´s.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación. (Sierra, 2009)

1.4. Conclusiones

Se realizó un estudio detallado del estado del arte concluyendo con la apremiante necesidad de desarrollar una documentación a la medida para el Marco de Trabajo Sauxe. Se planteó la vía de solución teniendo en cuenta la metodología señalada. Se investigó sobre el impacto en la productividad del programador, integridad del sistema y usabilidad del software.

CAPÍTULO 2. ESCENARIOS Y REQUISITOS ARQUITECTÓNICOS

2.1. Introducción

En este capítulo se realiza una explícita descripción de los principales escenarios y requisitos arquitectónicos de los 15 componentes que contiene la versión 2.0 del Marco de Trabajo del Departamento de Tecnología.

2.2. Escenarios arquitectónicos.

2.2.1. Elementos no controlados del sistema.

La arquitectura tecnológica de la plataforma proveerá algún mecanismo de gestión, configuración y administración de excepciones de manera dinámica y declarativa. El mismo deberá integrarse con el elemento responsable de las trazas, cuando ocurra el suceso de una excepción. La solución debe incluir además la capacidad de definir familias de excepciones y dentro de las familias de excepciones, excepciones tipo concretas, de cada excepción se podrá definir características tipo tales como: línea de código e instrucción en se generó, traceo de la excepción que la inició, fecha y hora entre otros metadatos que permiten una visualización y análisis de las trazas emitidas por el sistema por cada excepción. El mecanismo de excepciones permitirá además definir de manera declarativa el comportamiento de la excepción, si se persistirá, si será presentada con un mensaje de notificación que navegue hasta la capa cliente, si será una excepción ciega, o una excepción a reemplazar por otra nueva excepción que formatee la expresión. La solución de excepción controlará también aquellas excepciones disparadas por el sistema y conocidas como excepciones no controladas o errores internos. (Baryolo, 2010)

2.2.2. Permitir la definición de los nomencladores generales

Debe proveerse de un mecanismo para la gestión de objetos a nivel de datos (bases de datos, esquemas, secuencias, trigger[17], funciones, roles, tablas, atributos, entre otros.) desde la capa de aplicación. De esta forma los usuarios podrán configurar a nivel de sistema operaciones como exportar e importar datos, gestionar estructuras dinámicas como nomencladores, atributos y relaciones entre las estructuras desde las aplicaciones sin necesidad de interactuar directamente con un cliente de algún gestor en específico. (Baryolo, 2010).

2.2.3. Cargar ficheros XML

Realizar un mecanismo para cargar ficheros XML (Lenguaje de Etiquetado Extensible). Este mecanismo permite erradicar los conflictos o abrazos fatales que se producen cuando dos o más usuarios acceden y modifican información de un mismo fichero. Además garantiza que se cree una única instancia del archivo y que se almacene en caché para agilizar el proceso de consulta al mismo. Este componente le facilita a los desarrolladores el acceso o manipulación de los ficheros localizados en cualquier directorio.

2.2.4. Realizar validaciones antes de ejecutar una acción

Realizar un mecanismo de gestión, configuración y administración de validaciones para sucesos del sistema, definiéndose suceso como la unidad central de ejecución que plantee la solución arquitectónica que se defina, por ejemplo, ya sea un caso de uso o una acción de un Modelo Vista Controlador si fuese este el estilo arquitectónico que centra la plataforma tecnológica. En tal sentido la solución deberá inyectar tanto al script[18] presente o enviado al lado del cliente como al script o código perteneciente al lado del servidor las instrucciones o reglas de validación. Las mismas deberán estar basadas en un formato y podrán ser extendidas, modificadas o adicionadas para una configuración arquitectónica específica mediante la propuesta solución arquitectónica que provee la plataforma para la gestión de validación. La validación deberá estar concentrada en dos áreas fundamentales, una primera área de validación orientada a las clases de equivalencia de los atributos definidos según el contrato definido o bien para el formulario cliente o bien para la funcionalidad del lado del servidor que se invoque y la otra área orientada a las reglas precondicionales de las unidades de ejecución (acciones por ejemplo), de manera que el sistema pueda a través del marco de validación chequear mediante el uso de inversión de control las reglas de validación declarativamente configuradas para la acción en concreto. (Baryolo, 2010)

2.2.5. Integrar módulos o subsistemas permitiendo la ejecución de funciones de unos a otros

Proveer la capacidad para configurar y gestionar de manera dinámica la integración de las aplicaciones o dominios de soluciones que se instancien o construyan con la misma. Esta integración deberá permitir la comunicación entre componentes expresando sus contratos en servicios mediante el formato WSDL[19]. El soporte de integración permitirá de manera transparente para el usuario consumir un servicios gestionado por la plataforma desde 3 alternativas específicas, una por vía de un servicio web, otra por algún mecanismo de inversión de control que deberá proveer la arquitectura de integración del Marco de Trabajo, y la otra desde algún servidor de comunicación o servicio de mensajería, cualquiera de estas vías deberán ser transparentes en su instanciación para el desarrollador, puesto que la plataforma deberá

proveer una capa de abstracción basada en el patrón proxy que permita encapsular y tratar a lo interno la configuración e instanciación del mecanismo específico que se solicita en el enlace de integración. La plataforma deberá proveer además un mecanismo para configurar, gestionar, consultar y probar la cartera de servicios que ha sido configurada con ésta. (Baryolo, 2010)

2.2.6. Interfaz única de gestión centralizada de los diferentes subsistemas y módulos

Permite la capacidad de integrar todas las representaciones o respuestas del sistema que naveguen hasta la capa cliente de la arquitectura en un área de trabajo única, taxonómicamente ordenada, que se integra con los mecanismos de autenticación y autorización de la plataforma, con el marco proveedor de estructuras y composiciones y con el marco de representación de mensajes, el mismos deberá basarse en una plataforma que soporte las tecnologías de presentación HTML[20], CSS[21] 2.0, DHTML[22], JS[23], y mecanismos de comunicación asíncrono con el servidor como Ajax[24]. Se recomienda para ello el uso de ExtJs o la tecnología XUL[25] como dos marcos tecnológicos candidatos eficientes y que cumplen con las características anteriormente pactadas. (Baryolo, 2010)

2.2.7. Mecanismo que permita interoperar un conjunto de procesos o volúmenes de datos

Realizar un mecanismo de interoperabilidad o de exportar e importar procesos o volúmenes de información que permita configurar los datos que se desean extraer de las distintas aplicaciones para ser utilizadas por otros sistemas una vez importados. Este mecanismo viene a resolver los escenarios de despliegue en los que no existen conexión o integración entre los sistemas.

2.2.8. Almacenar en caché valores o conceptos muy reutilizados

Debe proveer la capacidad para configurar y gestionar de manera dinámica la caché del Marco de Trabajo. La caché estará orientada al cliente (páginas, imágenes, css, js), al código fuente (acelerador de código, caché del binario que se instancia), al dominio de la solución (caché de instancia de objetos, estructuras, consultas, ficheros XML). (Baryolo, 2010)

2.2.9. Acceder a datos generales desde cualquier parte de la aplicación

Realizar un mecanismo para la configuración dinámica y centralizada de variables globales que son aspectos muy importantes en los Marcos de Trabajo ya que agrupan información común conceptualizada que se maneja en todo el sistema, o sea, en todos los módulos y sesiones de una aplicación. Por lo general todo Marco de Trabajo que se implementa contiene un componente de este tipo para que cada parte del sistema pueda acceder de forma fácil y sencilla a datos que oferte otra sin incurrir en códigos engorrosos que implican gastos de tiempo y esfuerzos para los programadores. Por lo tanto la

implementación de estos contenedores de información común permite que la misma sea almacenada una sola vez y actualizada en el momento conveniente, así cuando un componente necesite de esta información la toma del contenedor sin tener que estar realizando peticiones constantes a los servicios que oferten los demás componentes. (Baryolo, 2010)

2.2.10. Permite obtener la secuencia de acciones de un usuario sobre un sistema

Realizar un mecanismo de administración y configuración dinámica de trazas de la solución. La solución deberá permitir configurar la arquitectura de traza de un dominio de aplicación específico, permitiendo para ello crear categorías de traza que constituyan familias de trazas, donde a su vez cada categoría de traza presentará instancias de trazas específicas o sub-categorías; las categorías de trazas pueden ser de diferentes dominios, según interés del cliente de la tecnología, aunque el Marco de Trabajo debe presentar categorías de trazas tecnológicas por defecto que permitan registrar los sucesos de excepciones, entradas al sistema de los usuarios, ejecución de una acción, ejecución de una integración interna o externa dentro de la arquitectura, estas trazas por defecto presentarán los respectivos metadatos de información que los caracteriza. La solución arquitectónica de traza de la plataforma deberá permitir que cada traza presente una cola de subscritores, que deberán ser disparados ante la ocurrencia de la misma. (Baryolo, 2010)

2.2.11. Contar con un mecanismo de gestión de estructuras de datos

Mecanismo para la gestión de estructuras de datos. La mayoría de la información se gestiona o registra en forma de arreglos, listas, colas, pilas, árboles, entre otros. Estas transformaciones se realizan para agilizar u optimizar la manipulación o transferencia de los datos. La arquitectura tecnológica debe proveer un mecanismo que permita trabajar con diversas estructuras de datos de forma rápida, fácil y eficiente.

2.2.12. Permite salvar uno o varios objetos de forma automática

Mecanismo para la administración de transacciones. De manera que las operaciones de modificación sobre el modelo de dominio sean transaccionales por defecto, este mecanismo será transparente para el programador. La solución arquitectónica permitirá configurar el esquema transaccional de un dominio específico de la solución que se construye, definiéndose en este caso para una transacción de negocio, cuando inicia y cuando debe terminar, además de permitir configurar notificaciones ante la posibilidad de fallas en la transacción, capturándose como un evento más el rollback[26] de la transacción, estas notificaciones pueden ser configuradas por un mecanismo central de notificaciones que implemente para

toda la tecnología un administrador de patrón observador, o podrán ser inyectadas por algún mecanismo de inversión de control. (Baryolo, 2010)

2.2.13. Mecanismo de relación de las interfaces de usuarios con la lógica del negocio

Realizar un mecanismo que relacione las interfaces de usuario con la lógica del negocio. Este componente se centra en la reutilización de los componentes y en la facilidad para conectar nuevos módulos de interfaz sin afectar al modelo. Muestra una simetría en el comportamiento del modelo con respecto a las vistas y el controlador. Ante un cambio en su estado, el modelo avisa de este cambio tanto a la vista como al controlador. Esto hace que el modelo sea algo más mecánico, sistemático, no hace distinciones entre el controlador y la vista, cuando cambia avisa a todos y al que le afecte el cambio será el que tenga que buscar los datos accediendo al modelo. Resuelve el problema de actualización de las vistas de una aplicación (ventanas de la interfaz de usuario) cuando cambian sus datos.

2.2.14. Permite activar o desactivar aspectos o funcionalidades del Marco de Trabajo

La arquitectura tecnológica de la plataforma proveerá algún mecanismo para la configuración dinámica y centralizada de aspectos que pueden o no utilizarse en el desarrollo o ejecución de un sistema. La solución deberá permitir activar o desactivar aspectos como la multi-entidad, seguridad, multi-moneda, trazas, historiales, validaciones y demás aspectos que pueden o no ser usados por los sistemas. Para esto se propone utilizar Programación Orientada a Aspectos (POA) que es un paradigma de programación relativamente reciente cuya intención es permitir una adecuada modularización de las aplicaciones y posibilitar una mejor separación de conceptos. Gracias a la POA se pueden encapsular los diferentes conceptos que componen una aplicación en entidades bien definidas, eliminando las dependencias entre cada uno de los módulos. (Baryolo, 2010)

2.2.15. Mecanismo que permita inicializar los elementos arquitectónicamente significativos

La arquitectura tecnológica de la plataforma proveerá algún mecanismo de gestión única y centralizada de todos los escenarios previstos antes o después de ejecutar una acción. Esta solución debe tener un alto nivel de extensión y sencillez a la hora de incluir los nuevos escenarios que surjan en el desarrollo de las aplicaciones.

