Universidad de las Ciencias Informáticas

Facultad 7

Trabajo de diploma para optar por el título

de

Ingeniero en Ciencias Informáticas

Título: Documentación técnica y ayuda del marco de desarrollo del ERP.

Autores:

Javier Limonta Neyra

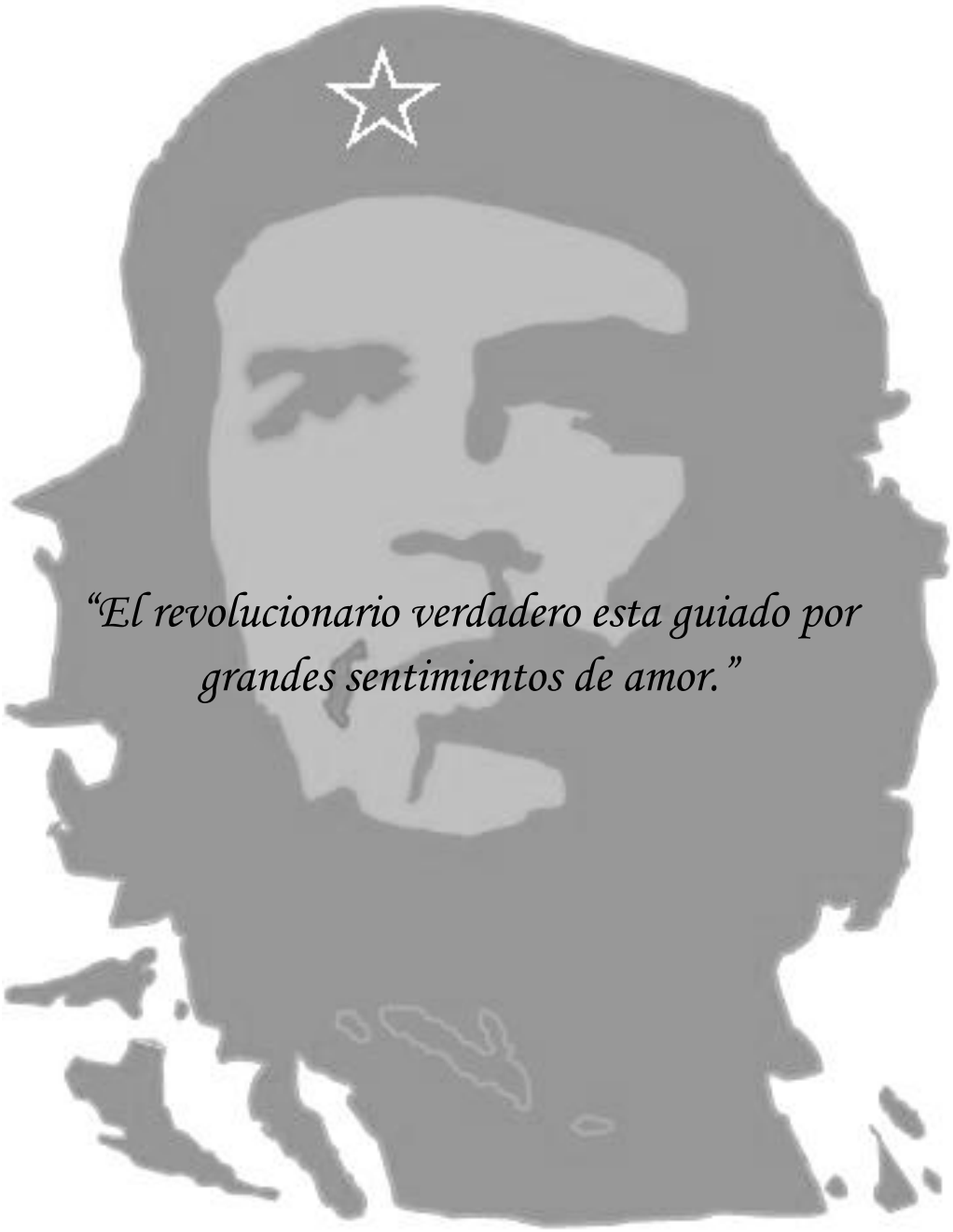
Rosabel Fernández Rivera

Tutor:

Ing. Yoandry Morejón Borbón

Cotutor:

Ing. René Lazo Ochoa



"El revolucionario verdadero esta guiado por grandes sentimientos de amor."

Rosabel:

Les agradezco a mis profesores de todos estos años de estudio, pues de una forma u otra cada uno de ellos contribuyó a mi formación profesional y personal.

A mi hermana Yarisleidis por su paciencia y ayuda, cada vez que tenía un problema con las clases. Además por ser mí segunda madre en la universidad

A mi novio por estar siempre presente, darme tanto amor y ayudarme a superarme cada día más.

A mis padres Juan y Rosabel y mi hermana Eilen por tratar de acortar la distancia que nos separaba y continuar dándome amor y aliento como si nos viéramos todos los días.

A los amigos del pre-universitario con los que compartí momentos importantes y dichosos, Danelia, Arianna, Yuleydis, Cadete, Daniellis, Elizabeth, Yanara, Franklin, Aroche, Darian y los demás, aunque no mencione sus nombres, por ser mis amigos en aquellos momentos y en estos.

A mis compañeros de estudio de estos 5 años por quererme con mis virtudes y defectos, por hacerme tan feliz la estancia en esta, nuestra segunda casa. Especialmente a Dayanis y Taymi.

A mi amiga Dairene que no nos vemos hace algún tiempo pero la quiero como a una hermana.

A las nuevas hermanas que encontré, Yeslayne, Dailyn, Aneyvis; por aconsejarme en la vida y los estudios y estar presentes siempre que las necesité.

No puede faltar el agradecimiento a mi compañero de tesis por su esfuerzo y gran trabajo para la realización del trabajo de tesis. A Borbon por ser un excelente tutor y a todos los nuevos amigos que encontré en el proyecto ERP, específicamente en la línea de arquitectura.

Un agradecimiento especial a Vinent, por ayudarme desinteresadamente a la realización del portal.

A toda mi familia por ser tan especial, a mis tíos, tías, primos y primas; muchas gracias por formar parte de mi vida y de quien soy.

A mi suegra por quererme como una hija.

También a la revolución por darme la oportunidad de ingresar a la universidad y forjarme como una Ingeniera en Ciencias Informáticas.

Javier:

Quiero agradecer ante todo a mi madre, la cual me ha dado su apoyo durante mi etapa como estudiante.

Agradecer a mi compañera de tesis por su arduo trabajo en el estudio y elaboración de este trabajo de diploma.

Les agradezco a todos aquellos profesores que me han enseñado a lo largo de estos cinco años contribuyendo con mi formación profesional.

A mis amigos que de una forma u otra han hecho mi estancia agradable en la universidad.

A todos mis compañeros de proyecto por haber hecho de este ultimo año, el mejor.

A la revolución por darme la oportunidad de ingresar a la universidad y forjarme como Ingeniero en Ciencias Informáticas.

Rosabel

Dedico esta tesis a mis padres, por apoyarme en todos estos años y darme fuerzas para superar los malos momentos. Por darme tanto amor y educación, por enseñarme a decidir mi camino y no hacerlo por mi.

A mis hermanas por hacerme tan feliz, y estar siempre conmigo dándome apoyo.

A mi novio que me ha hecho muy feliz en estos años. Que ha sabido quererme con mis virtudes y defectos, que me ha hecho parte de su vida. Y además me ha guiado en mi formación profesional.

A la memoria de mis abuelos, pues ayudaron a mi formación social y me dieron mucho amor.

A mis abuelas, que igualmente me han llenado de amor y en especial a mi abuela María, que me ha consentido, me ha orientado, y dado amor como si fuera su hija y no su nieta.

A mi tía Idalmis por ser una gran amiga.

En fin a todas las personas que de una forma u otra me han dado apoyo y amor en estos años de estudios.

Javier

A mi madre; la cual es merecedora de todos mis logros en esta vida y mi principal motivación.

A mis segundos padres; Alberto y Ana Luisa, por aconsejarme día a día y acogerme como otro de sus hijos.

A mis hermanos de la vida; Aliexer, Cesar y Edilberto, por su apoyo y amistad.

A mi sobrino; Jonatán, por servirme de inspiración a pesar de su corta edad.

A mi novia, por su amor incondicional, apoyo y paciencia.

A mis estimadas amigas; Adriana y Grette Leydi, por su apoyo, comprensión y consejos.

A mi compañeros de proyecto; por haber sido ante todo, amigos.

A todas las personas que de una forma u otra me han dado apoyo y amor en estos años de estudios.

A todos gracias.

Resumen

La documentación técnica del marco de trabajo del sistema de gestión integral Cedrux será apoyo fundamental para la construcción de soluciones integrales de referencia nacional. La clara, factible y 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 desarrolladores del sistema Cedrux. El principal objetivo que trae consigo una ordenada y concisa especificación de esta información, esta dada por la documentación técnica de los principales componentes del marco de trabajo del sistema Cedrux. La disposición de esta información es la principal fuente de apoyo para el desarrollo del sistema en cuestión, puesto que explica la implementación de los aspectos arquitectónicamente significativos y las funcionalidades propias del proceso de negocio. Además aporta un conjunto de innovaciones tecnológicas en el lenguaje PHP, para la comunidad de desarrollo de dicho lenguaje en la Universidad de las Ciencias Informáticas, incrementando el trabajo colaborativo en la misma mediante la socialización de la información y el conocimiento generado a la largo del desarrollo de sistema integral de gestión Cedrux.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

Introducción

En el sector empresarial la toma de decisiones en las organizaciones se vuelve compleja, pues se debe considerar gran cantidad de información, la cual debe representar los datos reales de lo que está ocurriendo en una empresa, por lo que directivos y administradores deben hacer uso de una herramienta que les permita analizar esa información y finalmente tomar la decisión correcta sobre cómo dirigir la empresa. Los ERP (Enterprise Resource Planning o Panificación de Recursos Empresariales) son soluciones informáticas cuyo objetivo es gestionar la información a través de las diferentes áreas de una empresa (fabricación, compras, recursos humanos, logística, etc.) para agilizar tareas, mejorar los procesos de producción y reducir costes.

Cuba a pesar de ser un país del tercer mundo, no quiere mantenerse alejada de la revolución tecnológica a la cual se ha girado el mundo actual. Es por ello que entre sus objetivos por lograr un avance tecnológico se ha trazado la meta de desarrollar un sistema de gestión integral Cedrux que traiga consigo la posibilidad de perfeccionar el sistema de gestión de la empresa socialista cubana y obtener como resultado final una tecnología tipo, basada en un marco legal beneficiario para Cuba, sin riesgos jurídicos y con garantía de independencia tecnológica; impactando en la calidad de los productos de la empresa de software cubana mediante la obtención de productos competitivos. Tecnología que sea competitiva a nivel mundial, que reutilice los principales avances del mundo, y con la inclusión del conocimiento endógeno que tenga en cuenta nuestras características nacionales.

En este tipo de sistema, por su extensión, durante el proceso de desarrollo, se conciben un sin número de artefactos en cada una de las áreas de trabajo. La línea de Arquitectura del proyecto ERP-Cuba , es la encargada de construir las tecnologías que posteriormente serán usadas por las diferentes áreas de desarrollo del sistema, durante este proceso se generan componentes y documentos los cuales tienen una importancia medular para todo el sistema Cedrux y por tanto resulta imprescindible mantenerlos ordenados y centralizados.

Actualmente se ha generado conocimiento en la Arquitectura del proyecto ERP-Cuba, pero en ocasiones su desconocimiento provoca incremento de dudas a la hora de reutilizar sus aspectos técnicos, lo que conlleva a un atraso en las líneas de desarrollo debido a que el tiempo de estudio es más prolongado.

Esta situación está dada porque al no tener el marco de trabajo estrictamente documentado, el cual no es mas que “el esqueleto de una aplicación que debe ser adaptado a necesidades concretas por el programador de la aplicación”, se hace difícil entender las aspectos tecnológicos del mismo para la posterior implementación del sistema Cedrux; así como su reutilización por otros proyectos que implementan soluciones integrales. Además si realmente el marco de trabajo es el apropiado, se ha de saber cómo utilizarlo, acomodarlo, especializarlo y posiblemente extenderlo, para integrarlo en su implementación concreta.

El problema de la documentación se debe principalmente a que no existe una forma clara de especificar los marcos de trabajo. En la mayoría de los casos esta documentación no esta unificada, es pobre o se encuentra dispersa. La razón fundamental de estos hechos es que los marcos de trabajos existentes han nacido normalmente como generalización de aplicaciones ya realizadas, cuya estructura ha sido extraída y recompuesta como un marco de trabajo para su posterior reutilización. Los marcos de trabajos de nueva creación, se enfrentan a la falta de estandarización de su documentación, por lo que dependen en gran medida de la voluntad y conocimientos de sus desarrolladores. (Lidia Fuentes)

Debido a la situación problémica que trae aparejada la falta de documentación del marco de trabajo, se plantea como **problema a resolver** que la documentación técnica del marco de trabajo del sistema Cedrux no garantiza su fácil entendimiento y mantenimiento.

Para darle solución al problema que se plantea el **objeto de estudio** lo constituye la Arquitectura de software cuyo **campo de acción** queda enmarcado en la formalización técnica de Frameworks y Patrones Arquitectónicos.

Para la realización del trabajo de diploma se traza como **objetivo general** formalizar la documentación técnica del marco de trabajo del sistema Cedrux de manera práctica e integral, que garantice su fácil entendimiento y mantenimiento. Para darle cumplimiento al objetivo general planteado se trazan los siguientes **objetivos específicos:**

- ✓ Describir y documentar los componentes que conforman la primera versión del marco de trabajo.
- ✓ Construir el manual de instalación del marco de trabajo.

- ✓ Construir un portal para la disposición de la documentación técnica del marco de trabajo.

Hipótesis

Si se realiza la formalización técnica del marco del marco de trabajo del sistema de gestión integral Cedrux, de manera práctica e integral, entonces se garantizará su fácil entendimiento y mantenimiento.

Las **tareas de la investigación** para darle solución a los objetivos específicos trazados son:

- ✓ Realizar un estudio del arte de la documentación de los marcos de trabajo, el trabajo colaborativo y sus herramientas para la socialización de la información y el conocimiento.
- ✓ Estudiar el flujo de implementación de los componentes del marco de trabajo del sistema Cedrux.
- ✓ Realizar la descripción técnica de los componentes de la primera versión del marco de trabajo.
- ✓ Elaborar el manual de instalación del marco de trabajo.
- ✓ Estudiar los distintos CMS para la construcción un portal para la disposición de la documentación técnica del marco de trabajo del sistema de gestión Cedrux.
- ✓ Definir CMS para la construcción del portal para la disposición de la documentación técnica del marco de trabajo del sistema de gestión Cedrux.
- ✓ Construir el portal para la disposición de la documentación técnica del marco de trabajo del sistema de gestión Cedrux.

Métodos científicos de investigación

Métodos teóricos

Análisis histórico – lógico

Se realiza un estudio del arte de la documentación de los frameworks en el mundo, para definir si existe una guía, metodología o estructura oficial de documentarlos. Se investiga sobre las herramientas asociadas al trabajo colaborativo para la socialización de la información y el conocimiento.

Resultados Esperados

Con la creación de un Portal de para la disposición de la documentación técnica del marco de trabajo, se incrementará el trabajo colaborativo en la Universidad de las Ciencias Informáticas, específicamente la socialización de las innovaciones tecnológicas realizadas en lenguaje PHP para la comunidad de desarrollo de este lenguaje en dicha universidad. Será una fuente de apoyo para la construcción del sistema Cedrux, así como para los interesados en reutilizar el marco de trabajo.

Estructura del trabajo

El trabajo esta estructurado en **tres capítulos**.

Capítulo 1

El primer capítulo incluye un estado del arte sobre los de marcos de trabajo y la documentación técnica de los mismos. Se aborda sobre el trabajo colaborativo y sus herramientas para la socialización de información y conocimiento.

Capítulo 2

El segundo capítulo incluye el manual de instalación del marco de trabajo y la descripción técnica de los componentes de la primera versión del marco de trabajo.

Capítulo 3

El tercer capítulo incluye la construcción del portal para la disposición de la documentación técnica del marco de trabajo.

1

CAPÍTULO I Fundamentación teórica

1.1 Introducción

En la actualidad muchos países desarrollan sistemas para la gestión empresarial, para la automatización e integración de prácticas de negocio asociadas a aspectos operativos o productivos de una empresa. Este tipo de sistemas suelen presentar una arquitectura modular, donde cada módulo gestiona las funciones de un área empresarial diferente, como pueden ser: nóminas, finanzas, gestión de proyectos, sistema de gestión geográfica, contabilidad, logística. Estos sistemas son integrales, es decir, una agrupación de todos los módulos que los componen, y que agrupan a su vez todos los procesos de gestión de una empresa.

La implementación de estos es muy compleja, puesto que la gestión empresarial engloba un sin número de áreas que gestionan gran cantidad de información, y en algunos casos, información confidencial. Los marcos de trabajo utilizados en estos sistemas tienen implementaciones de alta complejidad, las cuales pueden ser extensibles y recomendadas para su reutilización en soluciones de esta índole.

La incorporación de frameworks de desarrollo al marco de trabajo de un sistema, es de vital importancia puesto que su utilización, trae ventajas considerables para los desarrolladores, debido a que no necesitan plantearse una estructura global de la aplicación, sino que el framework le proporciona un esqueleto que hay que "rellenar". Además facilita la colaboración. Cualquiera que haya tenido que "pelearse" con el código fuente de otro programador (o incluso con el propio, pasado algún tiempo) sabrá lo difícil que es entenderlo y modificarlo; por tanto, todo lo que sea definir y estandarizar va a ahorrar tiempo y trabajo a los desarrollos colaborativos.

Es más fácil encontrar herramientas (utilidades, librerías) adaptadas al framework concreto para facilitar el desarrollo. (reinisch) Alta productividad, extensibilidad, facilidad del mantenimiento, reusabilidad. Los desarrolladores se concentran en la lógica solamente. EL marco conecta las capas de forma automática y extensible, mediante el uso de los aspectos significativamente arquitectónicos como:

- ✓ Excepciones
- ✓ Validación en el servidor
- ✓ Concurrencia
- ✓ Cargas y cacheo de variables y nomencladores globales
- ✓ Menú, multilenguaje, apariencia

1.2 Clasificación de los marcos de trabajo

Los marcos de trabajo se pueden dividir según su aplicabilidad en horizontales y verticales, de acuerdo al tipo de aplicación que acometan. Entre los horizontales existen aquellos marcos de trabajo dedicados a modelar infraestructuras de comunicaciones, las interfaces de usuario, los entornos visuales y, en general, cualquiera de los aspectos relacionados con el sistema subyacente.

Dentro de los marcos de trabajo horizontales merecen especial atención los denominados Marcos de Trabajo Distribuidos (Middleware Application Frameworks), diseñados para integrar componentes y aplicaciones en ambientes distribuidos, permitiendo altos niveles de modularidad y reutilización en el desarrollo de nuevas aplicaciones, y aislando la mayor parte de las dificultades conceptuales y técnicas que conlleva la construcción de aplicaciones distribuidas basadas en componentes. En la terminología de la programación orientada a componentes, este término es sinónimo de Plataforma de Componentes Distribuidos. Ejemplos de estos marcos de trabajo distribuidos son CORBA, ActiveX/OLE/COM, JavaBeans, ACE, Hector o Aglets.

Por otro lado, los marcos de trabajo verticales son aquellos desarrollados específicamente para un dominio de aplicación concreto, y cubren un amplio espectro de aplicaciones, desde las telecomunicaciones, la fabricación, o los servicios telemáticos avanzados y la multimedia. Por ser muy específicos y requerir un conocimiento muy preciso de los dominios de aplicación resultan los más costosos de desarrollar, y por tanto los más caros en caso de ser comercializados.