2.3. Requisitos arquitectónicos

RF1–Configuración y registro de las distintas excepciones que pueden ocurrir en cada acción de las clases de control y modelos que se implementan.

Precondiciones	Las excepciones deben estar definidas.
Flujo de eventos	
Flujo básico	
1	Debe permitir controlar de forma centralizada los diferentes tipos de excepciones y tratarlas según el tipo especificado.
Pos-condiciones	
1	Las excepciones desencadenan un conjunto de acciones.
Conceptos	Concepto: ZendExt_Exception

Tabla 1. Configuración y registro de las excepciones

RF2-Gestionar conceptos o tablas

RF2.1-Adicionar conceptos o tablas

Precondiciones	Debe existir la base de datos.
Flujo de eventos	
Flujo básico	
1	Existe un esquema para los nomencladores donde se van a almacenar todos los nomencladores que se creen. Además en una tabla que se crea es donde se especifican como va a ser cada uno de estos nomencladores creados.
Pos-condiciones	
1	Quedan adicionados todos los nomencladores necesarios para el proyecto.
Conceptos	Concepto: ZendExt_Nomencladores

Tabla 2. Adicionar tablas

RF2.2-Modificar conceptos o tablas

Precondiciones	Debe existir la base de datos.
Flujo de eventos	
Flujo básico	
1	Se modifican los campos, categorías, se habilita o deshabilita el nomenclador, se cambia además la forma de visualizar el nomenclador.
Pos-condiciones	

1	Quedan modificados los nomencladores según las necesidades del usuario.	
Conceptos	Concepto: ZendExt_Nomencladores	

Tabla 3. Modificar tablas

RF2.3-Eliminar conceptos o tablas

Precondiciones	Debe existir la base de datos.
Flujo de eventos	
Flujo básico	
1	Se elimina el nomenclador del esquema y de la tabla que lo reconocía como tal.
Pos-condiciones	
1	Quedan eliminados los nomencladores que no se utilizarán por el momento.
Conceptos	Concepto: ZendExt_Nomencladores

Tabla 4. Eliminar tablas

RF3-Gestionar campos

RF3.1-Adicionar campos a conceptos o tablas

Precondiciones	Debe existir la base de datos.	
Flujo de eventos		
Flujo básico		
1	Existe una tabla que se denomina campo. Aquí se recogen todos los campos que se crean por el nomenclador. Esta tiene el nombre de la tabla a la que pertenece el nomenclador, el tipo y si es llave primaria o no.	
Pos-condiciones		
1	Quedan adicionados todos los campos necesarios a las tablas.	
Conceptos	Concepto: ZendExt_Nomencladores	Atributos internos: Nombre de la tabla, tipo, llave primaria.

Tabla 5. Adicionar campos a la tabla

RF3.2-Modificar campos a conceptos o tablas

Precondiciones	Debe existir la base de datos.

Flujo de eventos		
Flujo básico		
1	Se modifican todos los atributos de los campos. Estos son: el nombre de la tabla a la que pertenece el nomenclador, el tipo y si es llave primaria o no.	
Pos-condiciones		
1	Quedan modificados todos los campos necesarios de la tabla.	
Conceptos	Concepto: ZendExt_Nomencladores	Atributos internos: Nombre de la tabla, tipo, llave primaria.

Tabla 6. Modificar campos a la tabla

RF3.3-Eliminar campos a conceptos o tablas

Precondiciones	Debe existir la base de datos
Flujo de eventos	
Flujo básico	
1	Se eliminan físicamente los campos en la tabla y de la tabla donde el componente los reconoce como tal.
Pos-condiciones	
1	Quedan eliminados todos los campos innecesarios.
Conceptos	Concepto: ZendExt_Nomencladores

Tabla 7. Eliminar campos a la tabla

RF4-Gestionar elementos

Precondiciones	Las tablas de la base de datos deben estar creadas.
Flujo de eventos	
Flujo básico	
1	Adicionar datos a las diferentes tablas de la base de datos.
Pos-condiciones	
1	Quedan adicionados todos los datos en la base de datos.
Flujos alternativos 1a	
1	Se insertan datos que están registrados ya en la base de datos.

2	Se muestra un mensaje diciendo que estos datos ya están archivados.
3	Volver al flujo básico 1.
Flujos alternativos 1b	
1.	Se entran datos incorrectos.
2	Se muestra un mensaje diciendo que los datos no son válidos.
3	Volver al flujo básico 1.
Conceptos	Concepto: ZendExt_Nomencladores

Tabla 8. Gestionar elementos

RF5–Validar acción

RF5.1-Buscar acción

Precondiciones	Debe haberse invocado alguna acción implementada en la clase control.	
Flujo de eventos		
Flujo básico		
1	Permite buscar en el xml de las validaciones las especificaciones de validación de la acción que se invocó en el cliente.	
Pos-condiciones		
1	Se encuentran y cargan las validaciones correspondientes al método.	
Conceptos	Concepto: ZendExt_Validation	Atributo interno: VALIDATION_EXCEPTION_ACTIVE

Tabla 9. Buscar acción

RF5.2-Validar los datos según el tipo especificado

Precondiciones	Deben haberse encontrado las especificaciones correspondientes a la acción en el xml.
Flujo de eventos	
Flujo básico	

1	Validar, según las especificaciones encontradas, los parámetros con los que va a trabajar la acción invocada, en cuanto al tipo de los mismos.
Pos-condiciones	
1	Quedan validados los datos introducidos por el usuario.
Conceptos	Concepto: No procede / Atributo interno: No procede

Tabla 10. Validar los datos

RF6-Integrar subsistemas con transferencia de datos

Precondiciones	Los subsistemas deben estar creados.
Flujo de eventos	
Flujo básico	
1	Debe permitir la comunicación entre los diferentes subsistemas con envío y recepción de datos una vez realizadas sus configuraciones pertinentes. Estas peticiones se ejecutarán de un subsistema a otro.
Pos-condiciones	
1	Se establece la comunicación entre los diferentes subsistemas.
Conceptos	Concepto: ZendExt_IoC

Tabla 11. Integrar subsistemas con transferencia de datos

RF7-Verificar acceso a entidad

Precondiciones	El usuario debe tener acceso a la aplicación.
Flujo de eventos	
Flujo básico	
1	Acceder a la aplicación donde aparece un cuadro de diálogo que contiene la estructura de entidades.
2	Se selecciona la entidad en la que se desee trabajar
3	Se muestra una ventana de autenticación en la que se inserta el usuario y la contraseña asignada a la entidad.
4	Se muestra la aplicación.

Pos-condiciones		
1	Realizar en la aplicación las actividades pertinentes.	
Conceptos	Concepto:	Atributos visibles en la interfaz:
	ZendExt_Portal	Entidad, ventana de autenticación, contraseña.

Tabla 12. Verificar acceso a entidad

RF8–Gestionar la información de la caché

RF8.1-Registrar información de la caché

Precondiciones	Debe existir la base de datos.
Flujo de eventos	
Flujo básico	
1	Se almacena información de cualquier tipo ya sean objetos, clases, ficheros, arreglos, entre otros.
Pos-condiciones	
1	Quedan almacenados los datos temporales asociados a una aplicación.

Conceptos	Concepto:	Atributos internos:
	ZendExt_Cache	objetos, clases, ficheros, arreglos

Tabla 13. Registrar información de la caché

RF8.2-Modificar información de la caché

Precondiciones	Debe existir la base de datos.
Flujo de eventos	
Flujo básico	
1	Se modifica la información previamente almacenada en la caché. Esta información consta de objetos, clases, ficheros, arreglos, entre otros.
Pos-condiciones	
1	Se modifican los datos temporales asociados a una aplicación.

Conceptos	Concepto:	Atributos internos:
	ZendExt_Cache	objetos, clases, ficheros, arreglos

Tabla 14. Modificar información de la caché

RF8.3-Eliminar información de la caché

Precondiciones	Debe existir la base de datos.
Flujo de eventos	
Flujo básico	

1	Se elimina la información previamente almacenada en la caché.
Pos-condiciones	
1	Se eliminan los datos temporales asociados a una aplicación.
Conceptos	Concepto: ZendExt_Cache — Atributos internos: objetos, clases, ficheros, arreglos

Tabla 15. Eliminar información de la caché

RF9-Configurar e inicializar las conexiones

Precondiciones	Debe existir el App.config.
Flujo de eventos	
Flujo básico	
1	Sauxe utiliza ficheros de configuración que en dependencia del modelo que se utilice se crearán las conexiones respectivas. Para inicializar las conexiones, el App.config lee los ficheros de configuración y los pone en un registro, después el Administrador de transacciones las inicializa.
Pos-condiciones	
1	Quedan configuradas las conexiones.
Conceptos	Concepto: ZendExt_App

Tabla 16. Configurar e inicializar las conexiones

RF10-Registrar traza de:

- Autenticación

Precondiciones	Deben existir las trazas.
Flujo de eventos	
Flujo básico	
1	Se registra una traza cuando se autentica un usuario en el sistema. Se guardan los datos siguientes: usuario, categoría, tipo de traza, fecha, IP de la máquina donde se autenticó.
Pos-condiciones	
1	Se registra la traza.

Conceptos	Concepto: ZendExt_Trace	Atributos visibles en la interfaz: Usuario, categoría, tipo de traza, fecha, IP de la máquina donde se autenticó.

Tabla 17. Registrar traza de autenticación

RF11-Gestionar la configuración

RF11.2-Validación

Precondiciones	Se activan e incluyen en el template y se aplica a la acción que se desea validar.
Flujo de eventos	
Flujo básico	
1	Este aspecto se encarga de realizar las validaciones en la capa del negocio de los datos provenientes del cliente y de algunas validaciones necesarias antes de ejecutar una acción.
Pos-condiciones	
1	Se ejecutan las validaciones de ser positivas sino se muestra un mensaje de error.
Conceptos	Concepto: ZendExt_ Aspect

Tabla 18. Gestionar configuración de validación

RF12-Separar los datos de la interfaz de usuario y la lógica de control de la aplicación

Precondiciones	Deben existir los componentes modelo, vista y controlador.
Flujo de eventos	
Flujo básico	
1	Se establece un estilo de llamada y retorno donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

Pos-condiciones	
1	Se establece la relación entre los tres componentes.
Conceptos	Concepto: ZendExt_MVC

Tabla 19. Separar datos de la interfaz de usuario y la lógica de control de la aplicación

RF13-Inicializar las conexiones

Precondiciones	Debe existir el App.config.
Flujo de eventos	
Flujo básico	
1	El administrador de transacciones inicializa las conexiones que el App.config lee y pone en un registro.
Pos-condiciones	
1	Quedan inicializadas las conexiones.
Conceptos	Concepto: ZendExt_TransactionManager

Tabla 20. Inicializar las conexiones

2.4. Conclusiones

Se detallaron todos los escenarios y requisitos arquitectónicos de los 15 componentes de la versión 2.0 del Marco de Trabajo llegando a la conclusión de la gran importancia que tienen estos para el proyecto porque son los que contienen las necesidades del usuario y en ellos están contenidas las soluciones a esos problemas que el usuario tiene, o sea, se asegura que el software que se realice cumpla con las necesidades indicadas inicialmente y además sea robusto y confiable.

CAPÍTULO 3. COMPONENTES ARQUITECTÓNICOS

3.1. Introducción

En este capítulo se hace una detallada descripción de los componentes de Sauxe que se desarrollan para la versión 2.0 del Marco de Trabajo. Hasta el momento son 15 componentes, estos son: ZendExt_Exception, ZendExt_Nomencladores, ZendExt_FastResponse, ZendExt_Validation, ZendExt_IoC, ZendExt_Portal, ZendExt_ExpImp, ZendExt_Cache, ZendExt_GlobalConcept, ZendExt_Trace, ZendExt_ADT, ZendExt_App, ZendExt_MVC, ZendExt_TransactionManager, ZendExt_Aspect. Se mostrará además el diagrama general que relaciona a todos los componentes que contendrá esta versión y una breve explicación de la arquitectura.

3.2. Explicación general de la arquitectura

La arquitectura está compuesta básicamente por cinco niveles o capas:

1. Capa de Presentación: En esta capa se emplea las facilidades que brinda el Marco de Trabajo ExtJS para la construcción de interfaces amigables a la vista de los usuarios. Ext centra su desarrollo en tres componentes fundamentales JS-File, CSS-File, Client-page y Server-Page.
2. Capa de Control o Negocio: En esta capa se emplea el patrón de arquitectura Modelo Vista Controlador (MVC). De forma vertical al modelo descrito hasta este momento, estarán los aspectos fundamentales del sistema así como el encargado del tratamiento de la seguridad a nivel de aplicación web.
3. Capa de Acceso a Datos: En esta capa estará presente el Mapeador de Objeto Relacional Doctrine, como persistidor para la comunicación con el servidor de datos mediante el protocolo PDO, también estará un persistidor de Configuración que es el encargado de comunicarse vía XML con los Ficheros de Configuración del sistema denominado FastResponse.
4. Capa de Datos: En esta capa estará ubicado como servidor de base de datos PostgreSQL y un conjunto de ficheros de configuración de la Arquitectura tecnológica.
5. Capa de Servicio: En esta última capa se encuentran todos los subsistemas que prestan y consumen servicios entre sí.

Esta gran estructura descrita, se comunica con una serie de servicios web mediante protocolos SOAP, e IoC, que interactúan con el sistema proporcionándole un conjunto de funcionalidades tanto de seguridad como de negocio.

3.3. Breve descripción de cada componente

A continuación se muestra un diagrama con la relación que existe entre cada uno de los componentes.

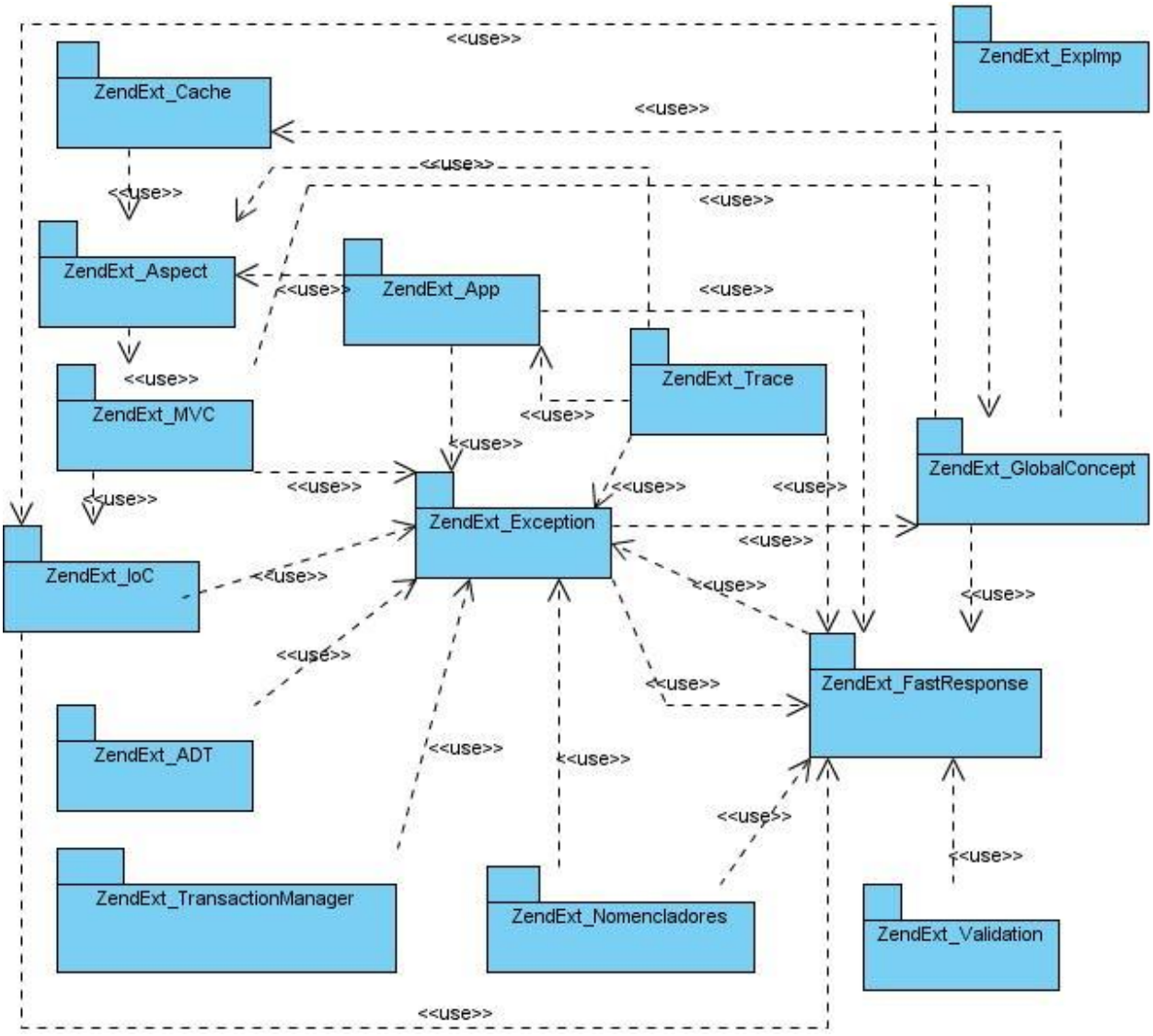

Figura 1.Diagrama general de componentes

Inicializador de aplicaciones (ZendExt_App)

Este componente se encarga de realizar las configuraciones iniciales, verificaciones y validaciones. Es el punto de inicio a partir del cual se desencadenan un conjunto de acciones necesarias antes de realizar cualquier operación.

Excepciones (ZendExt_Exception)

Es una situación anómala o condición de error que ocurre durante el tiempo de ejecución de un programa. El nombre "excepción" viene del hecho de que aunque un problema puede ocurrir lo hace con poca frecuencia, esto es, de manera excepcional. El mecanismo de manejo de excepciones sólo permite lanzar y atrapar objetos de la clase Exception y sus clases derivadas.

Modelo Vista Controlador (ZendExt_MVC)

Este componente separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres partes distintas las cuales son el modelo, la vista y el controlador. El modelo se limita a lo relativo de la vista y su controlador facilitando las presentaciones visuales complejas. El sistema también puede operar con más datos no relativos a la presentación, haciendo uso integrado de otras lógicas de negocio y de datos afines con el sistema modelado. La vista presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario y el controlador responde a eventos, como las acciones del usuario, e invoca peticiones al modelo y a la vista.

Validaciones (ZendExt_Validation)

Este componente debe validar todos los datos y acciones que se activan ya sean por un usuario o por un sistema externo, que serán ejecutadas en un controlador de una aplicación en específico. La acción será validada antes de llegar al controlador para no hacer peticiones innecesarias a éste.

Nomencladores (ZendExt_Nomencladores)

A los programadores de cualquier lenguaje les resulta ventajoso poder optimizar el código de programas muy grandes para que les sea fácil de implementar y reutilizar posteriormente estos. Con ese objetivo se implementa el componente de nomencladores quien tiene el deber de gestionar los nomencladores comunes para permitir después el acceso a ellos mismos, gestionar los nomencladores de cada subsistema y permitir el acceso de varios subsistemas a un nomenclador común.

FastResponse (ZendExt_FastResponse)

La implementación de este componente trae muchas ventajas a los programadores de cualquier lenguaje debido a que les permite el acceso único a recursos además de servir como fachada para cargar ficheros una sola vez, esto ayuda a que no haya repetición de código, es decir, que se pueda reutilizar el mismo

cuantas veces sea necesario haciendo así el trabajo más fácil y asequible al desarrollador, lo cual ahorra tiempo y trae resultados satisfactorios.

Inversión de Control (ZendExt_IoC)

En la inversión de control se especifican respuestas deseadas a sucesos o solicitudes de datos concretas, dejando que algún tipo de entidad o arquitectura externa lleve a cabo las acciones de control que se requieran en el orden necesario y para el conjunto de sucesos que tengan que ocurrir. El flujo habitual se da cuando es el código del usuario quien invoca a un procedimiento de una librería. La inversión de control sucede cuando es la librería la que invoca el código del usuario. Típicamente sucede cuando la librería es la que implementa las estructuras de alto nivel y es el código del usuario el que implementa las tareas de bajo nivel.

Exportar_Importar (ZendExt_ExpImp)

Este componente se utiliza para centralizar el proceso de exportar e importar objetos. En este momento cuando se necesita, por ejemplo, exportar algún tipo de información de una de las tablas de la base de datos, el programador toma de esta solamente los datos que le interesan y en la forma que desea. Permite que cuando se desee trabajar con esta información no ocurran conflictos en la base de datos.

Traza (ZendExt_Trace)

Las trazas son utilizadas, para comprobar que se realizan exactamente las funciones esperadas, y no otras, así como acreditar las validaciones de datos previstas, sin modificar el sistema en ningún momento. También son usadas con el fin de monitorear las acciones que se realicen (acceso a ficheros, dispositivos, empleo de los servicios) y para detectar indicios de hechos relevantes a los efectos de la seguridad que puedan afectar la estabilidad o el funcionamiento del sistema informático.

Conceptos Globales (ZendExt_GlobalConcept)

Los contenedores de variables globales son aspectos muy importantes en los Marcos de Trabajo ya que agrupan información común conceptualizada que se maneja en todo el sistema, o sea, en todos los módulos y sesiones de una aplicación. La implementación de estos contenedores de información común permite que la misma sea almacenada una sola vez y actualizada en el momento conveniente, así cuando un componente necesite de esta información la toma del contenedor sin tener que estar realizando peticiones constantes a los servicios que oferten los demás componentes y sin generar lentitudes al sistema.

Portal (ZendExt_Portal)

Al iniciar el desarrollo de un software los arquitectos y diseñadores orientan su trabajo en el diseño del prototipo de interfaz inicial que tendrá el sistema. En la mayoría de las situaciones es muy difícil reutilizar parte de estos códigos fuentes y diseños de estas interfaces, además, los clientes casi siempre tienen una concesión diferente de la arquitectura de presentación que no es más que la forma en la que se organiza la información a mostrar. Por esta razón, se hace necesario crear un componente genérico que sea capaz de adaptarse a las particularidades de los usuarios con un alto nivel de reutilización cumpliendo con estándares para el diseño de interfaces de usuarios.

Caché (ZendExt_Cache)

La caché no es más que un conjunto de información que generalmente se almacena en ficheros, tanto en clientes como en servidores web, en un formato determinado, de forma tal que pueda ser recuperada rápida y eficientemente, permitiendo almacenar información de cualquier tipo, textos, clases, funciones, objetos o instancias de clases, estructuras complejas como listas, pilas, árboles, arreglos o tablas hash (es una estructura de datos que asocia llaves o claves con valores), páginas, ficheros, por lo general esta información no debe cambiar en un periodo de tiempo determinado, en caso de que alguna información almacenada en la caché cambie o expire esta debe ser actualizada instantáneamente, para evitar problemas de inconsistencia. La mayoría de los gestores de caché almacenan esta información en forma serializada para ganar en rendimiento y espacio de almacenamiento.

Estructura de datos (ZendExt_ADT)

Una estructura de datos es una forma de organizar un conjunto de datos elementales con el objetivo de facilitar su manipulación. Las estructuras de datos más comunes que se utilizan son: grafo, lista, cola, pila y árbol.

Administrador de transacciones (ZendExt_TransactionManager)

Este componente permite controlar de forma centralizada un conjunto de operaciones a realizar en la base de datos. Además ayuda a establecer prioridades, eliminar los cuellos de botellas, recuperación ante la ocurrencia de errores y erradica la inconsistencia de las bases de datos.