También destacaremos otros marcos de trabajo de especial relevancia dentro de la programación orientada a componentes, que son los denominados Marcos de Trabajo para Componentes (Components Frameworks). Se trata de marcos de trabajo tanto verticales como horizontales, pero que están realizados exclusivamente para el desarrollo de aplicaciones basadas en componentes reutilizables, y por tanto presentan características especiales para tratar los problemas específicos que

plantean dichos componentes (composición tardía, extensibilidad, etc.). En general, pueden definirse como una implementación concreta de uno o más patrones de diseño mediante componentes reutilizables, realizado a medida para un dominio de uso específico. Desarrollados sobre una plataforma de componentes concreta, estos marcos de trabajo suponen el siguiente nivel al de los componentes base para el desarrollo de aplicaciones. OpenDoc y BlackBox son dos de los pocos marcos de trabajo para componentes visuales que existen actualmente. (Lidia Fuentes)

La **documentación técnica** de los marcos de trabajo juega un papel primordial, puesto que representa la mayor fuente de consulta de la implementación de un producto.

1.3 Documentación técnica

“La documentación técnica de un producto proporciona una información rápida, clara, concisa y fiable tanto para el fabricante como para el cliente. Es un signo evidente de la calidad de un producto”. (reinisch)

Sería preciso disponer de mecanismos y herramientas para describir de forma unificada y completa la arquitectura de un marco de trabajo, los problemas para los que está diseñado, y la forma en la que hay que utilizarlo. Existen diferentes propuestas en este sentido, que pasamos a describir.

En primer lugar, algunos autores proponen la construcción de entornos gráficos que permiten ‘navegar’ por la arquitectura del marco de trabajo, consultando en cada uno de sus puntos la información que el usuario considera relevante. La representación de la arquitectura se realiza de forma gráfica, ya sea mediante grafos o mediante secuencias de mensajes. Estas propuestas aportan un conocimiento más profundo del marco de trabajo, aunque no definen una metodología que permita utilizar ese conocimiento para adaptar el marco de trabajo a la nueva aplicación. Su utilidad queda por tanto reducida a la fase de identificación del marco de trabajo.
Los contratos de reutilización pueden ser también utilizados en estos entornos para definir la forma en la que pueden cooperar los diseñadores del marco de trabajo y los encargados de adaptarlo o extenderlo. Este enfoque define la información necesaria para documentar adecuadamente un marco de trabajo. Así, se deben documentar cada una de las clases con sus métodos, indicando los que se consideran puntos de entrada al marco de trabajo, y las relaciones de herencia, composición, etc. Por otro

lado, las extensiones que se realicen a un marco de trabajo deben documentarse siguiendo el mismo método, incluyéndose en la documentación final. (Lidia Fuentes)

A lo largo de todo su ciclo de vida, un producto siempre debe ir acompañado de documentación técnica de calidad: empezando por la documentación del proceso de desarrollo; pasando por la información y manuales de producto y finalizando con un entregable completo para el usuario final.

La documentación técnica debe considerarse una parte más del producto y adquiere una especial relevancia en la introducción de cualquier producto en el mercado. La importancia de generar una documentación técnica de calidad reside en varios factores:

Jurídico

Cada vez son más los sectores donde existe una normativa específica en cuanto a procesos de fabricación, obligación de instrucción y responsabilidad del fabricante, etc. (normativas europeas, leyes nacionales, normas y directrices).

Competitivo

La documentación técnica forma parte del producto y permite que el cliente extraiga de él la máxima eficiencia y rentabilidad. Presentar las características y uso de un producto de una manera adecuada es un signo claro de calidad y facilita la elección del producto frente a la competencia.

Técnico

Los productos, cada vez más complejos y con mayores prestaciones, requieren una mejor explicación de todas sus funciones y características.

Económico

La globalización de la economía y el comercio internacional exige una documentación técnica lo más clara, explicativa y adaptada al público destinatario (importancia de la traducción técnica). Es el medio más eficaz para lograr introducirse en nuevos mercados y diferenciarse de la competencia.

Marketing-Ventas

La documentación técnica se ha convertido en uno de los instrumentos de marketing más efectivos. Si está bien elaborada, contribuye a la fidelización del usuario final. Si la documentación es en cambio defectuosa, no sólo puede llevar a la pérdida de clientes finales, sino también a fuertes indemnizaciones por responsabilidad de producto.

La primera y principal pregunta que se debe plantear para elaborar correctamente cualquier documentación debe ser: ¿quién utilizará esa documentación? Una definición precisa e inequívoca del grupo receptor evitará errores en la documentación elaborada y reducirá posibles costes sucesivos de servicio y garantía al cliente. (reinisch)

1.4 Tipos de documentación

La documentación que se entrega al cliente se divide claramente en dos categorías, interna y externa:

- Interna: Es aquella que se crea en el mismo código, ya puede ser en forma de comentarios o de archivos de información dentro de la aplicación.
- Externa: Es aquella que se escribe en cuadernos o libros, totalmente ajena a la aplicación en si. Dentro de esta categoría también se encuentra la ayuda electrónica.

1. La guía técnica

En la guía técnica o manual técnico se reflejan el diseño del proyecto, la codificación de la aplicación y las pruebas realizadas para su correcto funcionamiento. Generalmente este documento esta diseñado para personas con conocimientos de informática, generalmente programadores.

El principal objetivo es el de facilitar el desarrollo, corrección y futuro mantenimiento de la aplicación de una forma rápida y fácil.

Esta guía esta compuesta por tres apartados claramente diferenciados:

- Cuaderno de carga: Es donde queda reflejada la solución o diseño de la aplicación.
  Esta parte de la guía es únicamente destinada a los programadores. Debe estar realizado de tal forma que permita la división del trabajo
- Programa fuente: Es donde se incluye la codificación realizada por los programadores. Este documento puede tener, a su vez, otra documentación para su mejor comprensión y puede ser de gran ayuda para el mantenimiento o desarrollo mejorado de la aplicación. Este documento debe tener una gran claridad en su escritura para su fácil comprensión.

- Pruebas: es el documento donde se especifican el tipo de pruebas realizadas a lo largo de todo el proyecto y los resultados obtenidos.

2. La guía de uso

Es lo que comúnmente llamamos el manual del usuario. Contiene la información necesaria para que los usuarios utilicen correctamente la aplicación.

Este documento se hace desde la guía técnica pero se suprimen los tecnicismos y se presenta de forma que sea entendible para el usuario que no sea experto en informática.

Un punto a tener en cuenta en su creación es que no debe hacer referencia a ningún apartado de la guía técnica y en el caso de que se haga uso de algún tecnicismo debe ir acompañado de un glosario al final de la misma para su fácil comprensión.

3. La guía de instalación

Es la guía que contiene la información necesaria para implementar dicha aplicación. Dentro de este documento se encuentran las instrucciones para la puesta en marcha del sistema y las normas de utilización del mismo.

Dentro de las normas de utilización se incluyen también las normas de seguridad, tanto las físicas como las referentes al acceso a la información.

Para la disposición de la **documentación técnica** de un producto, se utiliza el trabajo colaborativo como medio de socialización de esta información. (técnica, 2006)

Para la realización de la documentación técnica se va a realizar documentación interna y externa, pues esta va a ser utilizada tanto por los desarrolladores como por los usuarios finales.

1.5 Trabajo colaborativo

En la actualidad existen proyectos, empresas y organizaciones que socializan la información de sus productos e ideas desarrollando el trabajo colaborativo; ejemplo:

- ✓**Synergeia** (entorno gratuito para el trabajo cooperativo on-line y que corresponde una adaptación del BSCW [Basic Support for Cooperative Work])

Synergeia es un sistema de software desarrollado en el ITCOLE proyecto de investigación financiado por la Unión Europea en 2001-2003 para el trabajo colaborativo en las aulas de las escuelas. (Itcole, 2007)

- ✓**EFaber** (grupo de ingenieros informáticos y físicos) brinda un sistema intranet para la Gestión de proyectos, trabajo colaborativo y gestión de conocimiento dentro de una empresa. (eFaber, 2009)

- ✓**EDUTEKA** es un Portal Educativo gratuito actualizado mensualmente desde Cali, Colombia, por la Fundación Gabriel Piedrahita Uribe que tiene como misión desarrollar en los jóvenes un conjunto nuevo de competencias, muchas de ellas asociadas a la rápida evolución de las TIC. (Fundación Gabriel Piedrahita Uribe, 2009)

El **trabajo colaborativo** se conoce como procesos intencionales de un grupo para alcanzar objetivos específicos, más herramientas diseñadas para dar soporte y facilitar el trabajo.

En el marco de una organización, el trabajo en grupo con soporte tecnológico se presenta como un conjunto de estrategias tendientes a maximizar los resultados y minimizar la pérdida de tiempo e información en beneficio de los objetivos organizacionales.

El mayor desafío es lograr la motivación y participación activa del recurso humano. Además deben tenerse en cuenta los aspectos tecnológico, económico y las políticas de la organización. Trabajo colaborativo o groupware son palabras para designar el entorno en el cual todos los participantes del proyecto trabajan, colaboran y se ayudan para la realización del proyecto. (Wik09)

Los programas Informáticos Colaborativos o Groupware se refiere a los Programas informáticos que incorporan en un solo proyecto de varios usuarios que se encuentran en diversas estaciones de trabajo, conectadas a través de una red. El groupware se puede dividir en tres categorías: herramientas de colaboración-comunicación, herramientas de conferencia y herramientas de gestión colaborativa o en grupo. (Wik091)

1.5.1 Elementos del trabajo colaborativo

- ✓ Objetivos: los mismos de la organización; particulares, bien definidos y mesurables
- ✓ Ambiente: controlado y cerrado.
- ✓ Motivación: la persona es convencida por la organización.
- ✓ Tipo de proceso: se formaliza el proceso grupal.
- ✓ Aporte individual: conocimiento y experiencia personal al servicio de los interesares organizacionales.
- ✓Pasos del proceso grupal: se deben definir claramente y previamente. (Wik09)

1.5.2 Herramientas para el trabajo colaborativo

Las herramientas para el trabajo colaborativo propician el intercambio de información y la gestión del conocimiento en la red. Ello es posible, dado que internet ha dejado de ser un conjunto de páginas web donde se consulta información o se intercambia mensajes. Ha surgido un nuevo paradigma al que se denomina Web 2.0, donde la filosofía es convertir el ciberespacio en un Web de todos, por todos y para todos. (Ledo, y otros, 2008)

Categorías de los groupware

1. **Herramientas de comunicación electrónica** que envían mensajes, archivos, datos o documentos entre personas y facilitan la compartición de información (colaboración asincrónica), como por ejemplo:

- ✓ Correo electrónico.
- ✓ Correo de voz.
- ✓ Publicación en Web.

2. **Herramientas de conferencia** que facilitan la compartición de información, de forma interactiva (colaboración síncrona), como por ejemplo:

- ✓ Conferencia de datos- PC en red que comparten un espacio de representación compartido que cada usuario puede modificar.
- ✓ Conferencias de voz- teléfonos que permiten interactuar a los participantes.
- ✓ Conferencias de video (o audio conferencia)- PC en red que comparten señales de audio o video.
- ✓ Salas de chat o mensajería instantánea- una plataforma de discusión que facilita el intercambio inmediato de mensajes.

- ✓ Sistemas para facilitar reuniones- un sistema de conferencias integrado en una sala. Estas salas suelen disponer de un avanzado sistema de sonido y presentación que permite una mejor interacción entre participantes en una misma sala o entre salas separadas. Ejemplos de ello son los Sistemas de soporte de decisiones.

3. **Herramientas de gestión colaborativa** que facilitan las actividades del grupo, como por ejemplo:

- ✓ Calendarios electrónicos: para acordar fechas de eventos y automáticamente enviar notificaciones y recordatorios a los participantes.
- ✓ Sistemas de gestión de proyectos: para organizar y hacer seguimiento de las acciones en un proyecto hasta que se finaliza.
- ✓ Sistemas de control de flujo de actividad: para gestionar tareas y documentos en un proceso organizado de forma estructurada (burocracia).
- ✓ Sistema de control del conocimiento: para recoger, organizar, gestionar y compartir varios tipos de información.
- ✓ Sistemas de soporte de redes sociales: para organizar las relaciones de los grupos. (Altamar)

Otra herramienta utilizada para el trabajo colaborativo son las wiki, provenientes del hawaiano wiki, que significa rápido; son instrumentos que permiten la participación de varios usuarios creando, editando, modificando y vigilando artículos existentes en un sitio Web.

Otra herramienta utilizada es el Gforge; herramienta para el desarrollo de software en forma comunitaria que permite organizar y administrar gran cantidades de proyectos, proporciona un conjunto integrado de herramientas que facilitan el trabajo en colaboración, y, en concreto, la gestión de proyectos de software libre que pueden acceder a diversos servicios:

1. Proporciona un entorno configurable con control de versiones.

2. Herramientas para comunicación entre desarrollos y servidor web por proyecto.

3. Permite a los miembros del equipo desarrollar una base de conocimiento compartida del proyecto.

Ventajas de Gforge

- ✓ Permite centralizar y homogeneizar la gestión de proyectos.
- ✓ Es una página única.

- ✓ Se consigue aumento de productividad.
- ✓ Se tienen herramientas comunes a toda la empresa o departamento.
- ✓ Centralización de los recursos técnicos en un servidor (en vez de tener que soportar múltiples máquinas por proyecto).

Uso de Gforge

Actualmente, se están gestionando aproximadamente alrededor de 80000 proyectos a través de Gforge y miles de colaboradores participan en él todos los días. Por lo menos 93 sitios web utilizan este sistema. Estos son entre otros:

- Desarrollo Colaborativo RINDE (Red Nacional de Integración y Desarrollo de Software Libre de la República Bolivariana de Venezuela).
- Fundacite – Mérida.
- Philips.
- NASA Goddard Space Flight Center.
- NOAA (National Oceanic & Atmospheric Administration).
- National Science Digital Library.
- DARPA (Defense Advanced Research Projects Agency).

Un Sistema de gestión de contenidos (Content Management System en inglés, abreviado CMS) es un programa que permite la creación y administración de contenidos por parte de los usuarios principalmente en páginas web, permitiendo manejar de manera independiente el contenido y el diseño, es de fácil edición y administración, donde los usuarios participan de forma interactiva y colaborativa en la creación de un producto de interés para un grupo, institución o comunidad de práctica.

1.5.3 CMS

El CMS, o Sistema de Gestión de Contenidos, es un concepto desarrollado en los últimos años, a la par de la evolución sufrida en Internet como herramienta de comunicación y medio para realizar negocios.

Básicamente, el CMS integra una serie de herramientas que permiten mantener un Sitio Web fácilmente, tales como la actualización de contenidos y novedades, calendarios, agendas de eventos, notas, etc.

Un Sitio Web desarrollado bajo el concepto CMS, brinda a la empresa la autonomía de decidir cuándo y de qué modo se ingresan los nuevos contenidos, basándose en un amigable software integrador que simplifica enormemente dichas tareas.

Ventajas:

- Crecimiento garantizado: Si su empresa crece, el agregado de contenido en su sitio ya no es un problema.
- Bajo costo de mantenimiento: El costo de mantenimiento cae, como consecuencia de la autogestión de las actualizaciones en el Sitio Web.
- Reduzca el tiempo de actualización: En el mismo plazo que emplea para reunir la información, tenga su sitio Web actualizado.
- Nuevas funcionalidades: Incorpore sobre la misma plataforma nuevas funcionalidades para su Sitio Web (banners, buscador dinámico, nuevas secciones). (TechneZero)

Hay multitud de CMS diferentes. Los podemos agrupar según el tipo de sitio que permiten gestionar.

A continuación se muestran los más representativos:

- **Genéricos**: Ofrecen la plataforma necesaria para desarrollar e implementar aplicaciones que den solución a necesidades específicas. Pueden servir para construir soluciones de gestión de contenidos, para soluciones de comercio electrónico, blogs, portales,... Ejemplos: Zope, OpenCMS, Typo3, Apache lenya.
- **Foros**: sitio que permite la discusión en línea donde los usuarios pueden reunirse y discutir temas en los que están interesados. Ejemplos: phpBB, MyBB.
- **Blogs**: Publicación de noticias o artículos en orden cronológico con espacio para comentarios y discusión. Ejemplos: WordPress, Typo.
- **Wikis**: Sitio web dónde todos los usuarios pueden colaborar en los artículos, aportando información o reescribiéndola. También permite espacio para discusiones. Indicado para material que irá evolucionando con el tiempo. Ejemplos: Mediawiki, Tikiwiki.
- **eCommerce**: Son Sitios web para comercio electrónico.

- **Portal**: Sitio web con contenido y funcionalidad diversa que sirve como fuente de información o como soporte a una comunidad. Ejemplos: PHPNuke, Postnuke, Drupal, Plone.
- **Galería**: Permite administrar y generar automáticamente un portal o sitio web que muestra contenido audiovisual, normalmente imágenes. Ejemplo: Gallery.
- **e-Learning**: Sirve para la enseñanza de conocimientos. Los usuarios son los profesores y estudiantes, tenemos aulas virtuales donde se ponen a disposición el material del curso,.... La publicación de un contenido por un profesor es la puesta a disposición de los estudiantes, en un aula virtual, de ese contenido. Ejemplo: Moodle.
- **Publicaciones digitales**: son plataformas especialmente diseñadas teniendo en cuenta las necesidades de las publicaciones digitales, tales como periódicos, revistas, etc. Ejemplo: ePrints. (Franco, 2008)

1.5.4 CMS más utilizados

Todos los CMS brindan un sin número de entornos, opciones, funcionalidades que satisfacen y cumplen con el objetivo fundamental de los que desarrollan el trabajo colaborativo. Todos presentan una comunidad donde se debaten y dan sugerencias para el incremento potencial de estos.

Drupal

Drupal, es un sistema de gestión de contenido para sitios Web. Permite publicar artículos, imágenes, u otros archivos y servicios añadidos como foros, encuestas, votaciones, blogs y administración de usuarios y permisos. Drupal es un sistema dinámico: en lugar de almacenar sus contenidos en archivos estáticos en el sistema de ficheros del servidor de forma fija, el contenido textual de las páginas y otras configuraciones son almacenados en una base de datos y se editan utilizando un entorno Web incluido en el producto

Joomla

Joomla es un sistema de administración de contenidos de código abierto construido con PHP bajo una licencia GPL. Este administrador de contenidos se usa para publicar en Internet e intranets utilizando una base de datos MySQL. En Joomla se incluyen características como: hacer caché de páginas para mejorar el rendimiento, indexa

miento web, feed RSS, versiones imprimibles de páginas, flash con noticias, blogs, foros, polls (encuestas), calendarios, búsqueda en el sitio web, e internacionalización del lenguaje. Su nombre es una pronunciación fonética para anglófonos de la palabra swahili jumla que significa "todos juntos" o "como un todo". Se escogió como una reflexión del compromiso del grupo de desarrolladores y la comunidad del proyecto.

Ezpublish

Ezpublish es un moderno CMS (Content Management System) de código abierto desarrollado y distribuido por la compañía noruega EZ Systems. Es una aplicación que ofrece soluciones profesionales para sitios Web dinámicos. Por ese motivo puede ser usado para construir cualquier tipo de proyecto, desde sitios personales a sitios corporativos que requieran diferentes perfiles de acceso, tiendas online, foros de discusión y otras funciones complejas.

WordPress

WordPress es un sistema de gestión de contenido enfocado a la creación de blogs. Desarrollado en PHP y MySQL, bajo licencia GPL, tiene como fundador a Matt Mullenweg. WordPress fue creado a partir del desaparecido b2/cafelog y se ha convertido junto a Movable Type en el CMS más popular de la blogosfera. Las causas de su enorme crecimiento son, entre otras, su licencia, su facilidad de uso y sus características como gestor de contenidos. WordPress es un sistema de gestión de contenido enfocado a la creación de un sitio web periódicamente actualizado. Desarrollado en PHP y MySQL, bajo licencia GPL, tiene como fundador a Matt Mullenweg.

Xoops

XOOPS son las siglas en inglés de extensible Object Oriented Portal System (Sistema de portal extensible orientado a objetos). Comenzó como un sistema de portal; sin embargo, XOOPS se ha convertido en un Sistema de gestión de contenido que permite a los administradores crear fácilmente sitios web dinámicos. Se está convirtiendo en un framework como herramienta para pequeñas, medianas y grandes websites. XOOPS está escrito en PHP y utiliza como Base de datos a MySQL. XOOPS se encuentra bajo los términos de la licencia pública general (GPL) y cualquier persona es libre para utilizarlo, modificarlo y redistribuirlo bajo los mismos términos de la GPL.

Typo3

Typo3 es un software libre de portal y gestión de contenidos bajo la licencia libre GPL. Es el fruto de varios años de trabajo de Kasper Skårhøj. El producto vio la luz a finales de 2000, con la participación de una comunidad muy activa que se desarrolló, en primer lugar en los países nórdicos y germánicos y después en Francia, Estados Unidos y Canadá. (Buetas, 2009)

Tabla comparativa de los CMS

	Drupal	Joomla	EZ-Publish
Versión	6.10	1.5.10	4.0.2
	Requerimientos		
Servidor Web	Apache	Apache	Apache
Base de datos	MySQL y Pgsql	MySQL	MySQL
Lenguaje	PHP	PHP	PHP
SO	Unix/Windows/Mac	Unix/Windows	Unix

Tabla 1: Comparación entre CMS

El portal para la disposición de la documentación técnica del Marco de trabajo será implementado sobre el Sistema de Gestión de Contenido (CMS) Drupal por poseer la mayor comunidad de desarrollo, ser multiplataforma y tener compatibilidad con la plataforma de desarrollo del propio marco de trabajo LAPP (Linux, apache, php y PostgreSQL). Además cumple con las características que se necesitan para gestionar la información.

1.6 Conclusiones

La investigación realizada en el diseño teórico de la investigación da cumplimiento para el posterior desarrollo de la documentación técnica a realizar y para la creación de un portal, con una eficaz herramienta colaborativa, para la disposición de dicha documentación, dando cumplimiento a los objetivos trazados para la investigación.

La escasez y falta de organización de la documentación del marco de trabajo no permite un fácil entendimiento y mantenimiento del mismo. Por ello se realizará la documentación formal del marco de trabajo con el objetivo de lograr una correcta

utilización de los aspectos tecnológicos y arquitectónicos de la arquitectura planteada para el desarrollo del Sistema de gestión Cedrux.

2

CAPÍTULO II Documentación técnica del marco de trabajo

2. Introducción

El objetivo específico del proyecto ERP-Cuba es desarrollar un marco tecnológico endógeno competitivo a nivel mundial con garantía de independencia tecnológica y de alto grado de productividad en su explotación. En el desarrollo de un marco de trabajo o tecnológico de tal envergadura, la documentación del mismo, juega un papel específico, puesto que es base de aprendizaje y utilización para la reutilización de sus aspectos tecnológicos para el futuro desarrollo de sistemas de alta calidad, productividad y competitividad mundial de los productos informáticos de la industria del software cubana.

2.1 Disposición de la documentación del marco de trabajo para las líneas de desarrollo del sistema de gestión Cedrux.

Toda la información referente al sistema de gestión Cedrux se encuentra en un repositorio, el cual puede ser accedido desde las líneas de desarrollo utilizando la herramienta TorstoiseSVN[1]. La línea de Arquitectura es la encargada de definir, implementar y documentar las tecnologías, así como mantener la información generada actualizada. Entre las distintas áreas de la línea de Arquitectura, el área de tecnología es la encargada de poner a disposición toda la documentación referente al marco de trabajo. Para lograr una mayor organización la ubicación y distribución de la información queda dispuesta de la siguiente manera:

[1] Cliente subversión para el control de versiones

Figura 1: Estructura y ubicación de la documentación del Marco de trabajo

Ubicado en ERP/Ingeniería/Arquitectura/Desarrollo/Arquitectura_tecnológica

2.2 Manual de instalación

La Arquitectura de Software es una práctica joven de apenas unos pocos años de trabajo constante, sin embargo ya se ha convertido en un factor de vital importancia para lograr que los Sistemas de Software tengan un alto nivel de calidad. Poseer una buena Arquitectura de Software es de suma importancia, ya que ésta es el corazón de todo Sistema de Software y determina cuáles serán los niveles de calidad asociados al sistema.

2.2.1 Propósito

Este documento tiene como objetivo fundamental lograr que el usuario adquiera conocimientos en cuanto a cómo instalar el marco de trabajo.

2.2.2 Aplicabilidad

Este manual es una guía de todos los pasos y procedimientos que se deben seguir para alcanzar resultados óptimos en cuanto al desarrollo de aplicaciones con el marco de trabajo.

2.2.3 Instalación

El marco de trabajo cuenta con un instalador para ser ejecutado sobre el sistema operativo Linux. Para la ejecución de dicho instalador se deben cumplir los siguientes requerimientos:

PHP 5.2 o superior (Recomendado PHP 5.2.4) con los siguientes módulos o extensiones:

- ✓ pdo. Para el acceso a bases de datos.
- ✓ pgsql Para el acceso nativo a bases de datos postgres.
- ✓ pdo_pgsql Para el acceso bases de datos postgres con pdo.
- ✓ simplexml Para el trabajo con ficheros de configuración xml.
- ✓ soap Para el trabajo con servicios web.
- ✓ json Para el envío de información de forma dinámica al cliente.
- ✓ gd Para el trabajo con imágenes desde el servidor.
- ✓ bcmath Para el cálculo con números de gran tamaño.
- ✓ xsl Para la transformación de ficheros a través de ficheros xslt.

Para la ejecución de dicho instalador se deben seguir los siguientes pasos:

1. Copiar el fichero compactado INSTALL_MT_CEDRUX_22-03-2009.tar.gz para el servidor web.
2. Abrir una terminal como **root** en el mismo.
3. Descompactar el fichero INSTALL_MT_CEDRUX_22-03-2009.tar.gz.
   a. ***Ej. root@debian-server:~# tar -xvfz INSTALL_MT_CEDRUX_22-03-2009.tar.gz -C /root/***
4. Posicionarse en la raíz de la carpeta de instalación (INSTALL_MT_CEDRUX_22-03-2009).
   a. ***Ej: root@debian-server:~$ cd /root/INSTALL_MT_CEDRUX_22-03-2009***
5. Editar el fichero Configuración/codificar/users.xml para cambiar los password de los usuarios de base de datos usados por el sistema.
   a. ***Ej.: root@debian-server:~/INSTALL_MT_CEDRUX_22-03-2009#mcedit Configuración/codificar/users.xml***
   <user>password</user>
   <postgres>postgres*2008</postgres>
6. Ejecutar el instalador.
7. ***Ej.: root@debian-server:~/INSTALL_MT_CEDRUX_22-03-2009# ./install.sh***

8. Seguir las instrucciones que van apareciendo.
9. Correr la aplicación en un navegador web Firefox (Firefox 2.0.20 ó superior).
10. ***Ej.: root@debian-server:~/INSTALL_MT_CEDRUX_22-03-2009#***
    a. Firefox http://10.0.0.1/Cedrux &

Para los usuarios que desarrollan sobre el sistema operativo Windows solo debe des compactar en la carpeta de publicación web el compactado **Cedrux** y correr los scripts dispuestos en la carpeta **Scripts.**

2.2.4 Utilización

La descripción para la utilización del marco de trabajo está reflejada en la ayuda del caso de estudio: “usuario”.

2.2.5 Herramientas de desarrollo

Para los usuarios que trabajan con sistema operativo **Windows** se utilizarán las herramientas siguientes:

- ✓ Servidor Web Apache 2.0.
- ✓ PHP 5.2.4 con los siguientes módulos o extensiones:
  - pdo.
  - pdo_pgsql.
  - pgsql.
  - soap.
- ✓ Gestor de Base de Datos PostgreSQL 8.3.
- ✓ Explorador Web de la familia Mozilla Firefox u otro que implemente el DOM 2.0 y que soporte JavaScript.
- ✓ IDEs de desarrollo para PHP y/o JavaScript.
- ✓ ZendStudio Neon para PHP y para ZendFramework.
- ✓ Ecilpse 3.3, agregar pluing para PHP y pluing para JavaScript (Spket o Aptana).
- ✓ Aptana para JavaScript.
- ✓ Spket para JavaScript.

Para los usuarios que trabajan con sistema operativo **Linux** pueden utilizar cualquier herramienta compatible con las herramientas dispuestas para **Windows.**

2.3 Descripción de los documentos del marco de trabajo

Los documentos generados en el área de tecnología deben tener una tabla que represente el control de versiones y una tabla que refleje la aprobación del documento. Ejemplo:

Control de versiones

No.	Descripción	Observaciones	Responsable	Fecha
V 1.0	Descripción y especificación técnica del componente X.	Elaboración de la plantilla del documento	Yoandry Morejón Borbón	12/03/2009

Se establece la versión del documento, una idea del objetivo del documento, el responsable de la confección del mismo y la fecha de creación de dicho documento.

Tabla de aprobación

No.	Fecha	Aprobado por:	Cargo	Firma
1	12/03/2009	Yoandry Morejón Borbón	Arquitecto	

Se establece los responsables de aprobar el documento especificando el cargo que presenta. En caso de impresión se debe se debe validar el documento añadiéndole la firma de los encargados de ejecutar la aprobación.

2.4 Aspectos a tratar en la documentación de los componentes del marco de trabajo.

La esencia del marco de trabajo se ve reflejada en la descripción de los componentes realizados, específicamente los componentes que cubren los aspectos arquitectónicamente significativos y los que cubren las principales funcionalidades del proceso de negocio.

En la descripción de los componentes del marco de trabajo deben tener en cuenta los siguientes aspectos:

Introducción: Describir la situación problemática, el problema a resolver, los objetivos, y la estructura del documento.

Desarrollo: Especificación y descripción técnica del componente, requisitos, diseño y casos de prueba del componente

- ✓ Especificación de requisitos del componente: Descripción de los requisitos funcionales y no funcionales del componente, especificación de los prototipos de interfaz de usuario.
- ✓ Diseño del componente: Diseño del componente; especificación y descripción de los diagramas de clases, diagramas de secuencia, modelo de datos y diagramas de componentes.
  - Diagrama de clases: Especificación y descripción de los diagramas de clases.
- ✓ Configuración y uso del componente: Especificación y descripción de los elementos de configuración del componente, ficheros de configuración .xml, .php, .conf, .ini; descripción y especificación de los casos de estudio.
  - Ficheros de configuración: Especificación y descripción de los ficheros de configuración .xml, .php, .conf, .ini.
  - Casos de estudio: Especificación y descripción de los casos de estudio.

Conclusiones: Se hace un resumen de los principales aspectos mencionados en el documento, resaltando los más importantes a tener en cuenta.

2.5 Descripción de los componentes del marco de trabajo.

En este epígrafe se hará una descripción de los componentes del marco de trabajo, extendidos del ZendFramework (ZendExt), los cuales le dan solución a funcionalidades del proceso de negocio y que respaldan esencialmente los aspectos arquitectónicamente significativos:

- ✓ Componente Caché **(ZendExt_Cache)**
- ✓ Componente Estructuras de datos **(ZendExt_ADT)**
- ✓ Componente Excepciones **(ZendExt_Exception)**
- ✓ Componente ExpImp **(ZendExt_ExpImp).**
- ✓ Componente Estructuras dinámicas **(ZendExt_Nom)**

- ✓ Componente Conceptos globales **(ZendExt_GlobalConcept)**
- ✓ Componente Inversión de control **(ZendExt_IoC)**
- ✓ Componente Núcleo y configuración **(ZendExt_App)**
- ✓ Componente Trazas **(ZendExt_Trace)**
- ✓ Componente Validación **(ZendExt_Validation)**

2.5.1 Componente Caché **(ZendExt_Cache)**

Problema a resolver

La ausencia de un componente para la gestión de la caché en el marco de trabajo de Cedrux y de la UCID disminuye el rendimiento de las aplicaciones que se están desarrollando.

Objetivos

Desarrollar, adaptar o integrar un componente al marco de trabajo que permita a las aplicaciones que se están desarrollando la gestión de la cache, permitiendo almacenar y recuperar la información almacenada en la misma de forma rápida y sencilla. El componente debe permitir guardar información en cualquiera de los siguientes formatos, texto, arreglos o tablas hash, objetos o instancias de clases, estructuras complejas como listas, pilas, colas, arboles y otras estructuras de mayor complejidad.

Especificación de requisitos

Requisitos no funcionales

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.

Requisitos funcionales

R1 – Gestionar la información de la caché.

R1.1 – Registrar información en caché.

R1.2 – Modificar información de la caché.

R1.3 – Eliminar información de la caché.

R2 – Recuperar información de la caché.

R3 - Limpiar la caché o parte de ella.

R4 – Verificar si alguna información de la caché está disponible.

R4 - Configurar el almacenamiento de la caché.

R4.1 – Configurar el lugar donde se desea almacenar la caché.

R4.2 – Configurar el tiempo de expiración de la caché.

R4.3 – Configurar el mecanismo de almacenamiento de la caché.

R4.4 – Configurar el tipo de información que se desea almacenar

Diseño

Diseño de clases

Ver Anexo 1: Diagrama de clases componente ZendExt_Cache

Configuración y uso

1 Obtención de una instancia de ZendExt_Cache

```
static public function getInstance() {
    static $instance;
    if (! isset ( $instance ))
        $instance = new self ( );
    return $instance;
}
```

2 Inicialización de el gestor de caché de la aplicación

```
private function init() {
    $configCache = Zend_Registry::get ( 'config' )->cache;
    $frontendOptions = array ('lifetime' => $configCache->lifetime,
     'automatic_serialization' => $configCache->automatic_serialization );
    $backendOptions = array ('cache_dir' => $configCache->cache_dir );
    $cache_dir = Zend_Cache_Backend::getTmpDir ();
    if (! is_dir ( $cache_dir )) {
        $cache_dir = $configCache->cache_dir;
        if (! is_dir ( $configCache->cache_dir )) {
            mkdir ( $configCache->cache_dir, $configCache->chmod );
        }
    }
    $backendOptions = array ('cache_dir' => $cache_dir );
    $this->cache = Zend_Cache::factory ( $configCache->frontend,
     $configCache->backend, $frontendOptions, $backendOptions );
    if (extension_loaded ( 'apc' ))
        $this->cache->setBackend ( new Zend_Cache_Backend_Apc ( $backendOptions ) );
}
```

3 Almacenar datos en caché

```
public function save($pData, $pIdCache) {
    $this->cache->save ( $pData, $pIdCache );
}
```

4 Obtener los datos almacenados en caché

```
public function load($pIdCache) {
    return $this->cache->load ( $pIdCache );
}
```

$pIdCache: identificador de los datos de la caché

5 Limpiar Caché

```
public function clean() {
    return $this->cache->clean ();
}
```

6 Eliminar Caché

```
public function remove($pIdCache) {
    return $this->cache->remove ( $pIdCache );
}
```

$pIdCache: identificador de los datos de la caché

Ficheros de configuración

Los principales aspectos de la cache vienen referenciados en el fichero principal de configuración de la aplicación (config.php).

```
$config['cache']['frontend']                = 'Core'; //Cachear instancias de clases
$config['cache']['backend']                 = 'File'; //Cachear en ficheros
$config['cache']['lifetime']                = 7200; //Tiempo de Vida
$config['cache']['automatic_serialization'] = true; //Serializar
$config['cache']['cache_dir']               = $dir_www . '/tmp/'; //Directorio de cache
$config['cache']['chmod']                   = 0644; //Directorio de cache
```

Caso de estudio

Para la utilización del componente solo se debe instanciar ZendExt_Cache

Ejemplo: Limpiar caché

```
$a = new ZendExt_Cache ();
$a->clean();
```

2.5.2 Componente Estructuras de datos **(ZendExt_ADT)**

Problema a resolver

En PHP el uso de las estructuras de datos no está homogenizado lo que implica emplear mayor cantidad de tiempo y esfuerzo a la hora de organizar y manipular un conjunto de datos.

Objetivos

Desarrollar un componente capaz de homogenizar el uso de las estructuras de datos en PHP y de esta forma minimizar los tiempos de acceso, así como lograr formas más efectivas de inserción y eliminación de datos en estructuras de almacenamiento.

Especificación de requisitos

Requisitos no funcionales

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.

Requisitos funcionales

R1- Grafo

R1.1 Obtener la posición del nodo
R1.2 Determinar si el grafo es vacío
R1.3 Contar la cantidad de nodos
R1.4 Contar la cantidad de aristas
R1.5 Verificar si es un nodo
R1.6 Verificar si es una arista
R1.7 Adicionar una arista
R1.8 Adicionar un nodo
R1.9 Eliminar un nodo
R1.10 Eliminar una arista
R1.11 Buscar nodos adyacentes
R1.12 Buscar en anchura
R1.13 Buscar en profundidad

R2- Grafo Nodo

R2.1 Obtener el valor del nodo
R2.2 Poner el valor del nodo
R2.3 Obtener el peso del nodo
R2.4 Poner el peso del nodo

R3-Lista

R3.1 Moverse al siguiente nodo
R3.2 Dado del nodo devolver la posición en la que se encuentra
R3.3 Adicionar un nodo al final de la lista
R3.4 Eliminar un nodo
R3.5 Insertar un nodo dada una posición en la lista
R3.6 Dada una posición obtener el valor del nodo
R3.7 Contar la cantidad de nodos de la lista

R4-Nodo

R4.1 Obtener el valor del nodo

R4.2 Obtener el valor del próximo nodo
R4.3 Poner el valor del nodo
R4.4 Poner el valor del próximo nodo

R5-Cola

R5.1 Agregar un nodo al final de la cola
R5.2 Extraer un nodo de la cola
R5.3 Devolver el valor del primer elemento de la cola
R5.4 Devolver el valor del último elemento de la cola
R5.5 Contar la cantidad de elementos de la cola

R6-Pila

R6.1 Agregar un elemento a la pila
R6.2 Extraer un elemento de la pila
R6.3 Devolver el valor del elemento que está en el tope de la pila
R6.4 Determinar si la pila es vacía
R6.5 Contar la cantidad de elementos de la pila

R7-Árbol

R7.1 Adicionar un nodo al árbol

R8-Árbol Nodo

R8.1 Obtener el valor del nodo
R8.2 Obtener el valor del hijo

Diseño

Diseño de clases

Ver Anexo 2: Diagrama de clases componente ZendExt_ADT*Anexo 2*.

Anexo 2: Diagrama de clases componente ZendExt_ADT

Configuración y uso

Este componente creará estructuras de datos para PHP hasta ahora no existentes y así homogenizar el uso de estas en el desarrollo del proyecto.