Aspectos (ZendExt_Aspect)

Este componente permite activar y desactivar un conjunto de aspectos arquitectónicamente significativos que pueden ser usados o no según los escenarios de despliegue. Da la posibilidad de que los usuarios o desarrolladores cuenten con una vía fácil para realizar las configuraciones del Marco de Trabajo por medio de ficheros XML.

3.4. Explicación detallada de los componentes

3.4.1. Aspectos a tener en cuenta en la plantilla de los componentes del Marco de Trabajo.

Requisitos que implementa

Capacidad o condición que debe cumplir el componente.

Escenarios que resuelve

Es una vista a gran escala de las posibles situaciones que pueden darse según el componente en cuestión. Engloba un conjunto de requisitos.

Diagrama de clases

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro.

Diagrama de paquetes

Los diagramas de paquetes se usan para reflejar la organización de paquetes y sus elementos. Se pueden construir para representar relaciones tanto físicas como lógicas. El paquete debe mostrar el nombre del paquete y puede opcionalmente mostrar los elementos dentro del paquete en compartimientos extras. (Ambler, 2003-2009)

Modelo de datos

Relación de clases y entidades.

Es uno de los elementos más importantes a la hora de iniciar el desarrollo de cualquier proyecto. Esta es la estructura, sobre la que realmente reside la verdadera esencia de la aplicación. Incluso determina si el proyecto va a cumplir con su verdadero objetivo. El modelado de datos es una técnica independiente de la implementación a la base de datos. (Lemus, 2009)

Prototipo de interfaz de usuario

La interfaz de usuario es el vínculo entre el usuario y el programa de computadora. Una interfaz es un conjunto de comandos o menús a través de los cuales el usuario se comunica con el programa. Esta es una de las partes más importantes de cualquier programa ya que determina que tan fácilmente es posible que el programa haga lo que el usuario quiere hacer. (Loaiza, 2000)

Configuración y uso del componente

Explica de forma detallada la configuración que necesita el componente para que funcione y la forma correcta en que debe usarse.

Caso de estudio de cómo implementar el uso del componente

Se incluyen todas las posibles formas de uso del componente.

Caso de prueba

Se selecciona un caso de éxito para ser usado como caso de prueba.

Es una secuencia de pasos a seguir para verificar la funcionalidad de un componente.

Patrones usados

Un patrón es una solución de diseño de software a un problema, aceptada como correcta, a la que se ha dado un nombre y que puede ser aplicada en otros contextos. (Molpeceres, 2002)

Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. El objetivo de los patrones es crear un lenguaje común a una comunidad de desarrolladores para comunicar experiencia sobre los problemas y sus soluciones. Pueden referirse a distintos niveles de abstracción, desde un proceso de desarrollo hasta la utilización eficiente de un lenguaje de programación. (Figueroa, 2007)

Clasificación de los patrones:

Los patrones se clasifican en patrones arquitectónicos y de diseño.

- Patrones de arquitectura: Aquellos que expresan un esquema organizativo estructural fundamental para sistemas software.
- Patrones de diseño: Aquellos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas software. (Figueroa, 2007)

3.4.2. ZendExt_Nomencladores

Problema a resolver

En los sistemas de gestión, en muchos casos es un tanto complejo modelar toda la amalgama de probabilidades que componen el negocio, por lo que se hace necesario elevar el nivel de abstracción de datos que se gestionen a nivel de negocio en los que se configuran en tiempo de ejecución.

Cedrux no escapa a esta problemática por lo que se le asigna al equipo de desarrollo de arquitectura la tarea de llevar a vías de hecho un componente que satisfaga las necesidades previamente descritas y además de un mecanismo para la generación automática de interfaces gráficas para gestionar los elementos de estas estructuras aunque resuelve la problemática de los metadatos al menos en una primera versión se ha evadido una solución de ese entorno.

Objetivos

- Construir un componente que permita la gestión de las estructuras dinámicas no sólo a nivel de la interfaz creada al efecto sino además por la instanciación de la clase ZendExt_Nomencladores_ADT.
- Permitir el acceso de varios subsistemas a un nomenclador común.
- Gestionar los nomencladores comunes para acceder después a ellos.

Requisitos funcionales

Ver requisitos RF2, RF3, RF4 del capítulo 2.

Escenarios que resuelve

Ver escenario 2.2.2 del capítulo 2.