1. Clase Graph

1.1 Función para obtener la posición del nodo

```
private function GetVertexPos($pVertex) {
    $pos = $this->vertex_list->Find ( $pVertex );
    return $pos;
}
```

1.2 Determinar si el grafo está vacío

```
public function IsEmpty() {
    return $this->vertex_list->Count () == 0;
}
```

1.3 Función para contar la cantidad de nodos

```
public function CountVertex() {
    return $this->vertex_list->Count ();
}
```

1.4 Función para contar la cantidad de aristas

```
public function CountArcs() {
    $count = 0;

    for($i = 0; $i < $this->vertex_list->Count (); $i ++)
        for($j = 0; $j < $this->vertex_list->Count (); $j ++)
            if ($this->matrix [$i] [$j] != INFINITE)
                $count ++;

    return $count;
}
```

1.5 Verifica si es un nodo

```
public function IsVertex($pVertex) {
    return $this->vertex_list->Find ( $pVertex ) != - 1;
}
```

1.6 Verifica si es una arista

```
public function IsArc($pSource, $pTarget) {
    $posSource = $this->GetVertexPos ( $pSource );
    $posTarget = $this->GetVertexPos ( $pTarget );

    if ($posSource == - 1)
        throw new ZendExt_Exception ( 'O02' );

    if ($posTarget == - 1)
        throw new ZendExt_Exception ( 'O03' );

    return ($this->matrix [$posSource] [$posTarget] != INFINITE);
}
```

1.7 Adiciona una arista

```
public function AddArc($pSource, $pTarget, $pCost) {
    if (! $this->IsArc ( $pSource, $pTarget )) {
        if ($this->IsVertex ( $pSource )) {
            if ($this->IsVertex ( $pTarget )) {
                $posSource = $this->GetVertexPos ( $pSource );
                $posTarget = $this->GetVertexPos ( $pTarget );

                $this->matrix [$posSource] [$posTarget] = $pCost;
            } else
                throw new ZendExt_Exception ( 'A003' );
        } else
            throw new ZendExt_Exception ( 'A002' );
    } else
        throw new ZendExt_Exception ( 'A001' );
}
```

1.8 Adiciona un nodo

```
public function AddVertex($pVertex) {
    $this->vertex_list->Add ( $pVertex );
}
```

1.9 Elimina un nodo

```
public function RemoveVertex($pVertex) {
    if ($this->IsVertex ( $pVertex )) {
        $posVertex = $this->GetVertexPos ( $pVertex );

        for($i = 0; $i < $this->vertex_list->Count (); $i ++) {
            $this->matrix [$i] [$posVertex] = INFINITE;
            $this->matrix [$posVertex] [$i] = INFINITE;
        }

        $this->vertex_list->Remove ( $posVertex );
    } else
        throw new ZendExt_Exception ( 'A004' );
}
```

1.10 Elimina una arista.

```
public function RemoveArc($pSource, $pTarget) {
    if ($this->IsArc ( $pSource, $pTarget )) {
        if ($this->IsVertex ( $pSource )) {
            if ($this->IsVertex ( $pTarget )) {
                $posSource = $this->GetVertexPos ( $pSource );
                $posTarget = $this->GetVertexPos ( $pTarget );

                $this->matrix [$posSource] [$posTarget] = INFINITE;
            } else
                throw new ZendExt_Exception ( 'A007' );
        } else
            throw new ZendExt_Exception ( 'A006' );
    } else
        throw new ZendExt_Exception ( 'A005' );
}
```

1.11 Búsqueda de nodos adyacentes

```
public function Near($pVertex) {
    $posVertex = $this->GetVertexPos ( $pVertex );
    $list = $this->matrix [$posVertex];

    $result = new ZendExt_ADT_List ( );
    foreach ( $list as $tmp )
        $result->Add ( $this->vertex_list [$tmp] );
    return $result;
}
```

1.12 Búsqueda en anchura

```
public function BFS (& $pResultList) {}
```

1.13 Búsqueda en profundidad

```
public function DFS(& $pResultList, $pVertex) {
    if ($pResultList->Count () != $this->CountVertex ()) {
        $pos = $this->GetVertexPos ( $pVertex );
        $this->visited [$pos] = true;

        for($i = 0; $i < $this->CountVertex (); $i ++)
            if (! $this->visited [$i]) {
                $vertex = $this->vertex_list->Get ( $i );
                $pResultList->Add ( $vertex );
                $this->DFS ( $pResultList, $vertex );
            }
    }
}
```

2. Clase GraphNode

```
public function getData () {return $this->data ;}
```

2.1 Obtener el valor del nodo

```
public function getNext () {return $this->next ;}
```

2.2 Obtener el valor del próximo nodo.

```
public function setNext ($pNext) {$this->next = $pNext ;}
```

2.3 Poner el valor del próximo nodo.

```
public function setData ($pData) {$this->data = $pData ;}
```

2.4 Poner el valor del nodo.

```
public function Zend_ADT_Node ($pData = null, $pNext = null) {

$this->data = $pData; $this->next = $pNext ;}
```

3. Clase List

3.1 Moverse al siguiente nodo, es decir la próxima posición

```
private function MoveTo($pPos) {
    if ($pPos > $this->count)
        throw new ZendExt_ADT_Exception_ListOverflow ( );

    $resp = $this->head;

    for($i = 0; $i < $pPos; $i ++)
        $resp = $resp->getNext ();

    return $resp;
}
```

3.2 Dado el nodo devuelve la posición

```
public function Find($pNode) {
    for($i = 0; $i < $this->count; $i ++)
        if ($this->Get ( $i )->getData () == $pNode)
            return $i;

    return - 1;
}
```

3.3 Constructor de la clase

```
public function ZendExt_ADT_List() {
    $this->head = new ZendExt_ADT_Node ( );
    $this->count = 0;
}
```

3.4 Adicionar un nodo al final de la lista.

```
public function Add($pValue) {
    if ($this->count == 0) {
        $this->head = new ZendExt_ADT_Node ( $pValue );
    } else {
        $pos = $this->MoveTo ( $this->count - 1 );
        $pos->setNext ( new ZendExt_ADT_Node ( $pValue ) );
    }

    $this->count ++;
}
```

3.5 Eliminar un nodo

```
public function Remove($pPos) {
    if ($pPos == 0) {
        $this->head = $this->head->getNext ();
    } else if ($pPos == $this->count - 1) {
        $prev = $prev = $this->MoveTo ( $pPos - 1 );
        $prev->setNext ( new ZendExt_ADT_Node ( ) );
    } else {
        $prev = $this->MoveTo ( $pPos - 2 );
        $next = $this->MoveTo ( $pPos );
        $prev->setNext ( $next );
    }

    $this->count --;
}
```

3.6 Insertar un nodo dada una posición en la lista.

```
public function Insert($pPos, $pValue) {
    if ($pPos == 0) {
        $next = $this->head;
        $nuevo = new ZendExt_ADT_Node ( $pValue, $next );
        $this->head = $nuevo;
    } else if ($pPos == $this->count - 1) {
        $this->Add ( $pValue );
        return;
    } else {
        $prev = $this->MoveTo ( $pPos - 2 );
        $next = $this->MoveTo ( $pPos - 1 );
        $prev->setNext ( $nuevo );
    }

    $this->count ++;
}
```

3.7 Dada la posición obtener el valor del nodo.

```
public function Get($pPos) {
    return $this->MoveTo ( $pPos - 1 )->getData ();
}
```

3.8 Cuenta cuántos nodos tiene la lista.

```
public function Count() {
    return $this->count;
}
```

4. Clase Node

4.1 Obtener el valor del nodo.

```
public function getData ()
{
    return $this->data;
}
```

4.2 Obtener el valor del próximo nodo.

```
public function getNext ()
{
    return $this->next;
}
```

4.3 Poner el valor del próximo nodo.

```
public function setNext ($pNext)
{
    $this->next = $pNext;
}
```

4.4 Poner el valor del nodo.

```
public function setData ($pData)
{
    $this->data = $pData;
}
```

4.5 Constructor de la clase.

```
public function Zend_ADT_Node ($pData = null, $pNext = null)
{
    $this->data = $pData;
    $this->next = $pNext;
}
```

5. Clase Queque

5.1 Agregar un nodo al final de la cola.

```
public function Put($date) {
    if ($this->count == 0) {
        $this->head = new Zend_ADT_Node ( $date );
        $this->end = $this->head;
    } else {
        $temp = $this->end;
        $this->end = new Zend_ADT_Node ( $date );
        $this->temp->next = $this->end;

    }
    $this->count ++;
}
```

5.2 Extraer un nodo de la cola.

```
public function Extract() {
    if ($this->count > 0) {
        if ($this->count == 1) {
            $temp = $this->head;
            $this->head = null;
            $this->end = null;
            return $this->temp;
        } else {
            $temp = $this->head;
            $this->head = $this->head->next;
            return $this->temp;
        }
        $this->count --;
    }
}
```

5.3 Devuelve el valor del primer elemento de la cola.

```
public function Head() {
    if ($this->head != null)
        return $this->head;
}
```

5.4 Devuelve el valor del último elemento de la cola.

```
public function End() {
    if ($this->end != null)
        return $this->end;
}
```

5.5 Devuelve la cantidad de elementos de la cola.

```
public function Count() {
    return $this->count;
}
```

6. Clase Stack

6.1 Agregar un elemento a la pila.

```
public function Push($date) {
    if ($this->count == 0)
        $this->top = new Zend_ADT_Node ( $date );
    else {
        $temp = $this->top;
        $this->top = new Zend_ADT_Node ( $date );
        $this->top->next = $this->temp;
    }

    $this->count ++;
}
```

6.2 Extraer un elemento de la pila.

```
public function Pop() {
    if ($this->count > 0) {
        if ($this->count == 1) {
            $temp = $this->top;
            $this->top = null;
            return $this->temp;
        } else {
            $temp = $this->top->next;
            $end = $this->top;
            $this->top = $this->temp;
            return $end;

        }

        $this->count --;

    }
}
```

6.3 Devuelve el valor del elemento que está en el tope de la pila.

```
public function Top() {
    if ($this->count > 0)
        return $this->top;
}
```

6.4 Permite conocer si la pila se encuentra vacía

```
public function IsEmpty() {
    return ($this->count == 0);
}
```

6.5 Devuelve la cantidad de elementos de la pila.

```
public function Count() {
    return $this->count;
}
```

7. Clase Tree

7.1 Constructor de la clase

```
public function ZendExt_ADT_Tree($pRootValue)
{
    $this->root = new ZendExt_ADT_TreeNode($pRootValue);
}
```

7.2 Adicionar un nodo al árbol.

```
public function addNodep ($pRootValue)
{
    $newNode = (! $pRootValue instanceof ZendExt_ADT_Tree) ? new ZendExt_ADT_TreeNode($pRootValue)
                                                           : $pRootValue;
    $this->root->getChildren ()->Add ($newNode);
}
```

8. Clase TreeNode

8.1 Constructor de la clase.

```
public function ZendExt_ADT_TreeNode($pValue, $pChildren = null) {
    $this->value = $pValue;
    $this->children = ($this->children == null) ? new ZendExt_ADT_List ( ) : $pChildren;
}
```

8.2 Obtener el valor del nodo.

```
public function getValue() {
    return $this->value;
}
```

8.3 Obtener el hijo.

```
public function getChildren() {
    return $this->children;
}
```

Caso de estudio

Para la utilización de las estructuras de datos solo se hace una instancia de cualquier estructura

Ejemplo: agregar nodo a un árbol

```
$adt = new ZendExt_ADT_Tree();
$adt->addNodep('nodo');
```

2.5.3 Componente Excepciones **(ZendExt_Exception)**

Problema a resolver

En el sistema de gestión Cedrux no basta con el mero hecho del lanzamiento y captura de excepciones sino también se imponen algunos requerimientos descritos a continuación:

- ✓ La internacionalización de las excepciones.
- ✓ Manejo declarativo del comportamiento de las excepciones.
- ✓ Mecanismo extensible de creación de nuevos tipos de excepciones.
- ✓ Codificación de las excepciones.
- ✓ Tratamiento igualitario de las excepciones propias o de los marcos de desarrollo subyacentes (Doctrine, Zend y EzComponent) que ya poseen su propia forma de gestionarlas.

Objetivos

Desarrollar, adaptar o integrar un componente al marco de trabajo que permita a las aplicaciones que se están desarrollando la gestión de las excepciones, permitiendo el tratamiento y registro de las excepciones del sistema.

Especificación de requisitos

Requisitos no funcionales

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.

Requisitos funcionales

R1 – Configuración y registro de las distintas excepciones que pueden ocurrir en cada acción de las clases control y modelos que se implementan.

R2 – Configuración y registro de los datos de las excepciones que se adicionan.

R3 – Realizar el tratamiento de excepciones.

R3.1 – Obtener el identificador o código de la excepción.

R3.2 – Obtener la excepción interna que ocurrió.

R3.3 – Obtener la descripción de la excepción.

R3.4 – Obtener el tipo de la excepción.

R3.5 – Obtener la clase que lanzó la excepción.

R3.6 – Obtener la acción de la clase que disparó la excepción.

R3.7 – Obtener el subsistema en el que ocurrió la excepción.

R3.8 – Mostrar o no el mensaje de la excepción según el tipo con que se registró.

R3.9 – Salvar y mostrar los datos de la excepción.

Diseño

Diseño de clases

Ver Anexo 3: Diagrama de clases componente ZendExt_Exception

Configuración y uso

La utilización del componente es básicamente la configuración correcta de los ficheros **managerexception.xml** y **exception.xml**. Cuando las excepciones sean registradas desde el código php de las clases no es necesario realizar el lanzamiento y captura de la excepción ya que el componente es totalmente capaz de capturar la excepción y darle el tratamiento correspondiente.

Ficheros de configuración

Primeramente es necesario saber como registrar el tratamiento con el que se le dará curso a una excepción, esto se realiza mediante la configuración de **exception.xml**

que se encuentra en el directorio de ficheros comunes a las aplicaciones o subsistemas del proyecto. Las mismas se registran como se muestra a continuación:

```
<NOM011><!-- Identificador o codigo de la excepcion -->

      <tipo>LP</tipo><!-- Tipo de excepcion -->

      <es><!-- Lenguaje en que se dispara la excepcion -->

            <mensaje>Error en una transacción del
            nomenclador. En el HoleQuery</mensaje><!--
            Mensaje de la excepcion -->

            <descripcion>Ocurrió un error interno del
            sistema. Por favor contacte al
            administrador.</descripcion><!-- Descripcion de
            la excepcion -->

      </es>
```

```
<EST001>
   <tipo>LP</tipo>
   <es>
       <mensaje>Debe configurar la entidad a la que pertenece</mensaje>
       <descripcion>Ocurrió un error interno del sistema. Por favor contacte al administrador.</descripcion>
   </es>
</EST001>
```

EST001: Identificador o código de la excepción

Tipo: Tipo de excepción

Es: Lenguaje en el que se dispara la excepción

Mensaje: Mensaje de la excepción

Descripción: Descripción de la excepción

El código de la excepción lo registra el desarrollador en el formato que se determine en el subsistema.

Los tipos con los que se registra una excepción son:

Presentación ó Presentation (P en el xml): Este tipo de excepciones son presentadas al usuario con un mensaje descrito en un documento XML que garantiza la internacionalización del mensaje así como la modificación oportuna del mismo sin necesidad de interactuar con el código de la aplicación.

Registro ó Log (L en el xml): Estas excepciones no se muestran al usuario, sólo se registran con un formato específico en un archivo definido en el fichero de configuración del marco de trabajo.

Híbridas ó LogPresentation (LP en el xml): Poseen el comportamiento de las de registro y las de presentación pero realizan ambas actividades al unísono.

Ciegas ó Blind (B en el xml): Estas excepciones fueron creadas bajo el principio de encapsular las excepciones producidas por agentes externos al marco de trabajo (otros frameworks, servicios Web o procesos no controlados de la plataforma de desarrollo).

El mensaje puede ser en los idiomas:

Inglés ó English (en el xml): Mostrará y salvará los mensajes de la excepción en inglés.

Español (es en el xml): Mostrará y salvará los mensajes de la excepción en español.

El mensaje como tal es el texto explicativo de la excepción que se salvará del lado del servidor y la descripción será el texto del mismo tipo que le será mostrado al usuario por lo tanto el texto de la misma debe ser lo más sencillo posible.

Además se debe configurar también el fichero **managerexception.xml** para que al ocurrir una excepción determinada el framework sea capaz de gestionarla como se debe, dicho fichero se configura:

```
<exceptions>
    <usuario>
        <GestusuarioController>
            <cargargridusuarioAction>
                <exception type="Doctrine_Excepcion" code="EGU001" />
            </cargargridusuarioAction>
            <cargarformusuarioAction>
                <exception type="Doctrine_Excepcion" code="EGU002" />
            </cargarformusuarioAction>
            <insertarusuarioAction>
                <exception type="Doctrine_Excepcion" code="EGU003" />
            </insertarusuarioAction>
            <modificarusuarioAction>
                <exception type="Doctrine_Excepcion" code="EGU004" />
                <exception type="Doctrine_Connection_Pgsql_Exception" code="EGU005"
                          filter="CONCURRENCE RESTRICTION"/>
            </modificarusuarioAction>
            <eliminarusuarioAction>
                <exception type="Doctrine_Excepcion" code="EGU006" />
            </eliminarusuarioAction>
        </GestusuarioController>
    </usuario>
```

Usuario: Subsistema

GestusuarioController: Controlador donde se ejecuta la acción

CargargridusuarioAction: Método o Acción

Excepción

Type: tipo de excepción

Code: código de la excepción

En el registro de la excepción en específico se configura:

`Type:` tipo de la excepción interna que ocurrió por ejemplo:

`Doctrine_Excepcion:` excepción de Doctrine. Las excepciones de este tipo están relacionadas con errores en la base de datos en las consultas, fallas en la conexión a la misma, etc.

`Doctrine_Connection_Pgsql_Exception:` indica errores directamente en la ejecución de la base de datos a través del PostgreSQL que es el gestor determinado.

`Zend_Exception:` excepción lanzada en las clases control, relacionadas con errores en la utilización del Zend Framework.

`Exception:` se especifica así cuando en un método pueden ocurrir excepciones de las que no se tiene conocimiento, o sea cuando se quiere realizar un tratamiento a cualquier excepción que ocurra.

`Code:` no es más que el identificador del tratamiento que se le va a dar a la excepción, según el tipo, cuando ocurra, este identificador es con el que se registra la excepción en el fichero exception.xml. Es válido aclarar que cuando ocurra una excepción de un tipo en un método especificado el framework lee este código y lo busca en el xml de excepciones y le da curso a la excepción, o sea la salva o no en el servidor, se la muestra o no al cliente o ambas cosas, según lo que se haya especificado en el fichero.

`Filter:` es la declaración de filtrar que cuando ocurra una excepción de un tipo (`Doctrine_Connection_Pgsql_Exception),` es específicamente de otro prototipo determinado (`CONCURRENCE RESTRICTION`). Esto por lo general se realiza para los métodos de modificar información en el tratamiento de la concurrencia.

Finalmente como se puede concluir para capturar una excepción en un método determinado solamente se debe registrar y configurar debidamente en los ficheros exception.xml y managerexception.xml, no se realiza ninguna sentencia de try y catch en las clases php.

Caso de estudio

En el caso del módulo usuario se tiene un caso de uso que se llama “Gestionar Usuario” y el mismo tiene un requisito que implementa una acción que se llama “modificar usuario”, aquí se debe tener en cuenta que pueden ocurrir las siguientes excepciones:

1. De conexión a la base de datos (Doctrine).
2. Del trabajo con Zend Framework.
3. De concurrencia, que no son más que las peticiones de actualización de datos que se le realizan al servidor al mismo tiempo, donde un usuario modifico la información y un segundo al no tener actualizados sus datos esta observando en pantalla información antigua que ya no existe y por lo tanto debe actualizarse y ver la nueva información primero antes de modificar los datos evitando descontrol con la información.
4. Otras Excepciones que puedan ocurrir.

¿Cómo quedaría implementado el requisito?

La acción en la clase control, llamada “GestusuarioController” que pertenece al subsistema “usuario”, al no tener bloque try y catch puede quedar de esta forma:

```
function modificarusuarioAction()
{
    //Se captura el identificador de la persona
    $idpersona = $this->_request->getPost('idpersona');

    //Se obtiene el usuario a modificar
    $datUsuario = DatUsuario::getObjectById($idpersona);
    //Se setean los nuevos datos del usuario
    if($this->_request->getPost('alias'))
        $datUsuario->alias = $this->_request->getPost('alias');
    if ($this->_request->getPost('ididioma'))
        $datUsuario->ididioma = $this->_request->getPost('ididioma');
    if ($this->_request->getPost('idtema'))
        $datUsuario->idtema = $this->_request->getPost('idtema');
    if ($this->_request->getPost('version'))
        $datUsuario->version = $this->_request->getPost('version');
    //Se muestra un mensaje de notificacion
    $this->showMessage('El usuario fue modificado satisfactoriamente.');
}
```

Por consiguiente la configuración del fichero managerexception.xml para este método sería:

```
<modificarusuarioAction>
    <exception type="Doctrine_Excepcion" code="EGU004" />
    <exception type="Doctrine_Connection_Pgsql_Exception" code="EGU005"
            filter="CONCURRENCE RESTRICTION"/>
</modificarusuarioAction>
```

La configuración de exception.xml quedaría entonces:

```
<EGU005>
    <tipo>LP</tipo>
    <es>
        <mensaje>Error de concurrencia al modificar.</mensaje>
        <descripcion>Los datos del usuario fueron modificados por otra persona.
        Por favor recargue los datos e intente realizar la modificación nuevamente.</descripcion>
    </es>
</EGU005>
```

2.5.4 Componente ExpImp **(ZendExt_ExpImp)**

Problema a resolver

La ausencia de un componente para regir centralmente el modo de exportar e importar objetos, conlleva a que constantemente este proceso se realice de manera no uniforme provocando conflictos en la base de datos y el sistema.

Objetivos

Desarrollar un componente para exportar e importar objetos de manera que este proceso sea estandarizado y agilizado.

Especificación de requisitos

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.

Requisitos funcionales

R1 – Exportar objetos.

R2 – Leer el xml de los formatos.

R3 – Importar objetos.

Diseño

Diseño de clases

Ver Anexo 4: Diagrama de clases componente ZendExt_ExpImp

Configuración y uso

El componente CompExpImp permite exportar un objeto a un formato determinado el cual debe estar previamente implementado e importar luego dicho objeto en su formato original.

Clase `ZendExt_CompExImp_CompExImp:` es la clase principal del componente, en ella se encuentran los métodos exportar, importar y parsear.

Método exportar

```php
<?php
class ZendExt_CompExImp_CompExImp {
    public function __construct() {
    }

    public function exportar($obj, $formato) {
        $exp = new ZendExt_CompExImp_Exportacion_Exportador ( );
        $clase = $this->parsear ( $formato );
        return $exp->exportar ( $obj, $clase );
    }
```

Primeramente se instancia el exportador:

```php
public function exportar ($obj, $formato) {

            $exp = new ZendExt_CompExImp_Exportacion_Exportador ( );
```

Luego se lee el xml de los formatos para ver si está implementado el formato en el que se desea exportar el objeto y las clases que lo implementan:

```php
$clase = $this->parsear ($formato);
```

Devuelve el objeto exportado en xml:

```php
return $exp->exportar ($obj, $clase);
```

Método importar

```php
    public function importar($ident, $formato) {
        $exp = new ZendExt_CompExImp_Importacion_Importador ( );
        $clase = $this->parsear ( $formato );
        return $exp->importar ( $ident, $clase );
    }
```

Primeramente se instancia el importador:

```php
public function importar ($ident, $formato) {

            $exp = new ZendExt_CompExImp_Importacion_Importador ( );
```

Luego se lee el xml de los formatos y las clases que lo implementan:

```php
$clase = $this->parsear ($formato);
```

Devuelve el objeto importado en su formato original:

```
function parsear($formato) {
    $xml = simplexml_load_file ( dirname ( __FILE__ ) . DIRECTORY_SEPARATOR . 'Formatos' .
    DIRECTORY_SEPARATOR . 'Formatos.xml' );
    $clase = null;
    foreach ( $xml->File->tipo as $tipo ) {
        $strtipo = ( string ) $tipo ['name'];
        if ($strtipo == $formato)
            $clase = $tipo;
    }
    if ($clase == null) {
        foreach ( $xml->BD->tipo as $tipo ) {
            $tipo = ( string ) $tipo ['name'];
            if ($strtipo == $formato)
                $clase = $tipo;
        }
    }
    return $clase;
return $exp->importar ($ident, $clase);
```

Método parsear:

El método parsear lee el xml de los formatos para ver está implementado el formato en el que se desea exportar el objeto y las clases que lo implementan.

Ficheros de configuración

El componente CompExpImp tiene en: “ZendExt\CompExpImp\Formatos” un fichero de configuración “Formatos.xml” en el cual se especifica el formato en el cual se va a exportar. En este caso el que está implementado es el xml.

```
<?xml version="1.0" encoding="UTF-8"?>
< <tipo name="XML"
<File>
    <tipo name="XML"
          claseex="ZendExt_CompExImp_Exportacion_Fexp_ExportarXML"
          claseimp="ZendExt_CompExImp_Importacion_Fimp_ImportarXML"
          metodoex="exportar"
          metodoimp="importar"/>
</File>
```

Es el formato que se implementa que en este caso es xml.

```
claseex="ZendExt_CompExImp_Exportacion_Fexp_ExportarXML"
```

Clase para exportar

```
claseimp="ZendExt_CompExImp_Importacion_Fimp_ImportarXML"
```

Clase para importar

```
metodoex="exportar"
```

Método de exportación

```
metodoimp="importar"/>
```

Método de importación

Caso de estudio

```
<?php
// El fichero test.xml contiene un documento XML con el elemento raiz
// y al                                menos un elemento /[root]/title.

if (file_exists                                          ('test.xml')) {
    $xml = simplexml_load_file                             ('test.xml');

    var_dump                                                        ($xml);
} else {
    exit                                    ('Error al abrir test.xml.');
}
?>
```

2.5.5 Componente Estructuras dinámicas **(ZendExt_Nom)**

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 solo a nivel de la interfaz creada al efecto sino además por la instanciación de la clase ZendExt_Nomencladores_ADT.

Especificación de requisitos

Requisitos no funcionales

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.

Requisitos funcionales

R1 Conceptos o tablas

R1.1 Adicionar conceptos o tablas

R1.2 Modificar conceptos o tablas

R1.3 Eliminar conceptos o tablas

R1.4 Crear relaciones entre conceptos o tablas

R1.5 Categorizar conceptos o tablas

R1.6 Adicionar conceptos o tablas arbólicos

R2 Campos

R2.1 Adicionar campos a conceptos o tablas

R2.2 Modificar campos a conceptos o tablas

R2.3 Eliminar campos a conceptos o tablas

R3 Elementos

R3.1 Adicionar datos a conceptos o tablas

Diseño

Diseño de clases

Ver Anexo 5: Diagrama de clases componente ZendExt_Nom

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 si 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 solo 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

1.1 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 esta 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 creo o no la tabla

1.2 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

1.3 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.

1.4 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

1.5 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

2.1 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

2.2 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

2.3 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

2.4 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

2.5 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 categoria.
 * @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('tablal'));
```

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 categoria.
 *
 * @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 categoria
 *
 * @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 categorías 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 categoría.
 *
 * @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

4. Funciones para la gestión con los Campos

4.1 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

4.2 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_nomenclador1','campo','idnomenclador1');
```

Parámetro1 (pTable) => Nombre de la tabla

Parámetro2 (pForeignTable) => Nombre de la tabla foránea

Parámetro3 (pLocalField) => Campo local

Parámetro4 (pForeignField) => Nombre del campo foráneo

4.3 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

4.4 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

4.5 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

4.6 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.table1','nom_nomenclador4','campo1','idpadre');
```

Parámetro1 (pextTable) => Tabla externa

Parámetro2 (pNomenclador) => Nombre del nomenclador

Parámetro3 (pextField) => Campo externo

Parámetro4 (pField) => Nombre del campo

4.7 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.

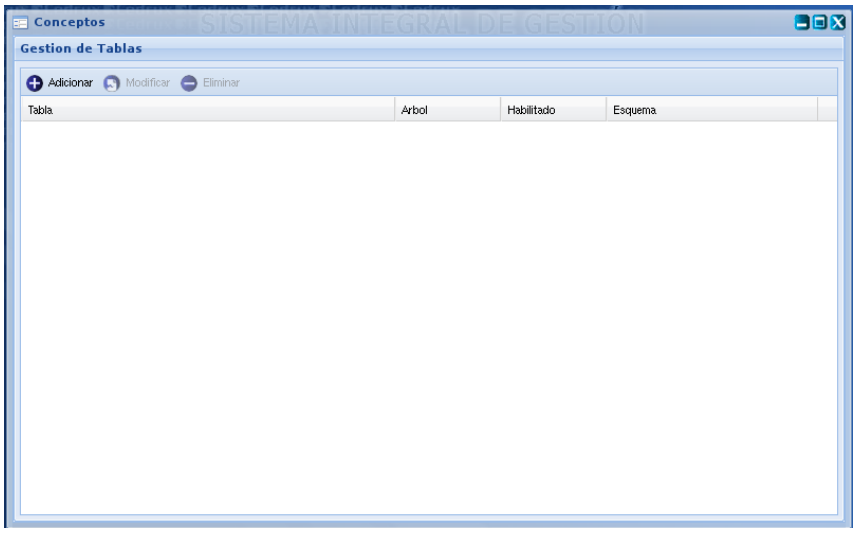


Figura 2: Interfaz para la gestión de tablas

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

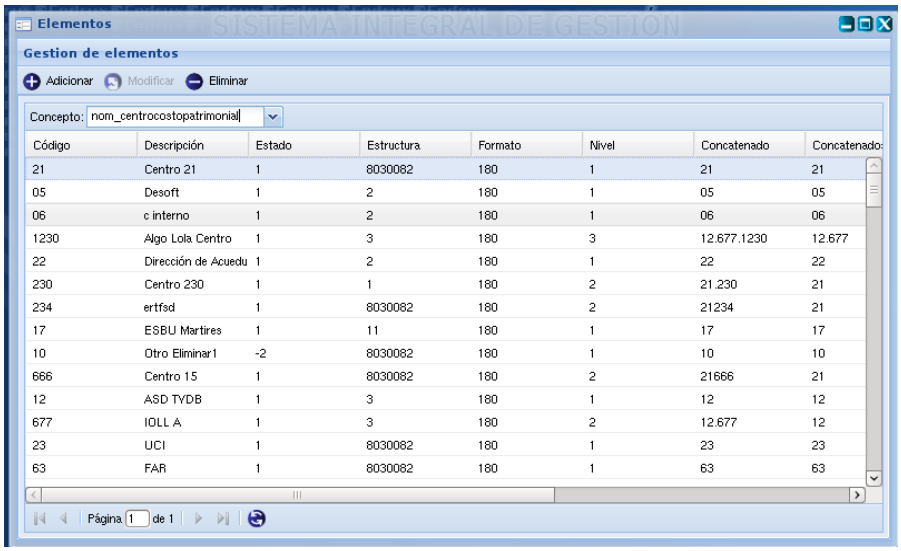


Figura 3: Interfaz para la gestión de elementos

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.

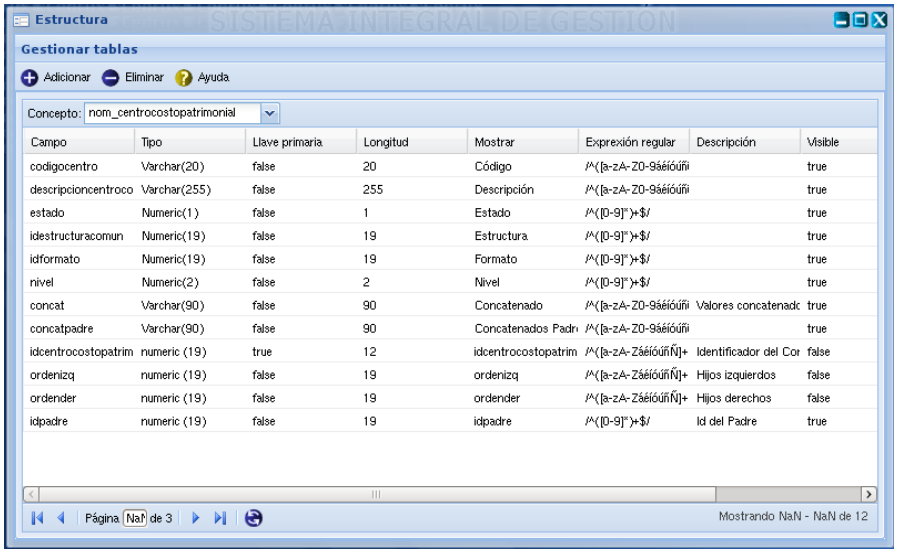


Figura 4: Interfaz para la gestión de campos

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 465366962925321460048054221925745692141442802811280774029406" />
       <datos nombre="mod_capitalhumano" usuario="capitalhumano"
           password="971360642824770761158004961371879619528636173593593928533558281250828264122970658" />
       <datos nombre="mod_cobroypago" usuario="contabilidad"
           password="46555536511679391612 465366962925321460048054221925745692141442802811280774029406" />
       <datos nombre="mod_caja" usuario="contabilidad"
           password="46555536511679391612 465366962925321460048054221925745692141442802811280774029406" />
       <datos nombre="mod_finanzas" usuario="contabilidad"
           password="46555536511679391612 465366962925321460048054221925745692141442802811280774029406" />
   </esquemas>
</config>
```

Caso de estudio

1. A nivel de interfaz

1.1 Crear tabla

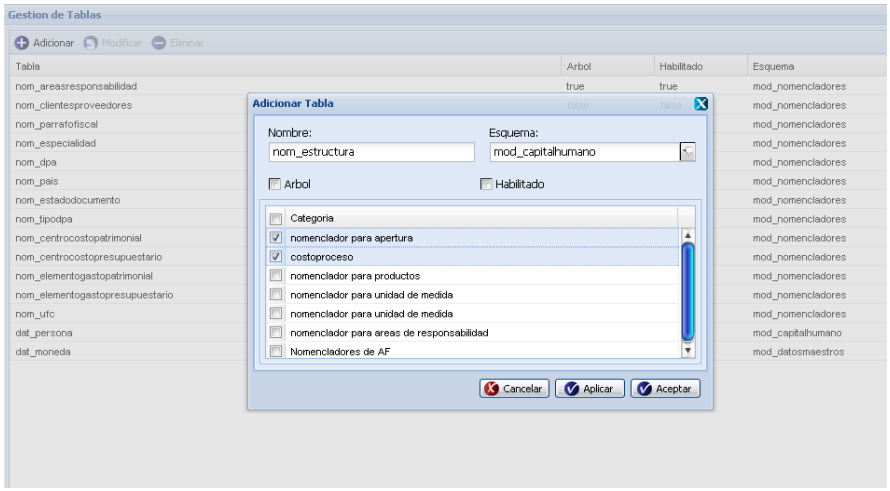


Figura 5: Interfaz para crear tablas

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

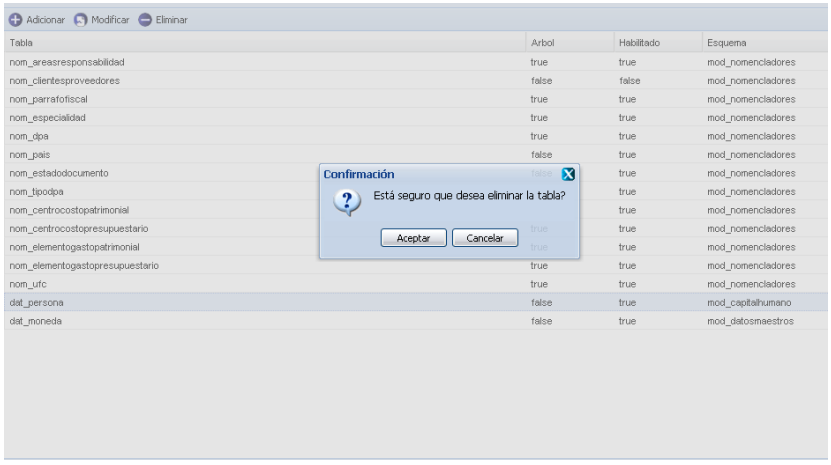


Figura 6: Interfaz para eliminar tablas

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’

2.5.6 Componente Conceptos globales (**ZendExt_GlobalConcept**)

Problema a resolver

La gestión de información común a todos los módulos a través de servicios disminuye el rendimiento de las aplicaciones que se desarrollan en la UCID, específicamente el de la aplicación del proyecto Cedrux o ERP.

Objetivos

Desarrollar, adaptar o integrar un componente al marco de trabajo que permita a las aplicaciones que se están desarrollando la gestión de los conceptos globales, permitiendo almacenar y recuperar la información recopilada en el contenedor de estos conceptos de forma rápida y sencilla. El componente debe permitir guardar información tanto de sesiones (usuario, funcionalidades, etc.) como de la aplicación como tal.

Especificación de requisitos

Requisitos no funcionales

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.

Requisitos funcionales

R1 – Salvar información en el contenedor.

R2 – Obtener concepto global.

R3 – Configuración y registro de los conceptos que se adicionan y los que estarán activos o no.

Diseño

Diseño de clases

Ver Anexo 6: Diagrama de clases componente ZendExt_GlobalConcept

Configuración y uso

El componente que se analiza no necesita ser configurado solamente tendrá un fichero xml donde se activarán o no los conceptos que se quieren salvar, y donde se irán adicionando los nuevo conceptos que vayan surgiendo.

Además se va a configurar como un aspecto que se va a cargar al inicio de la aplicación en el fichero xml de los conceptos.

Por ultimo deberá quedar especificada la dirección del xml de los conceptos en la configuración global del sistema.

Ficheros de configuración

La configuración del componente se realizará en un fichero xml del marco de trabajo que se identifica por concepts.xml, en código xml como indica la ilustración 2.

```
<conceptos>
    <GlobalConcept nombre="Estructura" active='false'/>
    <GlobalConcept nombre="Subsistema" active='false'/>
    <GlobalConcept nombre="Periodo" active='false'/>
    <GlobalConcept nombre="Moneda" active='false'/>
    <GlobalConcept nombre="Ejercicio" active='false'/>
    <GlobalConcept nombre="FechaContable" active='false'/>
    <GlobalConcept nombre="Perfil" active='false'/>
</conceptos>
```

Como se observa se registra bajo la etiqueta “nombre” el identificador del concepto y en “active” se indica si va a estar activo, o sea si se quiere guardar en ese momento (true) o si no se va a salvar (false) en el contenedor de los conceptos. Estos datos se insertan dentro de la etiqueta “GlobalConcept” que a su vez se incluye en la de “conceptos”. En el caso de necesitar adicionar un nuevo concepto que se implementó se pondría dentro de “conceptos” una nueva etiqueta “GlobalConcept” donde se indique el nombre del concepto y si estará activo o no.

Aclarar que esta configuración para el usuario final, que para el framework serán los programadores de la aplicación, no debe ser modificada independientemente de que si la deban conocer, solo se permite la modificación de la misma a los miembros del equipo de arquitectura.