Diagrama de clases

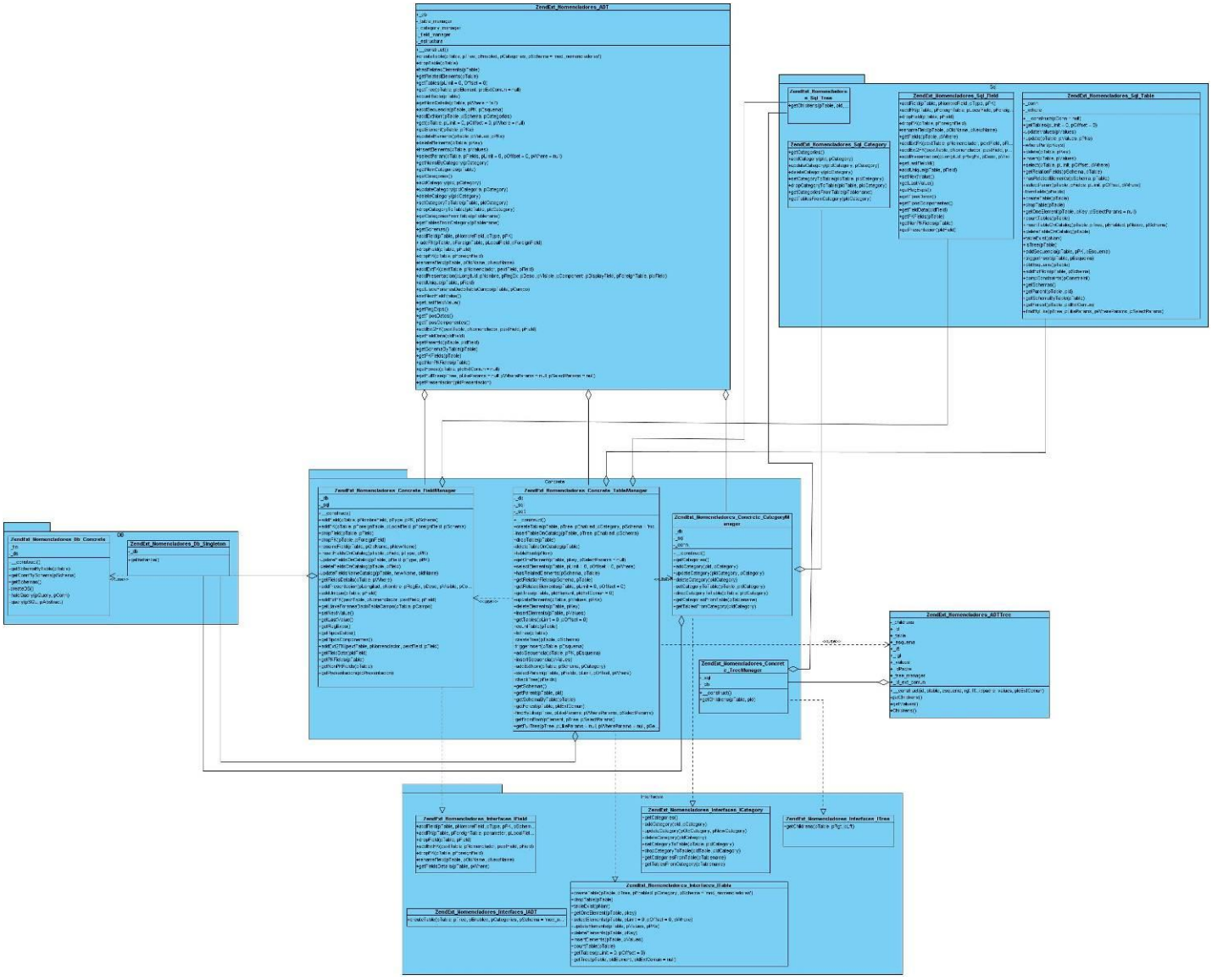

Figura 2. Diagrama de clases. ZendExt_Nomencladores

Diagrama de paquetes

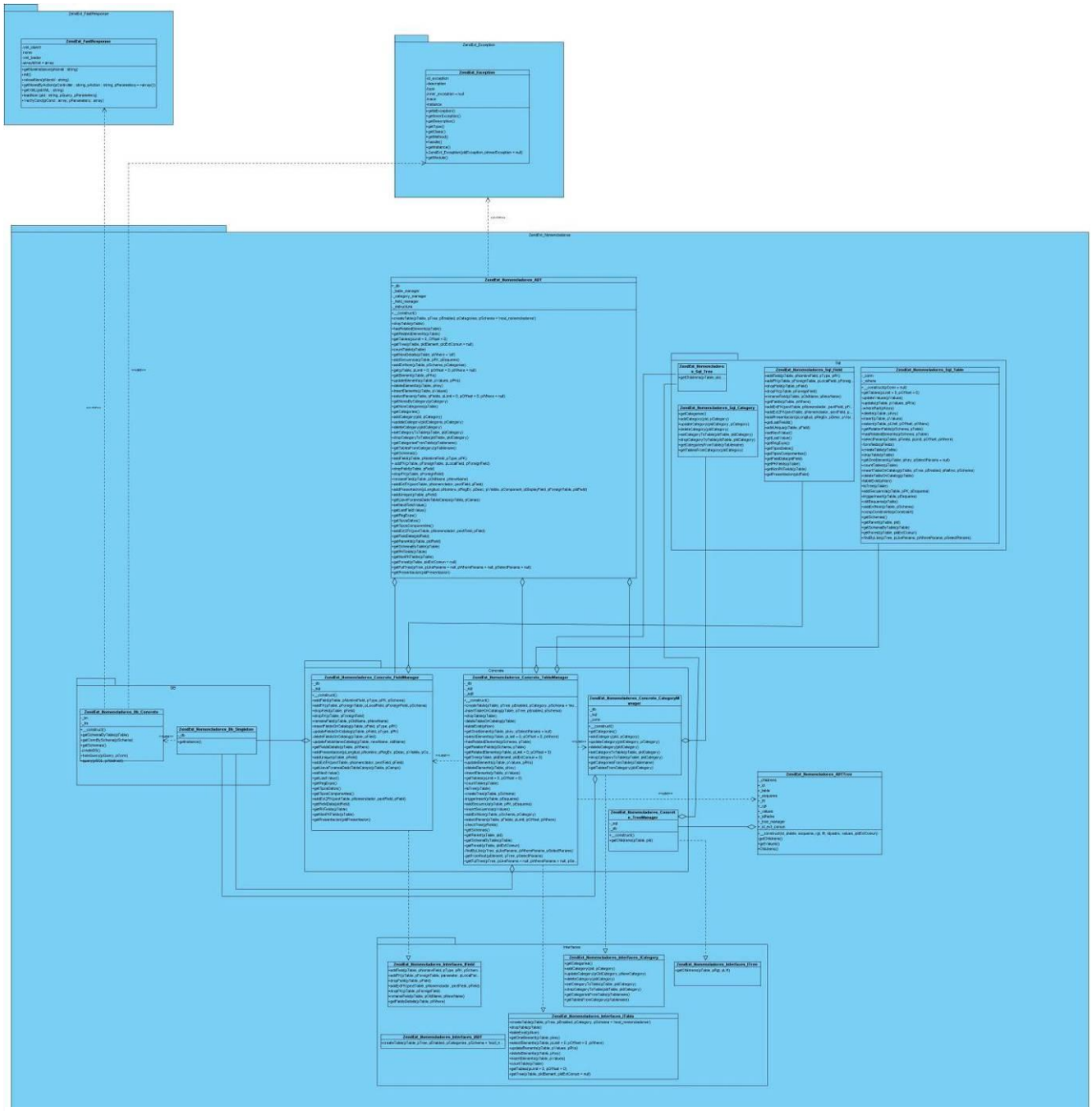

Figura 3. Diagrama de paquetes. ZenExt_Nomencladores

Modelo de datos

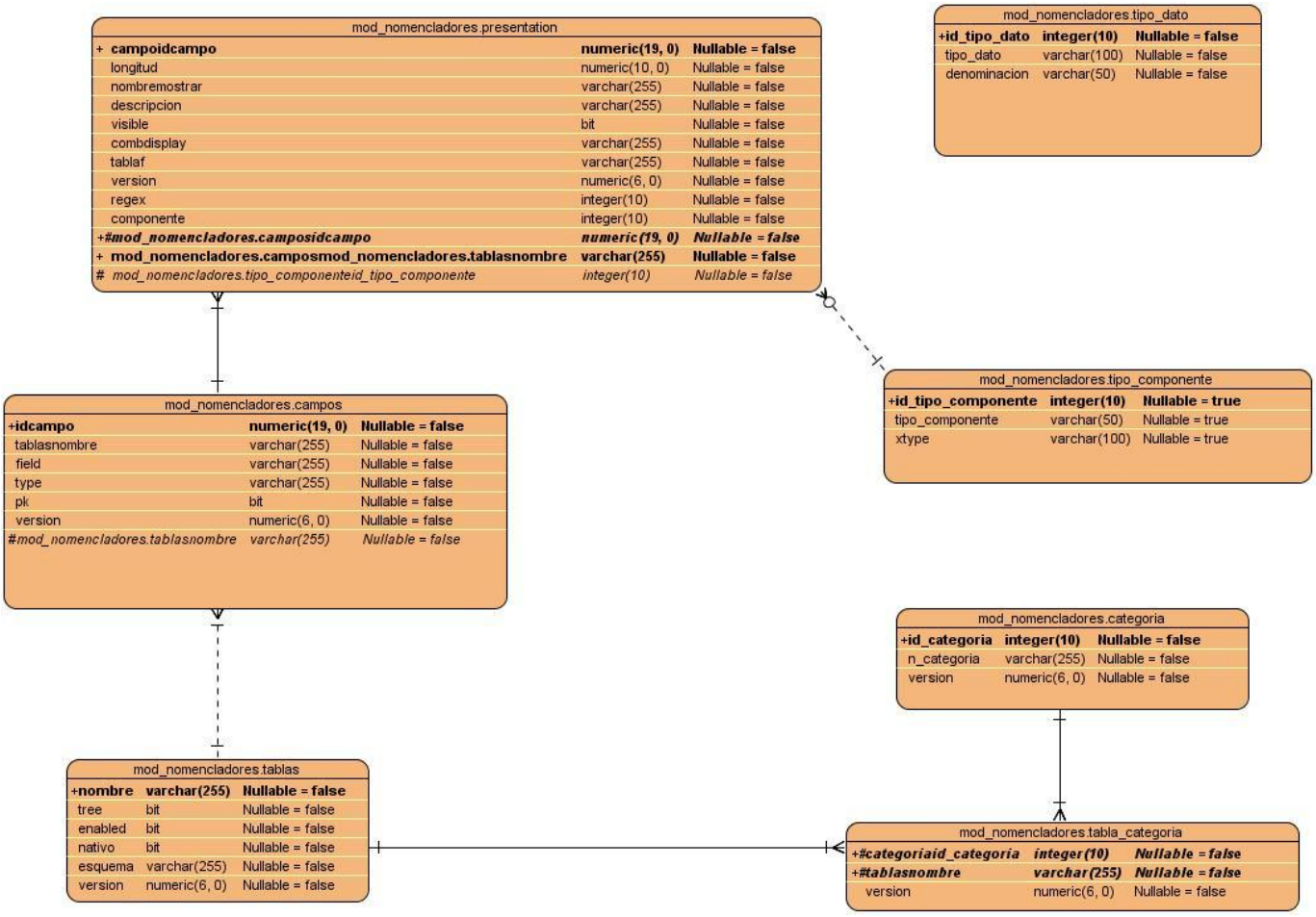

Figura 4. Modelo de datos. ZendExt_Nomencladores

Prototipo de interfaz de usuario

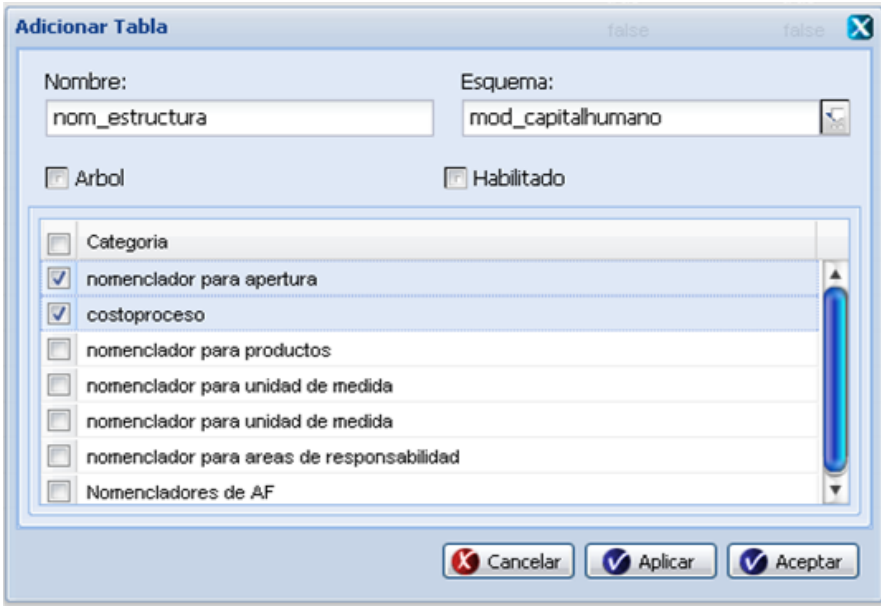

Figura 5. Prototipo de interfaz de usuario. ZendExt_Nomencladores

Configuración y uso

Para darle salida a estos objetivos se aplica un mecanismo de capas encargadas de:

Acceso a datos: Englobado en el espacios de nombre ZendExt_Nomencladores_Db se gestiona una instancia única para el cumplimiento de su objeto funcional con dos clases: Concrete y Singleton. En el caso de las conexiones se pueden obtener pasándole los siguientes parámetros, el nombre de una tabla, del esquema en concreto al que se desea acceder y en su defecto una instancia de una conexión de Doctrine. El mismo hace uso del componente ZendExt_Aspect_TransactionManager como elemento de la arquitectura para la gestión de las transacciones para las cuales el componente posee dos métodos para la ejecución de consultas, dos métodos query () y holeQuery (), que basan su más importante diferencia en que en el caso del primero ejecuta todos en una misma transacción y en el otro es para usar una sola transacción.

Manejadores concretos: Estos están agrupados por cada unas de las áreas temáticas que forman el componente o sea: la gestión de tablas a través de la clase TableManager, la gestión de categorías con la clase CategoryManager y la gestión de campos con FieldManager así como la gestión de árboles por parte TreeManager, todo en el espacio de nombre ZendExt_Nomencladores_Concrete.

Capa de gestión de SQL: Cada una de las clases que se agrupan en el espacio de nombres ZendExt_Nomencladores_Sql concentra en sí todo el código SQL que usa el componente para llevar a cabo cada una de las funcionalidades del componente.

El componente permite la gestión de las estructuras dinámicas no sólo a nivel de la interfaz creada al efecto sino además por la instanciación de la clase ZendExt_Nomencladores_ADT cuyos métodos se encuentran descritos en el manual de usuario descrito al efecto en el presente.

Utilización a nivel de lógica

1. Funciones para gestión con las tablas

Crear una Tabla

Para llamar a la función que permite la creación de tablas se hace de la siguiente forma:

```
/**
 * Crea una tabla.
 *
 * @param String $pTable
 * @param bool $pTree
 * @param bool $pEnabled
 * @return bool.
 **/
 $adt = new ZendExt_Nomencladores_ADT();

$adt->createTable('nom8','true','true',array(1));
```

Parámetro1 (pTable) => Nombre de la tabla

Parámetro2 (pTree) => Si es arbólico o no

Parámetro3 (pEnabled) =>Si está habilitado o no lo arbólico

Parámetro4 (pCategories) =>Arreglo de enteros con los id de las categorías que se van a pasar

Parámetro5 (pSchema) => Arreglo con los esquemas

Resultado que devuelve la función => Si se creó o no la tabla

Eliminar las tablas

Para llamar a la función que permite la eliminación de tablas se hace de la siguiente forma:

```
/**
 * Eliminar las tablas
 *
 * @param String $pTable
 * @return void.
 **/
```

```
$adt = new ZendExt_Nomencladores_ADT();

$adt->dropTable('nom_nomenclador3');
```

Parámetro1 (pTable) => Nombre de la tabla

Obtener las tablas que existen

Para llamar la función que permite obtener las tablas que existen se hace de la siguiente forma:

```
/**
 * Obtiene las tablas que existen
 *
 * @param Int $pLimit
 * @param Int $Offset
 * @return Array
 **/

$adt = new ZendExt_Nomencladores_ADT();

print_r($adt->getTables());
```

Parámetro1 (pLimit) => Cantidad de elementos a mostrar.

Parámetro2 (Offset) => A partir de donde se van a comenzar a mostrar los elementos.

Resultado que devuelve la función => Devuelve en un arreglo todas las tablas que existen.

Devuelve una tabla en forma ZendExt_Nomencladores_ADT Tree

Para llamar a la función que permite devolver una tabla en forma ZendExt_Nomencladores_ADT Tree se hace de la siguiente forma:

```
/**
*Obtiene devuelve una tabla en forma ZendExt_Nomencladores_ADTTree
*
*
**/
$adt = new ZendExt_Nomencladores_ADT();

$ar = $adt->getTree('nom_nomencladorl','idnomencladorl = 1');
```

Parámetro1 (pTable) => Nombre de la tabla

Parámetro2 (pIdElement) =>Id del elemento que se va a mostrar

Resultado que devuelve la función => Devuelve una tabla en forma de ZendExt_Nomencladores_ADT Tree

Retorna los campos del nomenclador

Para llamar a la función que permite retornar los campos del nomenclador se hace de la siguiente forma:

```
/**
 * Retorna los campos del nomenclador
 *
 * @param String $pTable
 * @return Array
 **/
```

```
$adt = new ZendExt_Nomencladores_ADT();

print_r($adt->getNomDetails('nom_nomencladorl','nopk'));
```

Parámetro1 (pTable) => Nombre de la tabla

Parámetro2 (pWhere) =>Las condiciones para mostrar el elemento

Resultado que devuelve la función => Devuelve en un arreglo los campos de un nomenclador

2. Funciones para la gestión con los Datos

Retorna la colección de elementos de un nomenclador

Para llamar a la función que permite retornar la colección de elementos de un nomenclador se hace de la siguiente forma:

```
/**
 * Retorna la colección de elementos de un nomenclador
 *
 * @param String $pNom Nombre del nomenclador.
 * @param Int $pLimit Límite de la consulta.
 * @param Int $pOffset Desplazamiento de la consulta
 *@param string $pWhere Para alguna seleccion especifica
 * @return Array
 **/
$adt = new ZendExt_Nomencladores_ADT();

print_r($adt->get('nom_nomenclador2'));
```

Parámetro1 (pTable) => Nombre de la tabla

Parámetro2 (pLimit) => Cantidad de elementos a mostrar

Parámetro3 (pOffset) => A partir de donde se van a comenzar a mostrar los elementos

Parámetro4 (pWhere) =>Las condiciones para mostrar el elemento

Resultado que devuelve la función => Devuelve en un arreglo la colección de elementos de un nomenclador

Retorna los elementos de un nomenclador

Para llamar a la función que permite retornar los elementos de un nomenclador se hace de la siguiente forma:

```
/**
 * Retorna los elementos de un nomenclador
 *
 * @param String $pTable
 * @param Array $pPKs
 * @return Array
 **/
 $adt = new ZendExt_Nomencladores_ADT();
print_r($adt->getElement('nom_nomencladorl','idnomencladorl = 3'));
```