Utilización

La obtención de los conceptos globales se puede realizar desde las clases controladoras que heredan de **ZendExt_Controller_Secure** y además en las clases modelos ubicadas en la carpeta **bussiness** donde se gestiona el negocio del acceso a datos que hereden de ZendExt_Model. Por el momento se implementan los conceptos globales de aplicación Subsistema, Estructura y Configuración, a continuación se explica cada uno de ellos y su forma de acceso:

Subsistema:

```
$this->global->Subsistema
```

Este concepto contiene datos propios que son:

Identificador:

```
$this->global->Subsistema->idsubsistema
```

Denominación:

```
$this->global->Subsistema->denomsubsistema
```

Uri:

```
$this->global->Subsistema->uri
```

Identificador del estado:

```
$this->global->Subsistema->idestado
```

Estado:

```
$this->global->Subsistema->estado
```

Versión:

```
$this->global->Subsistema->version
```

Crc:

```
$this->global->Subsistema->crc
```

Estructura:

```
$this->global->Estructura
```

Datos propios de la estructura que contiene:

Identificador:

```
$this->global->Estructura->idestructura
```

Denominación:

```
$this->global->Estructura->denominacion
```

Abreviatura:

```
$this->global->Estructura->abreviatura
```

Identificador de la especialidad:

```
$this->global->Estructura->idespecialidad
```

Identificador del órgano al que pertenece:

```
$this->global->Estructura->idorgano
```

Configuración:

```
$this->global->Configuracion
```

Este concepto contiene además los conceptos de configuración de los datos del Formato, Perfil, Moneda, Ejercicio y Periodo como se muestra:

Formato:

```
$this->global->Configuracion->Formato
```

Datos propios que porta:

Formato en que deben tener las fechas:

```
$this->global->Configuracion->Formato->fecha
```

Formato en que deben tener las horas:

```
$this->global->Configuracion->Formato->hora
```

Formato en que deben tener las fechas - horas:

```
$this->global->Configuracion->Formato->fechaHora
```

Formato en que debe tener una zona horaria:

```
$this->global->Configuracion->Formato->zHoraria
```

Formato en que debe tener el lenguaje del servidor:

```
$this->global->Configuracion->Formato->lServidor
```

Perfil:

```
$this->global->Configuracion->Perfil
```

Datos propios que porta:

Tema que se carga por defecto si el usuario no tiene ninguno especificado:

```
$this->global->Configuracion->Perfil->tema
```

Portal que se ejecuta por defecto:

```
$this->global->Configuracion->Perfil->portal
```

Idioma que por defecto van a tener las etiquetas:

```
$this->global->Configuracion->Perfil->idioma
```

Usuario:

```
$this->global->Configuracion->Perfil->usuario
```

Identificador de la estructura común:

```
$this->global->Configuracion->Perfil->idestructuracomun
```

Denominación de la estructura común:

```
$this->global->Configuracion->Perfil->denomestructuracomun
```

Certificado del usuario:

```
$this->global->Configuracion->Perfil->certificado
```

Identificador del usuario:

```
$this->global->Configuracion->Perfil->idusuario
```

Identificador del área:

```
$this->global->Configuracion->Perfil->idarea
```

Identificador del cargo:

```
$this->global->Configuracion->Perfil->idcargo
```

Moneda:

```
$this->global->Configuracion->Moneda
```

Configuración del identificador de la moneda contable:

```
$this->global->Configuracion->Moneda->idmonedacontable
```

Configuración del identificador del país:

```
$this->global->Configuracion->Moneda->idpais
```

Configuración del identificador de la moneda actual:

```
$this->global->Configuracion->Moneda ->idmoneda
```

Configuración de la descripción:

```
$this->global->Configuracion->Moneda ->descripcion
```

Configuración para el sistema de la moneda alternativa:

```
$this->global->Configuracion->Moneda ->monaltern
```

Configuración del ejercicio:

```
$this->global->Configuracion->Ejercicio
```

Configuración del ejercicio actual:

```
$this->global->Configuracion->Ejercicio->ejercicioactual
```

Periodo:

```
$this->global->Configuracion->Periodo
```

Periodo actual:

```
$this->global->Configuracion->Periodo->periodoactual
```

Fecha contable:

```
$this->global->FechaContable
```

Identificador de la fecha:

```
$this->global->FechaContable->idfecha
```

Fecha:

```
$this->global->FechaContable->fecha
```

Identificador de la fecha contable para el subsistema de la estructura en que se encuentra:

```
$this->global->FechaContable->idestructurasubsist
```

Incremento de la fecha:

```
$this->global->FechaContable->incremento
```

Fecha del sistema:

```
$this->global->FechaContable->fechasist
```

Precedencia:

```
$this->global->FechaContable->precedencia
```

Estructura económica:

```
$this->global->EstructuraEconomica
```

Identificador de la estructura:

```
$this->global->EstructuraEconomica->idestructurae
```

Descripción:

```
$this->global->EstructuraEconomica->descripcion
```

Identificador de la estructura superior:

```
$this->global->EstructuraEconomica->idestructuraepadre
```

Identificador del criterio:

```
$this->global->EstructuraEconomica->idcriterioe
```

Identificador del formato:

```
$this->global->EstructuraEconomica->idformato
```

Caso de uso

Una situación bastante simple es en el caso de gestionar los usuarios de la aplicación que para ejecutar el caso de uso se debe tener el identificador estructura sobre la cual se esta trabajando ya que en el listado de usuarios deben aparecer solo los que pertenecen a la entidad o estructura donde se está utilizando el sistema en el momento.

¿Cómo se obtendría entonces dicha estructura?

```
$estructura=$this->global-> Estructura->identificador
```

2.5.7 Componente Inversión de control **(ZendExt_IoC)**

Problema a resolver

¿Cómo diseñar e integrar sistemas que convivan en una única solución?

Objetivos

Construir una solución para integrar sistemas que convivan en una única solución y mantengan comunicación e integridad de la información (datos) entre ellos.

Especificación de requisitos

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.

Requisitos funcionales

RF1- Integrar subsistemas con transferencia de datos

RF2- Integrar componentes con transferencia de datos

RF3- Integrar componentes y subsistemas por medio interfaz gráfica

Diseño

Diseño de clases

Ver Anexo 7: Diagrama de clases componente ZendExt_IoC

Configuración y uso

Para la utilización del integrador se debe declarar el método de servicio en la carpeta **service** y configurado con antelación el fichero de configuración ioc.xml. La integración puede ser interna o externa cambiando la ubicación del fichero de configuración.

1. **A nivel de lógica**

Integración externa

Sintaxis

$this -> integrator->**sistema**->**servicio** (parámetros);

```
$idestructura = $session->idestructura;
$systemsArray = $this->integrator->seguridad->ObtenerSistemas($certificate, $idestructura);
```

Integración interna

Sintaxis

$this ->pintegrator->**sistema**->**servicio** (parámetros);

```
$datos = $this->pIntegrator->trabajador->ObtenerDatosTrabajador ( $doc->idtrabajador );
$datos1 = $this->pIntegrator->persona->BuscarPersonaID ( $datos->idpersona );
```

2. **A nivel de interfaz**

Compréndase integrar interfaces como el modo de utilizar los elementos y bondades que pueda implementarse en una interfaz que responde a una lógica determinada, desde otra totalmente independiente que necesite del servicio. Es similar a utilizar un servicio web, pero no a nivel de lógica sino a nivel de interfaz.

Para la integración de interfaces se implementaron 2 componentes visuales

Ver Anexo 11: Componente combo-iframe-árbol

Ver Anexo 12: Componente ventana-iframe.

Componente combo-iframe-árbol

Se utiliza cuando necesita se desea seleccionar y tomar de una interfaz determinada información, y esta información tiene una estructura arbórea, de selección simple.

El componente Ext.UCID.ComboIframe es una extensión de la clase Ext.form.ComboBox que tiene dentro de sus características distintivas, que es un combo que despliega un iframe y dentro hoy está preparado para desplegar un árbol e integrarse, mañana puede ser extendido con nuevas funcionalidades, y cumple casi todas las especificaciones de la clase padre.

Para utilizar este componente solo tiene que asegurar que está incluido en la plantilla luego de ext-base.js y ext-all.js el fichero ucid-combo-Iframe.js que está en la carpeta: \repo\VDesarrollo\erp\comun\frameworks\UCID\js y de esta forma puede instanciar y utilizar el componente. En la plantilla (.phtml) utilizando el framework solo tiene que incluir:

```
<script type="text/javascript" src="<? php echo $this->dir_ucid ;?> js/ucid-combo-Iframe.js"></script>
```

Especificaciones de uso

Como se utiliza

Se crea una instancia, almacena una referencia y luego la utiliza como a continuación se describe.

Nota: lo que tiene opcional significa que es la misma propiedad que implementa la clase padre, y que no tiene implicación ninguna para la correcta integración.

1. Defino la variable comboIframe1 donde almaceno una referencia al objeto que construyo a partir de la configuración especificada.

   var comboIframe1 = new Ext.UCID.ComboIframe ({

2. Defino el label del component. (opcional)

   fieldLabel:'Combo Arbol 1',

3. Defino el id que voy a enviar para el servidor (Si tengo el componente dentro de un formulario) con el valor del id del nodo seleccionado en el árbol. (obligatorio si es necesario capturarlo).

valueField: 'idcombo1',

4. Defino el id que voy a enviar para el servidor (Si tengo el componente dentro de un formulario) con el valor del text del nodo seleccionado en el árbol, y el id a partir del cual voy a capturar el componente dentro del iframe, por lo que es clave en el proceso. (obligatorio en cualquier caso).

   id:'combo1',

5. Defino el id a partir del cual voy a capturar el iframe dentro del componente, por lo que es clave en el proceso. (obligatorio en cualquier caso).

   idIframe:'comboIframe1',

6. Defino la acción a través de la que voy a recibir la plantilla a utilizar dentro del iframe. (obligatorio en cualquier caso).

   url:'tpCasoUso.html',

7. Defino el tamaño de la lista a desplegar. (opcional).

   listWidth:200,

8. Evento al que si le coloco una función, esta será llamada automáticamente con el nodo seleccionado. En este componente el evento onSelec es anulado, porque no selecciona de una lista de ítems comunes, pero utilizando esta propiedad, simula el evento anterior de la misma manera.

   onSelectNodo: function (nodo) {},

9. Defino el tamaño del componente. (opcional).

   width: 100

});

Es importante destacar que usted puede añadirle cualquier propiedad de las que implementa un combo ordinario de Ext siempre y cuando la funcionalidad no afecte el funcionamiento de su extensión. Hay algunas propiedades aún conservadas que atentan en contra del correcto funcionamiento y que se irán eliminando a medida que se comiencen a probar. Si el combo está colocado dentro de un formulario, al realizar submit, se envían los valores, y si usted quiere acceder al nodo seleccionado en el árbol, una vez que se selecciona, solo tiene que acceder a la propiedad del combo que

se denomina “nodoSelected” y cada vez que seleccione un nuevo nodo en el árbol, este cambiará su valor y podrá acceder a este a través de comboIframe1.nodoSelected y estará referenciando directamente el nodo en selección en el árbol. Para acceder a las propiedades y configuraciones utilice las bondades de Ext.tree.TreeNode para manipularlo.

No hacemos absolutamente nada con colocar este componente si al desplegarlo no aparece la interfaz adecuada que nos permita seleccionar desde un árbol ¿Verdad?, y esta sería nuestra segunda parte importante.

El que implementa el servicio

Hasta aquí tendríamos un combo que despliega un iframe y dentro del iframe entonces reenderizo la interfaz que me devuelve una acción, pero ¿Cómo logro a nivel lógico que una interfaz se comunique con otra sin tener que implementarla, solo especificando cierta configuración de conexión?

En el sistema de configuración que es donde se deben manejar los datos de la división política-administrativa tiene que existir un componente que me permita desplegarlo en el iframe del combo, y realizar la selección, y que este además me llene el formulario y me devuelva lo que necesito.

Primeramente, en el módulo deben haber sido identificado(s) determinado(s) servicio(s) a nivel de lógica, y este debe ser uno de ellos, básicamente este servicio me devolvería un objeto dado un id para poder ir confeccionando un árbol. Inicialmente habíamos dicho que era muy parecido lo que íbamos a implementar, y de hecho es el mismo servicio, pero que con una interfaz para el usuario permitimos que navegue en el árbol y que escoja el objeto que necesita.

Cumpliendo las características que tiene un servicio web (Fácil de utilizar, sencillo de implementar, lo más explícito posible, lo más atómico, entre otros) realizar una interfaz que me permita lo mismo. ¿Cómo llamarlo? A través del integrador, y vamos a tocarlo más adelante mas detallado.

¿Cómo implementamos la interfaz del servicio?

Partiendo de la idea que es solo de selección, y que podremos seleccionar solo un elemento a la vez, tenemos que implementar (una recomendación con un mínimo de configuración para su uso respondiendo a las características anteriores):

Un árbol (TreePanel**)**:

```
var tpCasoUso = new Ext.tree.TreePanel ({
        autoScroll: true,
        enableDD: false,
        border: false,
        root: new Ext.tree.AsyncTreeNode ({
                text: 'Node0',
                draggable: false,
                id: '0'
        }),
        margins:'2 2 2 2',
        loader: new Ext.tree.TreeLoader ({
                dataUrl:'.../xml/damenodoarbolcombo.php'
        })
});
```

Un viewport (Viewport**):**

```
var vpCasoUso = new Ext.Viewport ({
        layout:'fit',
        items:tpCasoUso
});
```

Hasta aquí aún no hay nada de integración con el otro componente, claro que si usted ejecuta un src a cualquier iframe con la acción que devuelve la vista, se debe visualizar como sigue:

Para garantizar la integración si fuera llamada la vista con este interés, pues hoy tal vez la cree para ello, pero más adelante puede continuar extendiéndose a nuevas funcionalidades, solo tiene que garantizar tres cosas muy sencillas.

1. Tiene que incluir en la plantilla (.phtml) luego de la inclusión de los ficheros de extjs (ext-base.js y ext-all.js) el fichero ucid-integra-interfaz.js como sigue:

<script type="text/javascript" src="<? php echo $this->dir_ucid ;?> js/ucid-integra-interfaz.js"></script>

Y con este fichero incluido el objeto **Ext.UCID** va a garantizar la comunicación entre las interfaces a través de la propiedad (integraInterfaz) con una serie de parámetros que va a pasarle.

2. Capturo el componente que invoca la interfaz de la siguiente manera:

```
if (window.idcomboiframe)

comboParent= window.parent.Ext.getCmp (window.idcomboiframe);
```

La variable global `window.idcomboiframe` siempre va a contener el id del componente que tiene el iframe (combo) la cual se pasa automáticamente a crearse con la configuración definida. En esta línea lo que hago básicamente es preguntar, si tiene algún valor es porque la interfaz fue invocada correctamente por un combo definido, lo capturo en la variable `comboParent` y a partir de aquí puedo integrarme con el tercer paso;

3. Si existe la variable `comboParent` y tiene algún valor ya habíamos dicho que puedo integrar la interfaz, y para ello llamo la función que está definida en el fichero ucid-integra-interfaz.js y puedo referenciarla a través del objeto Ext.UCID.integraInterfaz como sigue:

```
if (comboParent) Ext.UCID.integraInterfaz (tpCasoUso, comboParent);
```

Y le paso los objetos en ese orden primero el `TreePanel` y luego el `combo`. Esta función se encarga de todo lo demás.

El segundo componente lo vamos a utilizar cuando de un conjunto de resultados determinados necesitamos tomar una información específica, o toda la información del record, y tengamos selección simple o múltiple, pero lo fundamental es que nos permite utilizar una lógica totalmente independiente y llamarla dentro de cualquier interfaz.

Componente Ventana WinIframe

El componente Ext.UCID.WinIframe es una extensión del componente Ext.Window que tiene dentro de sus características distintivas, que es una ventana que en su body tiene un iframe y dentro hoy está preparado cargar cualquier interfaz que se necesite

integrar, mañana puede ser extendido con nuevas funcionalidades, y cumple casi todas las especificaciones de la clase padre.

Para utilizar este componente solo tiene que asegurar que está incluido en la plantilla luego de ext-base.js y ext-all.js el fichero ucid-win-iframe.js que está en la carpeta: \repo\VDesarrollo\erp\comun\frameworks\UCID\js y de esta forma puede instanciar y utilizar el componente. En la plantilla (.phtml) utilizando el framework solo tiene que incluir:

```
<script type="text/javascript" src="<? php echo $this->dir_ucid ;?> js/ucid-win-iframe.js"></script>
```

Especificaciones de uso.

Como se utiliza

Se crea una instancia, almacena una referencia y luego la utiliza como a continuación se describe.

Nota: lo que tiene opcional significa que es la misma propiedad del combo, y que no tiene implicación ninguna en la integración.

1. Defino la variable winIframe1 donde almaceno una referencia al objeto que construyo a partir de la configuración especificada.

   var winIframe1 = new Ext.UCID.WinIframe ({

2. Defino el titulo del componente. (opcional)

   title: 'Título de la ventana',

3. Defino el id que va a tener el componente ventana.

   id: 'winIframe1',

4. Defino el id a partir del cual voy a capturar el iframe dentro del componente, por lo que es clave en el proceso. (obligatorio en cualquier caso).

   idIframe: 'idWinIframe1',

5. Defino la acción a través de la que voy a recibir la plantilla a utilizar dentro del iframe. (obligatorio en cualquier caso).

   url: 'gpCasoUso.html',

6. Defino el tamaño de la ventana (en la horizontal).

   width: 400,

7. Defino el tamaño de la ventana (en la vertical).

   height: 250,

8. Defino si voy a proteger con una máscara el body o no.

   modal: true,

9. Defino que acción voy a realizar al oprimir el botón cerrar.

   closeAction: 'hide',

10. Esta va a ser una propiedad que voy a utilizar para colocar variables con algún valor que necesite el servicio para procesar algo. Estas variables se van a poder capturar dentro del iframe una vez que sean referenciadas con el formato que se determine.

    params: {idx:'idparametrox',idy:'idparametroy'},

11. Evento que si especifico como propiedad se dispara cuando se termine de cargar la página que se incluye dentro del iframe. A esta función se le puede pasar cualquier elemento como parámetro, depende del acuerdo al que se llegue con la parte que brinda el servicio.

    onLoad: function () {alert ('ready')},

12. Evento que si especifico como propiedad se dispara cuando realice las acciones que yo defina que implica procesamiento o ejecutar algo en la interfaz. A esta función se le puede pasar cualquier elemento como parámetro, depende del acuerdo al que se llegue con la parte que brinda el servicio.

    doAction: function () {procesarResultados () ;},

13. Arreglo de botones que va a contener la ventana. Generalmente van a ser los botones Aceptar y Cancelar, donde al oprimir el aceptar, realiza determinada acción si es necesario y cierra la ventana, y al oprimir cancelar, siempre cierra la ventana sin realizar ninguna acción, pero se deja a la dependencia de la necesidad del desarrollador.

```
        buttons: [{handler: function () {winIframe1.hide () ;}, text:'Cancelar'},
        {handler:function(){procesarResultados();winIframe1.hide();},
        text:'Aceptar'}]
});
```

Es importante destacar que se puede añadirle cualquier propiedad de las que implementa una ventana ordinaria de Ext siempre y cuando la funcionalidad no afecte el funcionamiento de su extensión. Hay algunas propiedades aún conservadas que atentan en contra del correcto funcionamiento y que se irán eliminando a medida que se comiencen a probar.

Como había mencionado anteriormente, estas configuraciones se especifican de acuerdo a como se necesite brindar el servicio, y tiene que quedar siempre una constancia del contrato de cómo se implementa para que todos los puedan utilizar, y tienen que crearse de tal forma que mañana pueda crecer y no afecte el funcionamiento de lo que tenía anteriormente para no afectar lo desarrollado anteriormente pues de esta forma se dará soporte de manera centralizada.

No se hace absolutamente nada con colocar este componente si al mostrar la ventana no aparece la interfaz adecuada que nos permita realizar las acciones que necesito ¿Verdad?, y esta sería nuestra segunda parte importante.

El que implementa el servicio

Hasta aquí se tendría una ventana que se muestra con un iframe en el body y dentro del iframe entonces reenderezo la interfaz que me devuelve una acción, pero ¿Cómo logro a nivel lógico que una interfaz se comunique con otra sin tener que implementarla, solo especificando cierta configuración de conexión? Pues esa es la segunda parte que expondremos aquí.

En el sistema de logística (ejemplo de subsistema a integrar) que es donde se deben manejar los datos de la existencia de los productos en el almacén, tiene que existir un componente que permita ser renderizado en el iframe de la ventana, y realizar las acciones que necesito, y que este además cumpla con todo lo anteriormente expuesto (que me dispare los eventos, pase los datos, etc.).

Primeramente, en el módulo (logística) deben haber sido identificado(s) determinado(s) servicio(s) a nivel de lógica, y este debe ser uno de ellos, básicamente

este servicio me devolvería un objeto dado un id para realizar una búsqueda de productos. Inicialmente habíamos dicho que era muy parecido lo que íbamos a implementar, y de hecho es el mismo servicio, pero con una interfaz donde permitimos que el usuario navegue por los productos y que escoja el elemento que necesita.

Tendríamos entonces que cumpliendo características que tiene un servicio web (Fácil de utilizar, sencillo de implementar, lo más explícito posible, lo más atómico, entre otros) realizar una interfaz que me permita esto mismo. ¿Cómo llamarlo? A través del integrador, y vamos a tocarlo más adelante mas detallado.

¿Cómo se implementa el servicio?

Partiendo de la idea que es solo un grid, y que podremos seleccionar uno o varios elementos a la vez, tenemos que implementar (una recomendación con un mínimo de configuración para su uso respondiendo a las características anteriores):

Un store (`Store`**)**:

```
stGpGestElemento = new Ext.data.Store ({
        url:'../xml/cargarelementos.php',
        listeners :{'load': function () {gpCasoUso.getSelectionModel ().selectFirstRow () ;}},
        reader: new Ext.data.JsonReader ({
                totalProperty: "total",
                root: "campos",
                id: "idelemento"
                }, [{name:'idelemento'}, {name:'muestra'}, {name:'muestra1'}]
        )
});
```

Un grid (`GridPanel`**)**:

```
gpCasoUso = new Ext.grid.GridPanel ({
        frame: true,
        autoExpandColumn:'expandir',
        store: stGpGestElemento,
        margins:'2 2 2 -4',
```

```
        sm: new Ext.grid.RowSelectionModel (),
        columns: [{Array columnas}],
        loadMask :{ store: stGpGestElemento},
        bbar: new Ext.PagingToolbar ({config})
});
```

Un viewport (Viewport**):**

```
var vpCasoUso = new Ext.Viewport ({
        layout:'fit',
        items:gpCasoUso
});
```

Y con esto estamos listos, claro, garantizando toda la parte de multi-lenguaje y otras características del Framework que tienen que cumplir todos los componentes.

Hasta aquí aún no hay nada de integración con el otro componente, claro que si usted ejecuta un src a cualquier iframe con la acción que devuelve la vista.

Para garantizar la integración si fuera llamada la vista con este interés, pues hoy tal vez la cree para ello, pero más adelante puede continuar extendiéndose a nuevas funcionalidades, solo tiene que garantizar 4 pasos críticos y fundamentales, como a continuación se reflejan:

1. Tiene que incluir en la plantilla (.phtml) luego de la inclusión de los ficheros de extjs (ext-base.js y ext-all.js) el fichero ucid-integra-interfaz.js como sigue:

```
<script type="text/javascript" src="<? php echo $this->dir_ucid ;?> js/ucid-integra-interfaz.js"></script>
```

   Y con este fichero incluido el objeto **Ext.UCID** va a garantizar la comunicación entre las interfaces a través de la propiedad (integraInterfaz) con una serie de parámetros que va a pasarle.

2. Capturo el componente que invoca la interfaz de la siguiente manera:

```
if (window.idWinParent)
winParent = window.parent.Ext.getCmp (window.idWinParent);
```

   La variable global `window.idWinParent` siempre va a contener el id del componente que tiene el iframe (ventana) la cual se pasa automáticamente al

crearse con la configuración definida. Es de vital importancia que se configure para su correcta integración. En esta línea lo que hago básicamente es preguntar, si tiene algún valor, y si lo tiene es porque la interfaz fue invocada correctamente por el componente, lo capturo en la variable `winParent` y a partir de aquí puedo integrarme con el tercer paso;

3. Capturo los parámetros que me son pasados directamente al Iframe en la configuración del componente. Pregunto si existe, y si existe lo capturo en la variable parámetros.

```
if (window.params)

parametros = window.params;
```

La variable global window.params siempre va a contener un objeto a través del cual podrá acceder a variables que son pasadas para el Iframe, siempre que esta sea especificada en la configuración del componente. Es de vital importancia que se configure en acuerdo de ambas partes para poder obtener una correcta integración. Esto va a jugar un papel primordial cuando el servicio requiera de una determinada configuración para su procesamiento. Ej. quiero buscar un comprobante a partir de un identificador, el identificador tiene previamente que haberse pasado en la configuración.

4. Pregunto, si existe la variable `winParent` y si tiene algún valor ya habíamos dicho que puedo integrar la interfaz, y para ello llamo la función que está definida en el fichero ucid-integra-interfaz.js y puedo referenciarla a través del objeto Ext.UCID.integraInterfaz como sigue:

```
if (winParent) {

   Ext.UCID.integraInterfaz (gpCasoUso, winParent);

   winParent.onLoad();

}
```

Y le paso los objetos en ese orden primero el `TreePanel` y luego el `combo`. Esta función se encarga de todo lo demás. Es importante destacar que la funcion onLoad y la de integración tienen que llamarse una vez que la interfaz esté lista para usarse, y es muy importante destacar, que con la función `onLoad()` que viene siendo un evento onLoad para la página que se carga en el Iframe, usted

puede pasar los objetos que desee capturar para manipular en el padre, ya sea funciones, arreglos, etc.

Hasta aquí se han realizado las responsabilidades a nivel de interfaces y se ha descrito que tiene que hacer a grandes rasgos cada parte en la interacción de una interfaz con otra.

El que usa el servicio de la interfaz

Después de crear el componente en el fichero javascript, tiene que haber implementado una acción que coincida con la definida en la propiedad “url” del componente escribiendo el código y dentro de la acción se declara una integración a nivel lógica (externa o interna) de la manera definida

El que brinda el servicio de la interfaz

Después de implementada la interfaz se define el servicio en el fichero (ioc.xml) correspondiente, en le formato establecido.

Ficheros de configuración

El IoC fue desarrollado para ser usado declarativamente esto quiere decir que si se desea integrar algún sistema o subsistema solo se debe dejarlo escrito en un XML en el formato que se muestra a continuación.

El fichero lleva por nombre ioc.xml y en caso que la integración sea entre sistemas (externa) el fichero se encuentra en el directorio de recursos comunes a las aplicaciones y si la integración es interna el fichero se encuentra en el directorio de recursos comunes dentro de cada subsistema o componente. La estructura del xml esta definido de la siguiente manera:

<IoC> inicialización del xml (etiqueta principal)

<Portal reference=”referencia”> Nombre por el cual se invoca el servicio.

<Inyector clase=”” método=””> Etiqueta que referencia la clase y método que implementa el servicio.

<Prototipo> encapsula los parámetros y resultado del servicio.

<Parámetros nombre="" tipo=""> Identifica el parámetro y el tipo[2] del mismo.

<Resultado tipo="array"> muestra el formato del resultado del servicio.

Caso de estudio

Integración externa

```
public function cargarsistemasAction ()
{
    $session = Zend_Registry::get('session');
    $certificate = $session->certificado;
    $idestructura = $session->idestructura;
    $systemsArray = $this->integrator->seguridad->ObtenerSistemas($certificate, $idestructura);
    if(count($systemsArray))
        echo json_encode(array('menu' => $systemsArray));
    else
        throw new ZendExt_Exception('EP007');
}
```

Integración interna

```
function cargarnominasAction(){

                $result = array();
                $docs = DatDocajuste::Nominilla($this->_request->getPost ( 'idnomina' ));
                if(count($docs)>0){
                foreach ($docs as $index =>$doc)
                    {
                    $result[$index]['tiempo'] = $doc->tiempo;
                    $result[$index]['importe'] = $doc->importe;
                    $result[$index]['descripcion'] = $doc->descripcion;
                    $datos = $this->pIntegrator->trabajador->ObtenerDatosTrabajador ( $doc->idtrabajador );
                    $datos1 = $this->pIntegrator->persona->BuscarPersonaID ( $datos->idpersona );
                    $result[$index]['nombre'] = $datos1->nombre.' '.$datos1->papel.' '.$datos1->sapel;
                    }
                echo json_encode(array('cantidad'=>count($result),'documentos'=>$result));
                }else echo json_encode(array('cantidad'=>0,'documentos'=>array()));
}
```

2.5.8 Componente Núcleo y configuración **(ZendExt_App)**

Problema a resolver

El sistema de gestión integral a Cedrux, debe cumplir entre sus procesos de negocios, la gestión de la multi-entidad, el multi-lenguaje y multi-escritorio. El SIGIS (Sistema de Gestión Integral de la Seguridad) brinda los servicios necesarios para cumplir con

[2] Establecido en el fichero de expresiones regulares expressions.xml

estos aspectos. El sistema, en su inicialización, no incluye los aspectos antes mencionados, impidiendo entre sus requisitos a relación entre la multi-entidad y el perfil de usuario.

Objetivos

Construir un componente para la inicialización del sistema integrado con el SIGIS.

Requisitos funcionales

R1.1	Iniciar la sesión de usuario
R1.2	Iniciar el perfil de usuario
R1.3	Iniciar la seguridad de la aplicación
R1.4	Inicializar la conexión a la base de datos
R1.5	Inicializar el controlador frontal

Diseño

Diseño de clases

Ver Anexo 8: Diagrama de clases componente ZendExt_App

Configuración y uso

1. Inicialización de la aplicación y control de las excepciones disparadas por el sistema

```
public function init($config, $idapp = 0) {
    try {
        $this->identificador = $idapp;
        $this->initApp ( $config );
    } catch ( ZendExt_Exception $e ) {
        $e->handle ();
    }
}
```

```
protected function initApp($config) {
    try //Control de Excepciones
    {//Se inicializa la configuracion de la aplicacion
        $this->initConfig ( $config );
        //Se inicializa la configuracion de la session
        $this->initSession (); //Aspect OKP
        //Se inicializa el registro
        $this->initRegister ();
        //Se chequea la seguridad de la aplicacion
        $this->checkSecurity (); //Aspect OKP
        //Se inicializa el controlador frontal
        $this->initFrontController ();
        //Se inicializa la conexion
        $this->initConexion (); //Aspect OKP
        //Se inicializa el perfil del usuario
        $this->initPerfil (); //Aspect OKP
        //Se actualiza el registro
        $this->updateRegister ();
        //Se intenta cargar el controlador solicitado
        $this->frontController->dispatch ();
    } catch ( Exception $e ) //Si se captura una excepcion
    {
if ($e instanceof ZendExt_Exception) //Si la excepcion capturada es de ZendExt
            $e->handle (); //Se dispara la excepcion
        elseif ($e instanceof ZendExt_Exception_NotDefined)
            throw new ZendExt_Exception ( 'NODEFINED', $e );
        else //Si es una excepcion no controlada
            throw new ZendExt_Exception ( 'NOCONTROLLED', $e ); //Se dispara una excepcion controlada
    }
}
```

2. Inicio de la configuración de la aplicación

```
protected function initConfig($config) {
    $configApp = new ZendExt_App_Config ( );
    $this->config = $configApp->configApp ( $config );
}
```

3. Inicio de la sesión del usuario

```
protected function initSession() {
    //Se inicia la sesion.
    Zend_Session::start ();

    $this->session = new Zend_Session_Namespace ( 'ERP_' . $_SERVER ['PHP_AUTH_USER'] );
    //Se protege el id de la session contra ataques.
    if (! isset ( $this->session->initialized )) {
        Zend_Session::regenerateId ();
        $this->session->initialized = true;
    }
```

4. Inicialización de la seguridad de la aplicación

```
protected function checkSecurity() {
    $security = ZendExt_Aspect_Security_Sgis::getInstance ();
    $security->authenticateUser ();
}
```

5. Inicialización del controlador frontal

```
protected function initFrontController() {
    //Se inicia el controlador frontal.
    $this->frontController = Zend_Controller_Front::getInstance ();
    //Se obtiene la peticion o solicitud al servidor web
    $this->request = $this->frontController->getRequest ();
    //Se activan las excepciones
    $this->frontController->throwExceptions ( true );
    //Se especifica el directorio de los controladores
    $this->frontController->setControllerDirectory ( $this->config->controllers_path );
    if (count ( $this->controllerArray )) //Si el array de controladores externos no esta vacio

        //Se adicionan los controladores externos
        foreach ( $this->controllerArray as $controller ) {
            //Adiciono cada uno de los controladores
            $this->frontController->addControllerDirectory ( $controller [0], $controller [1] );
        }
    }
}
```

6. Inicialización de la conexión a la base de datos

```
protected function initConexion() {
    //Obtengo una instancia del administrador de transacciones
    $transactionManager = ZendExt_Aspect_TransactionManager::getInstance ();
    //Creo la conexion activa
    $this->conexion = $transactionManager->openConections ( null, true );
}
```

7. Inicialización del perfil de usuario

```
public function initPerfil() {
    $security = ZendExt_Aspect_Security_Sgis::getInstance ();
    $security->getUserData ();
}
```

Ficheros de configuración

La configuración de este componente para la inicialización de la aplicación esta contenido en el fichero config.php ubicado en la carpeta raíz de la aplicación. Donde se incluye:

Dirección del Marco de trabajo relativa a la carpeta de publicación

```
$dir_rel_mt = substr($_SERVER['SCRIPT_NAME'], 0, strrpos($_SERVER['SCRIPT_NAME'], 'aplicaciones'));
```

Dirección de la carpeta de publicación

```
$dir_www = $_SERVER['DOCUMENT_ROOT'];
```

Dirección absoluta del marco de trabajo

```
$dir_abs_mt = str_replace('//','/', $dir_www . $dir_rel_mt);
```

Dirección del módulo en ejecución

```
$dir_modulo = substr($_SERVER['SCRIPT_FILENAME'], 0, strrpos($_SERVER['SCRIPT_FILENAME'], '/') + 1);
$config['modulo_path'] = $dir_modulo;
```

Include_Path de PHP con la dirección de los frameworks y los modelos de los módulos.

```
$config['include_path'] = '.' . PATH_SEPARATOR . $dir_abs_mt . 'comun/frameworks/'
        . PATH_SEPARATOR . $dir_abs_mt . 'comun/frameworks/Doctrine'
        . PATH_SEPARATOR . $dir_abs_mt . 'utiles/impresion'

        . PATH_SEPARATOR . $dir_modulo . 'models/bussines'
        . PATH_SEPARATOR . $dir_modulo . 'models/domain'
        . PATH_SEPARATOR . $dir_modulo . 'models/domain/generated'
        . PATH_SEPARATOR . $dir_modulo . 'validators'
        . PATH_SEPARATOR . $dir_modulo . 'services'
        . PATH_SEPARATOR . $dir_abs_mt . 'comun/frameworks/ZendExt/InterpreterRules/models/domain'
        . PATH_SEPARATOR . $dir_abs_mt . 'comun/frameworks/ZendExt/InterpreterRules/models/domain/generated'
        . PATH_SEPARATOR . $dir_abs_mt . 'comun/frameworks/ZendExt/Traza/models/bussiness'
        . PATH_SEPARATOR . $dir_abs_mt . 'comun/frameworks/ZendExt/Traza/models/domain'
        . PATH_SEPARATOR . $dir_abs_mt . 'comun/frameworks/ZendExt/Traza/domain/generated'
        . PATH_SEPARATOR . $dir_abs_mt . 'comun/frameworks/ZendExt/Trace/domain'
        . PATH_SEPARATOR . $dir_abs_mt . 'comun/frameworks/ZendExt/Trace/domain/generated';
```

$config: variable contenedora de la configuración general del marco de trabajo

Configuración de los xml

```
$config ['xml']['expresiones']       = $dir_abs_mt . 'comun/recursos/xml/expressions.xml';
$config ['xml']['tipos_excepciones'] = $dir_abs_mt . 'comun/recursos/xml/tipos_excepciones.xml';
$config ['xml']['aspect']            = $dir_abs_mt . 'comun/recursos/xml/aspect.xml';
$config ['xml']['aspecttemplate']    = $dir_abs_mt . 'comun/recursos/xml/aspecttemplate.xml';
$config ['xml']['aspecttemplatemt']  = $dir_abs_mt . 'comun/recursos/xml/aspecttemplatemt.xml';
$config ['xml']['log']               = $dir_abs_mt . 'comun/recursos/xml/log.xml';
$config ['xml']['traza']             = $dir_abs_mt . 'comun/recursos/xml/CategoriaTipo.xml';
$config ['xml']['concepts']          = $dir_abs_mt . 'comun/recursos/xml/concepts.xml';
$config ['xml']['traceconfig']       = $dir_abs_mt . 'comun/recursos/xml/traceconfig.xml';
```

Fichero de log de excepciones

```
$config ['exception_log_file'] = $dir_abs_mt . 'comun/recursos/xml/exception.log';
```

Configuración de ZendExt_Cache

```
$config['cache']['frontend']               = 'Core'; //Cachear instancias de clases
$config['cache']['backend']                = 'File'; //Cachear en ficheros
$config['cache']['lifetime']               = 7200; //Tiempo de Vida
$config['cache']['automatic_serialization'] = true; //Serializar
$config['cache']['cache_dir']              = $dir_www . '/tmp/'; //Directorio de cache
$config['cache']['chmod']                  = 0644; //Directorio de cache
```

Dirección de las fuentes

```
$config['fonts_path'] = $dir_abs_mt . 'utiles/fuentes';
```

Nombre de la carpeta que contiene los Controladores dentro de los módulos

```
$config['controllers_path'] = 'controllers';
```

Dirección del Framework de Presentación EXTJS relativa a la carpeta de publicación

```
$config['extjs_path']                = $dir_rel_mt . 'comun/frameworks/ExtJS/';
$config['idioma']['es']['extjs_path'] = $dir_rel_mt . 'comun/frameworks/ExtJS/idioma/es/';
$config['idioma']['en']['extjs_path'] = $dir_rel_mt . 'comun/frameworks/ExtJS/idioma/en/';
```

Dirección del Framework UCID relativa a la carpeta de publicación

```
$config['ucid_path']     = $dir_rel_mt . 'comun/frameworks/UCID/';
$config['schema_reglas'] = 'mod_arquitectura';
$config ['enable_security'] = true;
```

Caso de estudio

Para la inicialización de la aplicación solo se debe configurar previamente el fichero de configuración (config.php) y posteriormente instanciar ZendExt_App.

2.5.9 Componente Trazas **(ZendExt_Trace)**

Problema a resolver

En el sistema en e numerables ocasiones ocurren errores que son muy difíciles de detectar o identificar, además no se tiene un registro de los eventos que ocurren en dicho sistema por lo que se hace necesario un componente que notifique la ocurrencia y una aplicación que posibilite la visualización de estos factores.

Objetivos

Desarrollar un componente que detecte y registre los eventos del proceso de negocio del sistema en todos sus subsistemas y que visualice este proceso.