Parámetro1 (pTable) => Nombre de la tabla

Parámetro2 (pPKs) =>Llaves primarias por donde se van a mostrar

Resultado que devuelve la función => Devuelve en un arreglo los elementos de un nomenclador

Actualiza un elemento

Para llamar a la función que permite actualizar un elemento se hace de la siguiente forma:

```
/**
 * Actualiza un elemento
 *
 * @param String $pTable
 * @param Array $pValues
 * @param Array $pPKs
 * @return void
 **/
 $adt = new ZendExt_Nomencladores_ADT();
$adt->updateElements('nom_nomencladorl',array('idpadre'=>2),'idnomencladorl = 3');
```

Parámetro1 (pTable) => Nombre de la tabla

Parámetro2 (pValues) => Arreglo con los valores para insertar los elementos

Parámetro3 (pPKs) =>Llaves primarias por donde se van a mostrar

Elimina un elemento

Para llamar a la función que permite eliminar un elemento se hace de la siguiente forma:

```
/**
 * Elimina un elemento
 *
 * @param String $pTable
 * @param Array $pKey
 * @return void
 **/
$adt = new ZendExt_Nomencladores_ADT();
```

```
$adt->deleteElements('nom_nomencladorl','idnomencladorl = 3');
```

Parámetro1 (pTable) => Nombre de la tabla

Parámetro2 (pKey) =>Llave primaria

Inserta un elemento

Para llamar a la función que permite insertar un elemento se hace de la siguiente forma:

```
/**
 * Inserta un elemento
 *
 * @param String $pTable
 * @param Array $pValues
 * @return void.
 **/
 $adt = new ZendExt_Nomencladores_ADT();

$adt->insertElements('nom_nomencladorl',array('idnomencladorl'=>1,'idpadre'=>1));
```

Parámetro1 (pTable) => Nombre de la tabla

Parámetro2 (pValues) => Arreglo con los valores para insertar los elementos

3. Funciones para la gestión con las Categorías

3.1 Retorna todos los nomencladores que respondan a una categoría

Para llamar a la función que permite retornar todos los nomencladores que respondan a una categoría se hace de la siguiente forma:

```
/**
 * Retorna todos los nomencladores que respondan a una categoría.
 * @param String $pCategory.
 * @return Array.
 **/

 $adt = new ZendExt_Nomencladores_ADT();

print_r($adt->getNomsByCategory('categoria'));
```

Parámetro1 (pCategory) => El nombre de la categoría

Resultado que devuelve la función => Devuelve en un arreglo los nomencladores que respondan a una categoría

3.2 Retorna las categorías de un nomenclador

Para llamar a la función que permite retornar las categorías de un nomenclador se hace de la siguiente forma:

```
/**
 * Retorna las categorías de un nomenclador.
 *
 * @param String $pTable
 * @return Array
 **/
 $adt = new ZendExt_Nomencladores_ADT();

print_r($adt->getNomCategories('tabla1'));
```

Parámetro1 (pTable) => Nombre de la tabla

Resultado que devuelve la función => Devuelve en un arreglo las categorías de un nomenclador

3.3 Devuelve todas las categorías

Para llamar a la función que permite devolver todas las categorías se hace de la siguiente forma:

```
/**
* Devuelve todas las categorías.
*
* @return Array
**/
 $adt = new ZendExt_Nomencladores_ADT();

print_r($adt->getCategories());
```

Resultado que devuelve la función => Devuelve en un arreglo las categorías

3.4 Adiciona una categoría

Para llamar a la función que permite adicionar una categoría se hace de la siguiente forma:

```
/**
 * Adiciona una categoria
 *
 *@param int $pId
 * @param String $pCategory
 * @return void
 **/


  $adt = new ZendExt_Nomencladores_ADT();

$adt->addCategory(2,'Categoria2');
```

Parámetro1 (pId) => El id de la categoría

Parámetro2 (pCategory) => El nombre de la categoría

3.5 Actualiza el nombre de una categoría

Para llamar a la función que permite actualizar el nombre de una categoría t se hace de la siguiente forma:

```
/**
 * Actualiza el nombre de una categoría.
 *
 * @param Int $pIdCategoria
 * @param String $pCategory
 * @return void.
 **/
 $adt = new ZendExt_Nomencladores_ADT();

$adt->updateCategory(1,'Categoria2');
```

Parámetro1 (pIdCategoria) => El id de la categoría

Parámetro2 (pCategory) => El nombre de la categoría

3.6 Elimina una categoría

Para llamar a la función que permite eliminar una categoría se hace de la siguiente forma:

```
/**
 * Elimina una categoría
 *
 * @param Int $pIdCategory
 * @return void
 **/
 $adt = new ZendExt_Nomencladores_ADT();

$adt->deleteCategory(1);
```

Parámetro1 (pIdCategoria) => El id de la categoría

3.7 Adiciona una categoría a la tabla

Para llamar a la función que permite adicionar una categoría a la tabla se hace de la siguiente forma:

```
/**
 * Le adiciona una categoría a la tabla.
 *
 * @param Int $pTable
 * @param Int $pIdCategory
 * @return void.
 **/
 $adt = new ZendExt_Nomencladores_ADT();

$adt->setCategoryToTable('nom_nomenclador4',2);
```

Parámetro1 (pTable) =>Nombre de la tabla

Parámetro2 (pIdCategoria) =>El id de la categoría

3.8 Elimina una categoría de una tabla

Para llamar a la función que permite eliminar una categoría de una tabla se hace de la siguiente forma:

```
/**
 * Elimina una categiria de una tabla.
 *
 * @param Int $pIdTable
 * @param String $pIdCategory
 * @return void
 **/
 $adt = new ZendExt_Nomencladores_ADT();

$adt->dropCategoryToTable('nom_nomencladorl',2);
```

Parámetro1 (pIdTable) => Nombre de la tabla

Parámetro2 (pIdCategoria) => El id de la categoría

3.9 Obtiene las categorías de una tabla

Para llamar a la función que permite obtener las categorías de una tabla se hace de la siguiente forma:

```
/**
 * Obtiene las categorias de una tabla
 *
 * @param String $pTablename.
 * @return void.
 **/
$adt = new ZendExt_Nomencladores_ADT();

print_r($adt->getCategoriesFromTable('nom_nomenclador2'));
```

Parámetro1 (pTablename) => Nombre de la tabla

Resultado que devuelve la función => Devuelve las categorías de una tabla

3.10 Obtiene las tablas de una categoría

Para llamar a la función que permite obtener las tablas de una categoría se hace de la siguiente forma:

```
/**
 * Obtiene las tablas de una categoria.
 *
 * @param Int $pTablename.
 * @return void.
 **/
 $adt = new ZendExt_Nomencladores_ADT();

print r($adt->getTablesFromCategory(2));
```

Parámetro1 (pTablename) => Nombre de la tabla

Resultado que devuelve la función => Devuelve las tablas de una categoría

3.11 Obtiene los esquemas

Para llamar a la función que permite obtener los esquemas se hace de la siguiente forma:

```
/**
 * Obtiene los esquemas.
 *
 * @return void.
 **/
 $adt = new ZendExt_Nomencladores_ADT();

print_r($adt->getSchemas());
```

Resultado que devuelve la función => Devuelve los esquemas

Funciones para la gestión con los Campos

a. Adiciona un campo a una entidad de la BD

Para llamar a la función que permite adicionar un campo a una entidad de la BD se hace de la siguiente forma:

```
/**
 * Adiciona un campo a una entidad de la BD.
 *
 * @param String $pTable.
 * @param String $pNombreField.
 * @param String $pType.
 * @param bool $pPK.
 * @return void.
 **/


 $adt = new ZendExt_Nomencladores_ADT();

$adt->addField('nom nomenclador2',"campo","numeric (19)",'false');
```

Parámetro1 (pTable) => Nombre de la tabla

Parámetro2 (pNombreField) => Nombre del campo

Parámetro3 (pType) => El tipo

Parámetro4 (pPK) => Si es llave primaria o no

b. Adiciona una llave foránea

Para llamar a la función que permite adicionar una llave foránea se hace de la siguiente forma:

```
/**
 * Adiciona una llave foranea
 *
 * @param String $pTable
 * @param String $pForeignTable
 * @param String $pForeignField
 * @return void
 **/
```

```
$adt = new ZendExt_Nomencladores_ADT();
$adt->addFK('nom_nomenclador2','nom_nomencladorl','campo','idnomencladorl');
```

Parámetro1 (pTable) => Nombre de la tabla

Parámetro2 (pForeignTable) => Nombre de la tabla foránea

Parámetro3 (pLocalField) => Campo local

Parámetro4 (pForeignField) => Nombre del campo foráneo

c. Elimina un campo a una tabla

Para llamar a la función que permite eliminar un campo a una tabla se hace de la siguiente forma:

```
/**
 * Elimina un campo a una tabla
 *
 * @param String $pTable
 * @param String $pField
 * @return void
 **/
```

```
$adt = new ZendExt_Nomencladores_ADT();
$adt->dropField('nom_nomenclador2','campo');
```

Parámetro1 (pTable) => Nombre de la tabla

Parámetro2 (pField) => Nombre del campo

d. Elimina una llave foránea

Para llamar a la función que permite eliminar una llave foránea se hace de la siguiente forma:

```
/**
 * Elimina una llave foranea
 *
 * @param String $pTable
 * @param String $pForeignTable
 * @param String $pForeignField
 * @return void.
 **/
 $adt = new ZendExt_Nomencladores_ADT();

$adt->dropFK('nom_nomenclador4','nom_nomenclador4_idpadre');
```

Parámetro1 (pTable) => Nombre de la tabla

Parámetro2 (pForeignField) => Nombre del campo foráneo

e. Renombra un campo de la BD

Para llamar a la función que permite renombrar un campo de la BD se hace de la siguiente forma:

```
/**
 * Renombra un campo de la BD.
 *
 * @param String $pTable
 * @param String $pOldName
 * @param String $pNewName
 * @return void
 **/
 $adt = new ZendExt_Nomencladores_ADT();

$adt->renameField('nom_nomenclador2','campo','campo1');
```

Parámetro1 (pTable) => Nombre de la tabla

Parámetro2 (pOldName) =>Nombre antiguo

Parámetro3 (pNewName) =>Nuevo nombre

f. Crear llave foránea de externa

Para llamar a la función que permite crear llave foránea de externa se hace de la siguiente forma:

```
/**
 * Crear llave foranea de externa
 *
 * @param String $extSchema Esquema de la tabla externa
 * @param String $extTable tabla externa
 * @param String $nomenclador Nomenclador
 * @param String $field Campo de enlace de la foranea
 * @return void
* */
```

```
$adt = new ZendExt_Nomencladores_ADT();
$adt->addExtFK('public.tablel','nom_nomenclador4','campol','idpadre');
```

Parámetro1 (pextTable) => Tabla externa

Parámetro2 (pNomenclador) => Nombre del nomenclador

Parámetro3 (pextField) => Campo externo

Parámetro4 (pField) => Nombre del campo

g. Adicionar la presentación de un campo

Para llamar a la función que permite adicionar la presentación de un campo se hace de la siguiente forma:

```
/**
 * Adicionar la presentación de un campo.
 *
 * @param String $pTable. Nombre de la tabla.
 * @param String $pField Nombre del campo.
 * @param String $pComponent Define el componente de Ext para la presentación.
 * @param String $pRegEx Expresión regular.
 * @param String $pDesc Descripción del campo.
 * @param String $pVisible ¿Se verá o no?
 * @param String $pValores En el caso de los combos se especifican los valores del combo.
 * @return void
 **/
```

```
$adt = new ZendExt_Nomencladores_ADT();
      $adt->addPresentacion('nom_nomenclador3', 'idpadre',19, 'padre', 'expresion', 'descripcion',
'true', 'textfield');
```

Parámetro1 (pTable) => Nombre de la tabla

Parámetro2 (pField) => Nombre del campo

Parámetro3 (pLongitud) => Longitud

Parámetro4 (pNombre) => Nombre con el cual se va a mostrar

Parámetro5 (pRegEx) => La expresión regular

Parámetro6 (pDesc) => Descripción

Parámetro7 (pVisible) => Si es visible o no

Parámetro8 (pComponent) => Tipo de componente que se va a mostrar

Parámetro9 (pDisplayField) => Campo que se va a mostrar

Parámetro10 (pForeignTable) => Nombre del campo foráneo

Parámetro11 (pIdField) => El Id del campo

Utilización a nivel de interfaz

La gestión de los nomencladores se puede realizar a través de interfaz gráfica donde se pueden gestionar los conceptos o tablas para su posterior utilización, designándole una categoría, habilitándole si se va a explotar, y especificándole su condición de nomenclador arbólico.

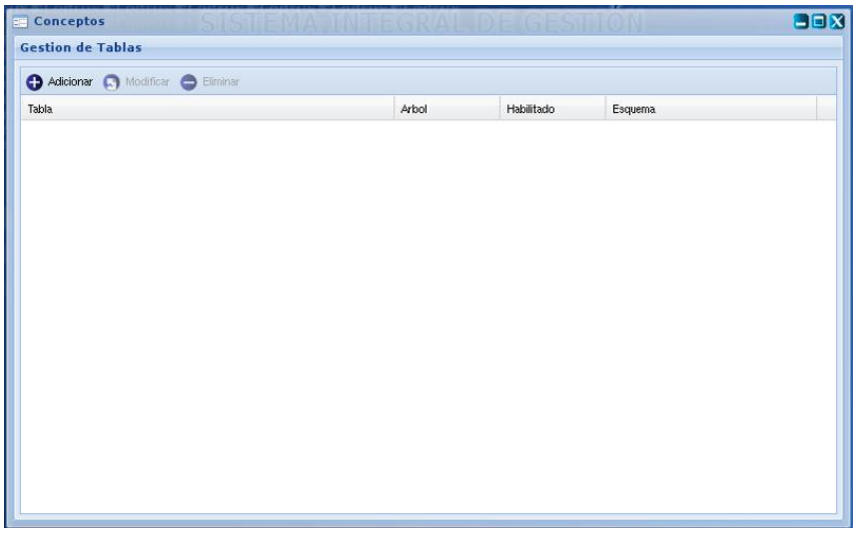

Figura 6. Interfaz para la gestión de tablas

Mediante la interfaz de gestión de elementos se brinda la posibilidad de adicionar datos físicos a un nomenclador creado.

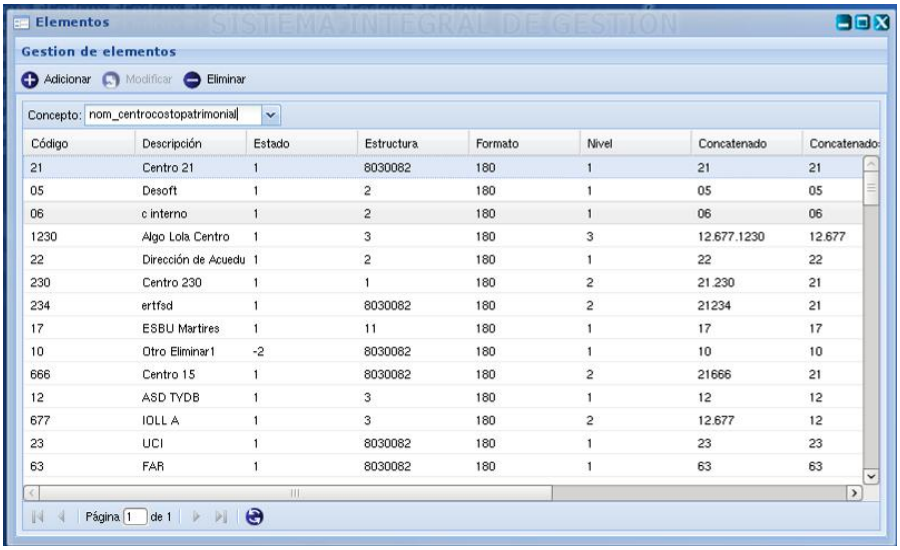

Figura 7. Interfaz para la gestión de elementos

La interfaz de gestión de estructura brinda la posibilidad de gestionar los atributos de un nomenclador gestionando el tipo de datos, tamaño, la expresión regular, si es llave primaria entre otros.

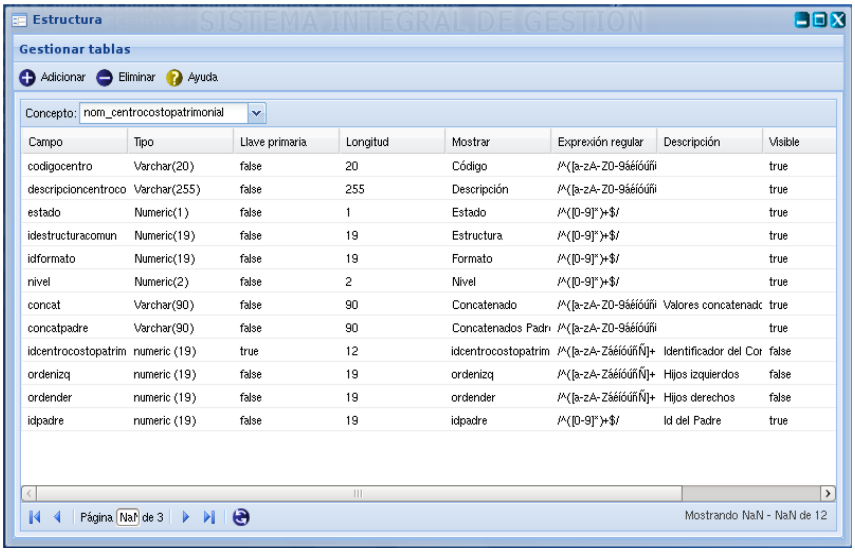

Figura 8. Interfaz para la gestión de campos

Ficheros de configuración

En el componente los datos específicos de las conexiones a datos están plenamente desacoplados de la lógica del mismo a través de un documento XML. Dentro del elemento destinado a la conexión se especifica los parámetros de la conexión global pasándole el nombre de la base de datos, el gestor y el número IP del servidor.

Luego se especifican cada uno de los esquemas de la bases de datos a los que se desea acceder con el componente en los que se deben especificar el nombre del mismo, el usuario de datos para eso y la contraseña.

Formato

```
<config>
    <conn bd="erp2" gestor="pgsql" host="10.12.171.233" />
    <esquemas>
        <datos nombre="mod_contabilidad" usuario="contabilidad"
            password="46555536511679391612 46536696292532146004 80542219257456921414 42802811280774029406" />
        <datos nombre="mod_capitalhumano" usuario="capitalhumano"
            password="97136064282477076115 80049613718796195286 36173593593928533558 28125082826412297065" />
        <datos nombre="mod_cobroypago" usuario="contabilidad"
            password="46555536511679391612 46536696292532146004 80542219257456921414 42802811280774029406" />
        <datos nombre="mod_caja" usuario="contabilidad"
            password="46555536511679391612 46536696292532146004 80542219257456921414 42802811280774029406" />
        <datos nombre="mod_finanzas" usuario="contabilidad"
            password="46555536511679391612 46536696292532146004 80542219257456921414 42802811280774029406" />
    </esquemas>
</config>
```

Caso de estudio

1. A nivel de interfaz

1.1 Crear tabla

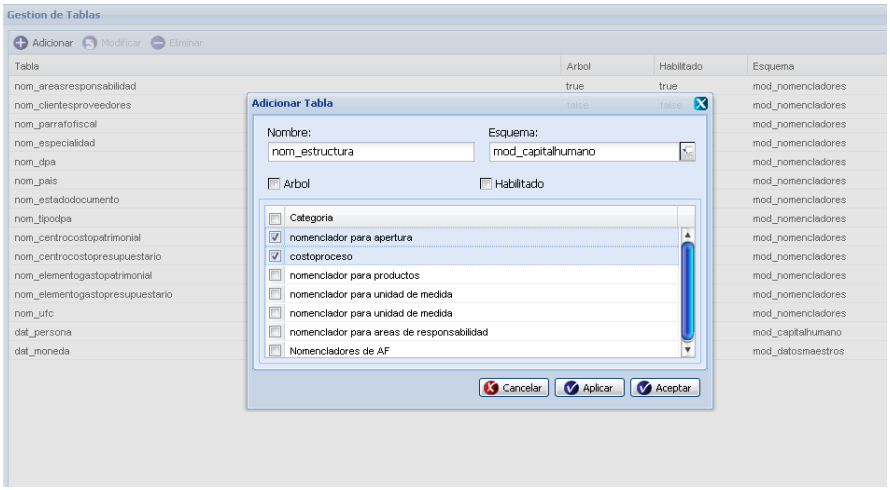

Figura 9. Interfaz para crear tablas

Crear el nomenclador **nom_estructura** para el esquema **mod_capitalhumano** habilitado para su uso y perteneciente a las categorías nomenclador para apertura y **costo_proceso**.

1.2 Eliminar tabla

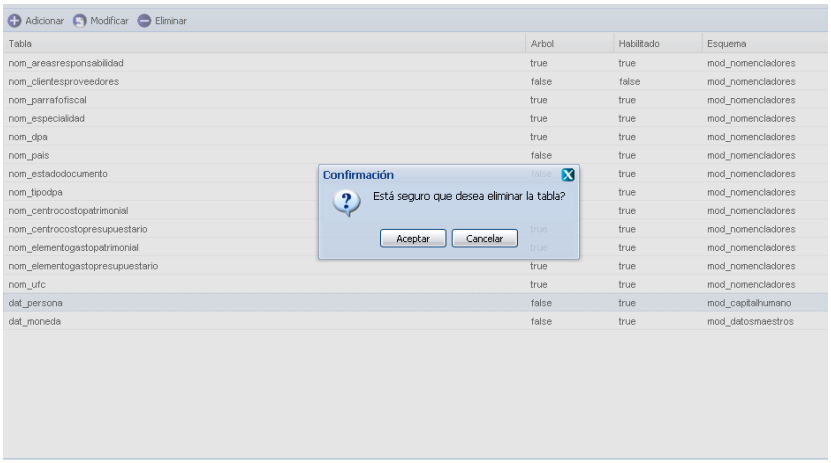

Figura 10. Interfaz para eliminar tablas

Eliminar la tabla dat_persona (previamente seleccionado).

2. A nivel de lógica

2.1 Adicionar tabla

```
$this->_adt = new ZendExt_Nomencladores_ADT();
$this->_adt->createTable ($nombre, $tree, $enabled ,$categorias, $esquema);
$this->_adt->addSecuencia($nombre,"id$nombre",$esquema);
parent :: init ();
```

Juego de datos

$nombre = “nom_estructura”

$tree = ‘true’

$enabled = ‘true’

$esquema = ‘mod_capitalhumano’

2.2 Eliminar tabla

```
$this->_adt = new ZendExt_Nomencladores_ADT();
$this->_adt->dropTable($table);
```

Juego de datos

$table = ‘dat_persona’

Caso de prueba

En la asignación que se muestra, se instancia la clase y se guarda en una tabla los resultados de los datos de los nomencladores. Para que sea incorrecto este código se pone nom_especialidad1, de esta forma como no se encuentra esa variable en la base de datos archivada, da un error o no se cumple esta línea de código.

Caso Correcto:

$especialidad = $obj->getForest ('nom_especialidad');

Caso incorrecto:

$especialidad = $obj->getForest ('nom_especialidad1');

Patrones usados

El patrón que se utilizó para la realización de ese componente es el Singleton para permitir que no se hicieran varias instancias de la misma clase.

3.5. Validación de la documentación

La documentación técnica del Marco de Trabajo es sumamente importante, tanto en el mantenimiento de la aplicación como en el uso de la misma por los desarrolladores y usuarios finales. Esta está validada por los diferentes proyectos productivos que utilizan este Marco de Trabajo tales como La Línea de Sistemas de Información Geográfica (GIS) y la Línea de Contabilidad Material.

3.6. Conclusiones

Con la descripción de estos componentes del Marco de Trabajo se conoció en profundidad el problema que resuelven cada uno de ellos, los objetivos principales, los requisitos que implementan, los escenarios que tienen, así como sus diferentes diagramas como el de clases, el de paquetes, el modelo de datos entre otros. Se explicó brevemente la arquitectura del Marco de Trabajo y se mostró la relación de todos los componentes de esta versión del Marco de Trabajo.

CONCLUSIONES

Se realizó un estudio detallado del estado del arte concluyendo con la apremiante necesidad de desarrollar una documentación a la medida para el Marco de Trabajo Sauxe. Se planteó la vía de solución teniendo en cuenta la metodología señalada. Se investigó sobre el impacto en la productividad del programador, integridad del sistema y usabilidad del software. Además se detallaron todos los escenarios y requisitos arquitectónicos de los 15 componentes de la versión 2.0 del Marco de Trabajo y con la descripción de estos componentes se conoció en profundidad el problema que resuelven cada uno de ellos, los objetivos principales, los requisitos que implementan, los escenarios que tienen, así como sus diferentes diagramas como el de clases, el de paquetes, el modelo de datos, entre otros. También se explicó brevemente la arquitectura del Marco de Trabajo y se dio a conocer la relación de todos los componentes de esta versión.

RECOMENDACIONES

Se recomienda que el resultado propuesto en la investigación sea implantado como uso cotidiano e introducido en el desarrollo de las aplicaciones web de gestión que se desarrollen sobre el Marco de Trabajo Sauxe.

Continuar actualizando este trabajo, en la medida que se avance en las nuevas versiones del Marco de Trabajo.

BIBLIOGRAFÍA

Altamirano, Ing. Alfonso Valdez. 2009. Comparativo de Entornos de Desarrollo Integrados. [En línea] 5 de 2009. http://ubicuos.com/wp-content/uploads/2009/05/comparativoides.pdf.

Alvarez, Miguel Angel. 2009. Code Igniter. [En línea] 23 de noviembre de 2009. http://www.desarrolloweb.com/articulos/codeigniter.html.

Ambler, Scott W. 2003-2009. [En línea] 2003-2009. http://www.agilemodeling.com/artifacts/packageDiagram.htm.

Badilla, Ricardo Montaño. 2010. Sistema ERP. Definición, funcionamiento, ventajas y desventajas. [En línea] 10 de febrero de 2010. http://www.gestiopolis.com/administracion-estrategia/erp-definicion-funcionamiento-ventajas-desventajas.htm.

Bareiro, Héctor A. y López, Ricardo G. 2007. Frameworks, conceptos.Comparativa entre Ruby on Rails, Kumbia y Symfony. [En línea] 12 de octubre de 2007. http://www.sicuma.uma.es/sicuma/independientes/argentina08/Bareiro-Lopez/index.htm.

Baryolo, Oiner Gomez. 2010. *MARCO DE TRABAJO PARA APLICACIONES WEB DE GESTIÓN.* Habana : s.n., 2010.

Baryolo, Olner Gómez y García Tejo, Darien. 2009. *XVI Fórum de Ciencia y Técnica. 2009.*

Baryolo, Oiner Gómez, y otros. 2008. *Plantilla Registro de la Propiedad intelectual(Sauxe).* 2008.

Boda, Pedro, y otros. 2009. Zend Framework Manual en español. [En línea] 14 de noviembre de 2009. http://www.google.com.cu/#hl=es&q=documentacion+de+zend+framework&start=60&sa=N&fp=bdc9a97574402bb1.

BuenMaster. 2007. Modelo Vista Controlador. [En línea] 14 de agosto de 2007. http://buenmaster.com/?a=536.

Casanovas, Josep. 2004. El Iceberg de la usabilidad . [En línea] 9 de septiembre de 2004. http://www.desarrolloweb.com/articulos/1622.php.

Comunidad de Desarrollo. 2006-2010. ExtJs. [En línea] 2006-2010. http://www.extjs.com/products/js/download.php?dl=extjs32&ref=extjsgreenbutton .

Comunidad de desarrollo. 2006-2010. ExtJs en español. [En línea] 2006-2010. http://extjs.es/.

Corzo, Giancarlo. 2008. ExtJS lo bueno, lo malo y lo feo. [En línea] 22 de octubre de 2008. http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/.

Diseño web y desarrollo de software. 2009. Software y programación a medida. [En línea] 2009. http://www.proyectosbds.com/software-y-programacion-a-medida/programacion-php-lamp-zf/mas-sobre-zend-framework/121/.

Figueroa, Pablo. 2007. [En línea] 2007. http://agamenon.uniandes.edu.co/~pfiguero/soo/Magister_Patrones/intropatrones.html.

Fumero, Ing. Dorisbel Muro, Lazo Ochoa, Ing. René y Lage Codorniu, Ing. Cesar. 2010. *Guía de referencia para el desarrollo de marcos de trabajo tecnológicos de sistemas en dominios de gestión.* Habana : s.n., 2010.

Giraldo, Luis y Zapata, Yuliana. 2005. Herramientas de desarrollo de Ingeniería de Software para Linux. [En línea] 24 de septiembre de 2005. http://hugolopez.phi.com.co/docs/download/file=Giraldo-Zapata-Herramientas%20de%20ISW.pdf,_id=17.

González, Mario. 2001. Arquitectura de Software. [En línea] 2001. http://www.ldc.usb.ve/~abianc/materias/ci4712/Software%20Architecture-kazman.pdf.

Gutiérrez, Javier J. 2009. ¿Qué es un framework web? [En línea] 2009. http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

Hooping.net. 2009. Glosario. [En línea] 2009. http://www.hooping.net/glossary/aplicaciones-web-146.aspx.

Iberia, Sescoi. 2010. Las empresas que gestionan proyectos necesitan soluciones ERP especializadas. [En línea] 2010. http://www.workplan-enterprise.com/fileadmin/pdf/myworkplan/WhitePaper_ERP-JobManag-2_SP.pdf.

Lemus, Juan Manuel. 2009. Modelado de datos e implementación de la base de datos . [En línea] 2009. http://www.maestrosdelweb.com/editorial/modelado-de-datos-e-implementacion-de-la-base-de-datos-primer-nivel-l5/.

Leopoldo, Carlos. 2007. Lo nuevo en CakePHP 1.2 y las diferencias con CakePHP 1.1. [En línea] 15 de octubre de 2007. http://techtastico.com/post/lo-nuevo-en-cakephp-12-y-las-diferencias-con-cakephp-11/.

—. 2007. Zend Framework, una introducción. [En línea] 27 de noviembre de 2007. http://techtastico.com/post/zend-framework-una-introduccion/.

—. 2007. Zend Framework, una introducción. [En línea] 27 de noviembre de 2007. http://techtastico.com/post/zend-framework-una-introduccion/.

Loaiza, Cuauhtémoc Rivera. 2000. Interfaz de usuario. [En línea] 23 de mayo de 2000. http://www.fismat.umich.mx/~crivera/tesis/node6.html.

Mata, Manel Pérez. 2009. ¿Qué es Doctrine ORM? [En línea] 3 de julio de 2009. http://www.tecnoretales.com/programacion/que-es-doctrine-orm/comment-page-1/.

Molpeceres, Alberto. 2002. Diseño de software con patrones. [En línea] 15 de octubre de 2002. http://www.javahispano.org/contenidos/es/patrones_de_software_parte_1/;jsessionid=9FA29892C8F873780374C1FF0F4AEE27.

Pensando en Red. 2009. ¿Por qué usar Symfony? [En línea] 7 de diciembre de 2009. http://www.pensandoenred.com/2009/12/07/%C2%BFpor-que-usar-symfony/.

Pérez, Mileidy Magalys Sarduy y Hernández Cisneros, Sergio. 2009. *Trabajo de Diploma para optar por el título de Ingeniero Informático.* Ciudad de La Habana : s.n., 2009.

Portal de Radio Habana Cuba. 2007. Seminario Nacional con directores de empresas, grupos y uniones. [En línea] 30 de agosto de 2007. http://www.radiohc.cu/espanol/economia/ago07/reglamento.htm.

Reynoso, Billy. 2005. Introducción a la Arquitectura de Software. [En línea] 2005. http://sophia.javeriana.edu.co/~cbustaca/Arquitectura%20Software/Presentaciones/arquitecturas_software02.pdf.

Reynoso, Carlos Billy. 2004. Introducción a la Arquitectura de Software. [En línea] marzo de 2004. http://www.willydev.net/descargas/prev/IntroArq.pdf.

Rojas, Maribel Ariza y Molina García, Juan Carlos. 2004. INTRODUCCIÓN Y PRINCIPIOS BÁSICOS DEL DESARROLLO DE SOFTWARE BASADO EN COMPONENTES. [En línea] 30 de septiembre de 2004. http://pegasus.javeriana.edu.co/~jcpymes/Docs/DSBC.pdf.

ServerGrove. 2010. Doctrine. [En línea] 2010. http://www.doctrine-project.org/.

Sierra, María. 2009. Trabajando con Visual Paradigm for UML. [En línea] 2009. http://personales.unican.es/ruizfr/is1/doc/lab/01/is1-p01-trans.pdf.

Symfony.es. 2009. Mejoras en la documentación de Symfony. [En línea] 25 de febrero de 2009. http://www.symfony.es/2009/02/25/mejoras-en-la-documentacion-de-symfony-2/.

Universidad Rey Juan carlos. 2009. Metodologias de desarrollo. [En línea] 2009. http://www.escet.urjc.es/~gtazon/IS/Metodologias.pdf.

Zend Technologies Ltd. 2006-2010. Introducing the Best Stack for your ZendFramework Apliccations. [En línea] 2006-2010. http://framework.zend.com/.

GLOSARIO DE TÉRMINOS

1. Extender: Enseñarle a los usuarios como a partir de lo que actualmente se tiene, pueden reutilizar los componentes.
2. MySQL: Sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.
3. Oracle: Sistema de gestión de base de datos relacional.
4. SQLite: Sistema de gestión de base de datos relacional.
5. SQL: Vulnerabilidad informática en el nivel de la validación de las entradas a la base de datos de una aplicación.
6. API: La interfaz de programación de aplicaciones es un conjunto de funciones residentes en bibliotecas.
7. Layouts: Son capas. Se utilizan para crear la maquetación de un sitio.
8. Swing: Es una librería gráfica para Java.
9. Safari: Navegador web de código cerrado.
10. CASE: Ingeniería de Software Asistida por Ordenador. Es un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación.
11. UML: Lenguaje Unificado de Modelado. Es un lenguaje para el desarrollo de software orientado a objetos, su propósito es visualizar, especificar, construir y documentar proyectos de software.
12. IDE's: Entornos de Desarrollo Integrados.
13. NetBeans: La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos.
14. Eclipse: Entorno de desarrollo integrado de código abierto multiplataforma.
15. IBM: International Business Machines (conocida coloquialmente como el Gigante Azul) es una empresa multinacional que fabrica y comercializa herramientas, programas y servicios relacionados con la informática.

16. .NET: Es un Marco de Trabajo de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.
17. Trigger: Un trigger o disparador en una Base de datos, es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción, actualización o borrado.
18. Script: Conjunto de instrucciones.
19. WSDL: Lenguaje de Descripción de Servicios Web, un formato XML que se utiliza para describir servicios web.
20. HTML: Lenguaje de Marcado de Hipertexto, es el lenguaje de marcado predominante para la elaboración de páginas web.
21. CSS: Hojas de estilos en cascada.
22. DHTML: HTML dinámico.
23. JS: Java Script es un lenguaje de scripting basado en objetos.
24. Ajax: Java Script asíncrono y XML, es una técnica de desarrollo web para crear aplicaciones interactivas.
25. XUL: Lenguaje basado en XML para la interfaz de usuario.
26. Rollback: Operación que devuelve a la base de datos a algún estado previo.
27. Tipo: Establecido en el fichero de expresiones regulares expressions.xml.