Requisitos funcionales

Permitir buscar trazas por:

- ✓ Fecha de generación
- ✓ Tipo
- ✓ Categoría

Generar una traza siempre que:

- ✓ Se inicia una acción en el sistema
- ✓ Se produce un acceso a datos

- ✓ Se visite una URL
- ✓ Se termine una acción en el sistema
- ✓ Se produzca un error en una acción
- ✓ Se produzcan excepciones
- ✓ Se produzcan errores en el IoC externo
- ✓ Se produzcan errores en el IoC interno
- ✓ Se ejecute IoC interno
- ✓ Se ejecute IoC externo

Notificar a los sistemas del ERP la ocurrencia de un evento determinado

Exportar las trazas

Diseño

Diseño de clases

Ver Anexo 9: Diagrama de clases componente ZendExt_Trace

Configuración y uso

Para el seguimiento de las trazas del sistema se Diseñó e implementó una aplicación que permite el registro de los eventos que ocurren en el sistema permitiendo de esta forma poder corregir problemas que se presenten. Estos eventos serían:

- ✓ Inicio de una acción: Se dispara un evento al comienzo de una acción.
- ✓ Terminación de una acción: Se dispara un evento al finalizar una acción.
- ✓ Error en una acción: Se dispara un evento cuando en una acción ocurre un error.
- ✓ Autenticar usuario: Se dispara un evento por cada URL visitada.
- ✓ IoC Externo: Se dispara un evento cuando ocurre integración entre diferentes subsistemas.
- ✓ IoC Interno: Se dispara un evento cuando ocurre integración entre diferentes componentes de un subsistema.
- ✓ Excepciones: Se dispara un evento cuando ocurre una excepción en el sistema.
- ✓ Rendimiento: Es el tiempo de ejecución de una acción.
- ✓ Error IoC Externo: Se dispara un evento cuando ocurre un error en la integración entre diferentes subsistemas.
- ✓ Error IoC Interno: Se dispara un evento cuando ocurre un error en la integración entre diferentes componentes de un subsistema.

Caso de estudio

La gestión de las trazas presenta una interfaz gráfica la cual mediante la disposición de elementos para la búsqueda avanzada de las trazas generadas en todo el sistema.

Elementos

- ✓ Rango de fechas
- ✓ Categoría: Engloba todos los subsistemas.
- ✓ Tipo: Engloba los tipos de trazas que se generan (Acción, Excepción, Rendimiento, Integración, Excepción de integración).

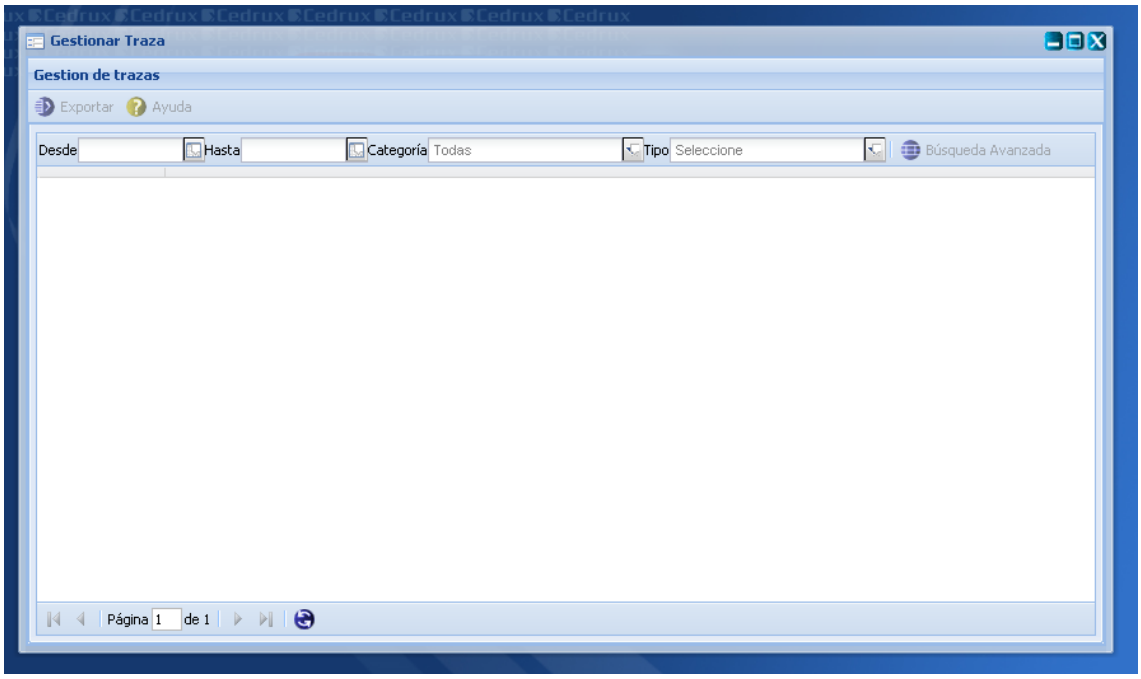


Figura 7: Interfaz de gestión de trazas

Ejemplo: Todas las trazas de **Acción** del subsistema de **Estructura y composición**

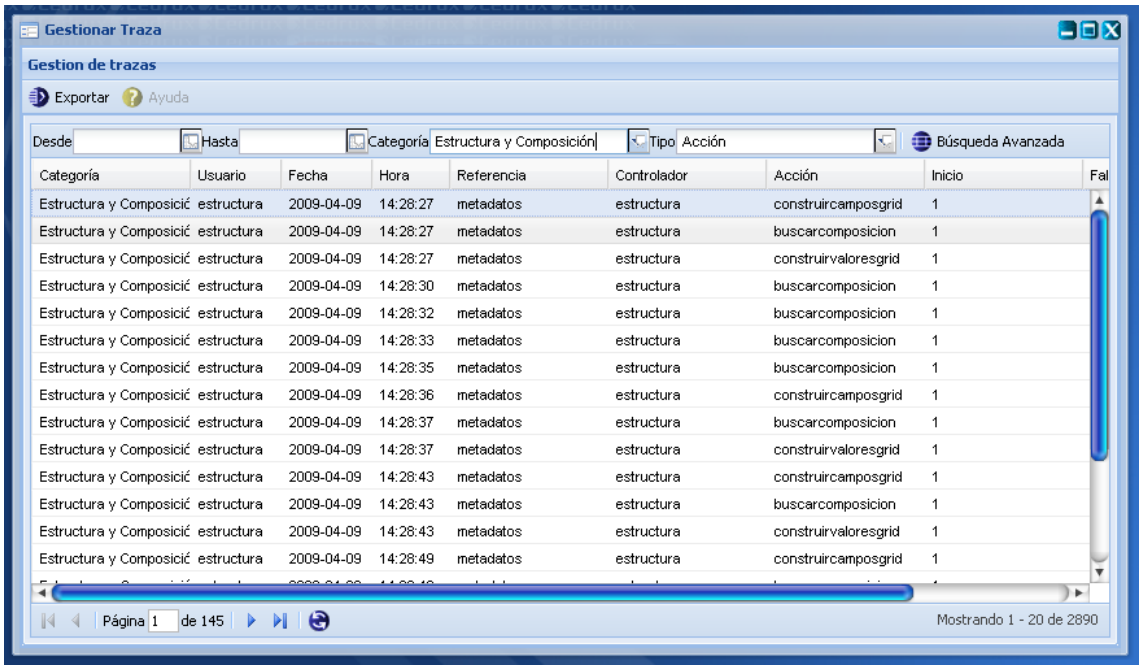


Figura 8: Interfaz de gestionar trazas. Búsqueda

Esta interfaz brinda la posibilidad de exportar las trazas seleccionadas en formato xml.

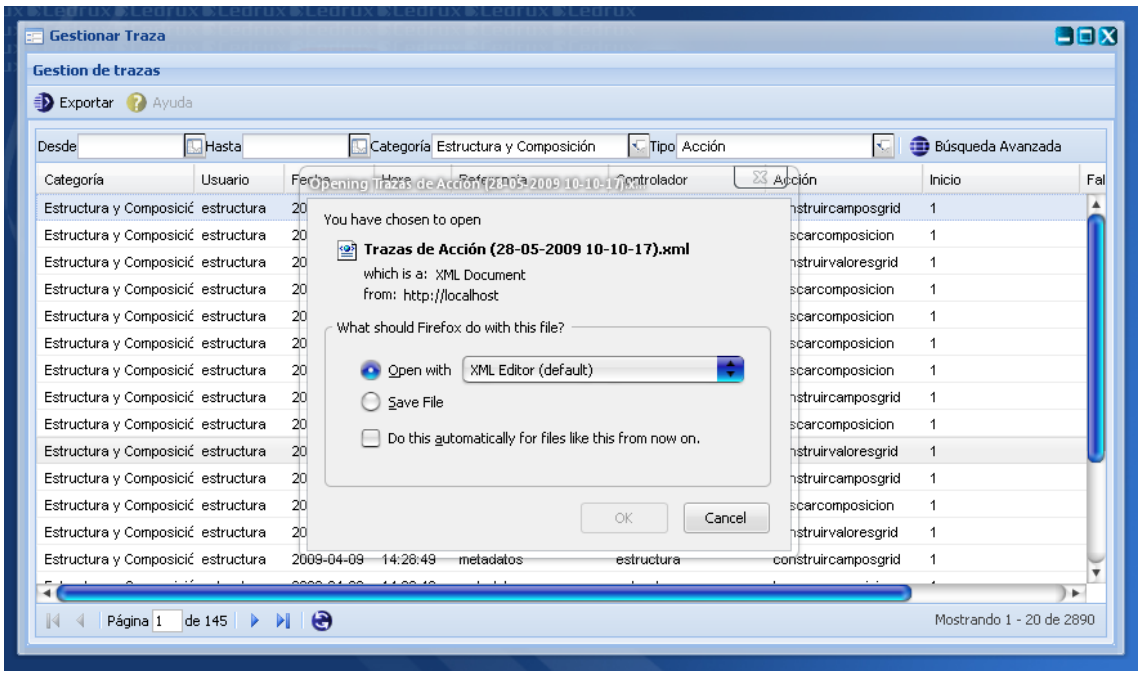


Figura 9: Interfaz gestionar trazas. Exportar trazas

De la funcionalidad **exportar** trazas devuelve un xml en el siguiente formato:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <xml>
  - <trazas tipo="Acción">
    - <traza_1>
        <idestructuracomun>100000001</idestructuracomun>
        <usuario>estructura</usuario>
        <hora>14:28:27</hora>
        <fecha>2009-04-09</fecha>
        <referencia>metadatos</referencia>
        <controlador>estructura</controlador>
        <accion>construircamposgrid</accion>
        <inicio>1</inicio>
        <falla>No posee</falla>
        <categoria>Estructura y Composición</categoria>
      </traza_1>
    - <traza_2>
        <idestructuracomun>100000001</idestructuracomun>
        <usuario>estructura</usuario>
        <hora>14:28:27</hora>
        <fecha>2009-04-09</fecha>
        <referencia>metadatos</referencia>
        <controlador>estructura</controlador>
        <accion>buscarcomposicion</accion>
        <inicio>1</inicio>
        <falla>No posee</falla>
        <categoria>Estructura y Composición</categoria>
      </traza_2>
```

2.5.10 Componente Validación **(ZendExt_Validation)**

Problema a resolver

La gestión de la seguridad en el aspecto de las validaciones de los datos que se introducen a las aplicaciones es muy baja ya que solamente se realizan dichas comprobaciones del lado del cliente.

Objetivos

Desarrollar, adaptar o integrar un componente al marco de trabajo que permita a las aplicaciones que se están desarrollando la validar todas aquellas acciones y datos que lo requieran, antes de ser procesados por el controlador.

Especificación de requisitos

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.

Requisitos funcionales

R1 – Validar acción.

R1.1 – Buscar acción.

R1.2 – Validar los datos según el tipo especificado.

R1.3 – Chequear precondición.

Diseño

Diseño de clases

Ver Anexo 10: Diagrama de clases componente ZendExt_Validation

Configuración y uso

Este componente se utiliza declarativamente por lo que el desarrollador solo debe entrar los datos pertinentes en cada uno de los XML que se utilizan. En el XML se registran cada una de las precondiciones que se deben cumplir para que una acción sea ejecutada así como la clase y el método que ejecutara cada una de estas precondiciones. También se pueden validar parámetros para esto se debe registrar el nombre de cada uno de los parámetros que viajan por el POST y el tipo de parámetro que es. El tipo de parámetro se consulta en el XML de **expressions**.

Ficheros de configuración

El validador fue desarrollado para ser usado declarativamente esto quiere decir que si se desea validar algún dato o acción en un sistema solo se debe dejarlo escrito en un XML en el formato que se muestra a continuación.

El fichero lleva por nombre validation.xml y se encuentra en el directorio de recursos comunes a las aplicaciones.

En dicho fichero **validation.xml** serán declarados todos los datos y acciones que se desean validar, la aplicación a la que pertenecen y el controlador que la ejecutara si es válida. Después de entrar estos datos se procede a especificar las precondiciones y dentro de ellas las validaciones que se deben verificar. En los validadores se especifican el nombre de la clase y dentro de ella el método que ejecutara esta validación, con esta información el componente automáticamente cargara esta clase y ejecutara el método, si el método retorna true quiere decir que la validación fue exitosa y si retorna false inmediatamente mostraremos al usuario una excepción que describa cual fue la validación que no tuvo éxito. Este componente esta completamente integrado al componente de excepciones por lo cual todas estas serán descritas por el usuario declarativamente en el XML de excepciones.

```
<validador>
    <usuario>
        <GestusuarioController>
            <modificarusuarioAction>
                <precondition>
                    <validator class="UsuarioValidator" method="validarExistenciaUsuario" error="EGU010"/>
                </precondition>
                <precondition>
                    <validator class="UsuarioValidator" method="validarConfirmacionClave" error="EGU009"/>
                    <validator class="UsuarioValidator" method="validarRepeticionClave" error="EGU008"/>
                </precondition>
                <param name="idpersona" type="enterospos" not_null="true" null_error="EGU011" type_error="EGU012"/>
                <param name="version" type="enterospos" not_null="true" null_error="EGU013" type_error="EGU014"/>
                <param name="alias" type="letras" not_null="true" null_error="EGU015" type_error="EGU016"/>
                <param name="nombre" type="letras" not_null="true" null_error="EGU017" type_error="EGU018"/>
                <param name="papell" type="letras" not_null="true" null_error="EGU019" type_error="EGU020"/>
                <param name="sapell" type="letras" not_null="false" type_error="EGU021"/>
                <param name="ididioma" type="enterospos" not_null="true" null_error="EGU022" type_error="EGU023"/>
                <param name="idtema" type="enterospos" not_null="true" null_error="EGU024" type_error="EGU025"/>
                <param name="idespecialidad" type="enterospos" not_null="false" type_error="EGU026"/>
            </modificarusuarioAction>
```

Donde la línea de código encerrada en las etiquetas de la precondición:

```
<validator class="GestpagosadicionalesValidator" method="validarExistenciaPagos" error="CHPA203"/>
```

No es más que el indicador de la clase donde se encuentra la acción que se debe ejecutar antes del método (`class="GestpagosadicionalesValidator"`), el método de esta clase que se debe cumplir (`method="validarExistenciaPagos"`) y el código de la excepción que se va a lanzar en caso de que el método indicado devuelva “false”, es decir en el caso de que la precondición no se cumple, que es cuando se deja de ejecutar todo lo demás y se muestra en pantalla la excepción indicada.

Si todas las validaciones tuvieron éxito y en la acción también se quieren validar los datos que han sido ingresados pues se debe declarar cada uno de estos parámetros y el tipo que es. Como muestra la línea que se indica al terminarse de especificar las precondiciones:

```
<param name="denom" type="letras" not_null="true" null_error="CHPA208" type_error="CHPA216"/>
```

Indica que el parámetro con un nombre determinado (`name="denom"`) debe cumplir con la expresión regular indicada en el fichero expressions.xml (`type="letras"`), luego se indica si puede ser un parámetro vacio (`not_null="true"`), que vale aclarar que este atributo podrá llevar valores solo de “true” si el parámetro no puede ser vacío o “false” si puede estarlo, y finalmente se indican los códigos de las excepciones a lanzar en el caso de que el parámetro no cumpla con uno u otra

condición (`null_error="CHPA208" type_error="CHPA216"`). Aclarar que si un parámetro puede estar vacío (`not_null="false"`) no se especifica ninguna excepción a lanzar en el caso de que esta vacío.

Para declarar el tipo de los datos es necesario consultar el XML de expresiones llamado
expressions.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<expressions>
<enterospos>^[0-9]+$</enterospos>
<numerosguionbajo>^[0-9]+(_[0-9]+)*$</numerosguionbajo>
<letras>^([a-zA-ZáéíóúñüÑ]+ ?[a-zA-ZáéíóúñüÑ]*)+$</letras>
<numerosletras>^([a-zA-Z0-9áéíóúñüÑ]+ ?[a-zA-Z0-9áéíóúñüÑ]*)+$</numerosletras>
<email>^[^@ ]+@[^@ ]+.[^@ .]+$</email>
<real>^-?^[0-9]+(.?[0-9]*)?$</real>
<ireal>^-?[0-9]+(\.[0-9]*)?$</ireal>
<fecha>([0-9]{1,2})/([0-9]{1,2})/([0-9]{4})</fecha>
<fechacont>([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})</fechacont>
</expressions>
```

Para consultar hay que ver si el tipo de parámetro y expresión regular que lo valida se encuentra declarados. Si no están declarados pues se declaran los parámetros propios y las expresiones que lo validan, siempre respetando el formato estándar. Una vez introducidos todos estos datos cada vez que se ejecute la acción a validar, pasará por este componente de validación y si todas las precondiciones fueron exitosas y cada uno de los parámetros es del tipo especificado pues se ejecutará la acción en el controlador y seguirá el curso normal de los eventos.

Caso de estudio

El caso de uso “Gestionar Usuario” perteneciente al sub-módulo de ‘Registro’ que a su vez se inserta en el modulo ‘usuario’, lleva incluido la modificación de la información de usuarios en la base de datos pero para modificar un usuario es necesario comprobar que el mismo ya esté insertado con anterioridad en la base de datos. Luego de verificar esta precondición, si no se cumple no se ejecuta el método de modificar, se debe validar los datos que se pasan por parámetro a la acción, estos son:

- ✓ ci: carnet de identidad, identificador de la persona que debe estar constituido por números enteros positivos.
- ✓ alias: alias del usuario que debe estar constituido por letras solamente.

- ✓ nombre: nombre real del usuario que debe estar constituido por letras solamente.
- ✓ papell: primer apellido que debe estar constituido por letras solamente.
- ✓ email: dirección electrónica en el formato correspondiente.

El carnet, el alias y el e-mail son parámetros indispensables, o sea no pueden dejar de introducirse o quedar vacios. Sin embargo el nombre y el primer apellido no son esenciales y pueden estar vacíos.

Además las excepciones a lanzar en caso de errores están registradas bajo el código de 'PREC01' en el caso de que falle la precondición y bajo el de 'PAR02' en el caso de que algún parámetro no cumple con algún formato indicado.

La clase validadora del caso de uso lleva por nombre `'UsuarioValidator'` y el método que va a comprobar la precondición debe identificarse como `'validarExistenciaUsuario'`.

¿Cómo quedaría registrada y configurada la validación en el servidor en este caso? ¿Cómo quedaría programada la acción de la precondición?

Registro y configuración:

En el xml llamado validation.xml se adiciona las validaciones del módulo 'usuario' como se muestra:

```
:validador>
    <usuario>
        <GestusuarioController>
            <modificarusuarioAction>
                <precondition>
                    <validator class="UsuarioValidator" method="validarExistenciaUsuario" error="EGU010"/>
                </precondition>
                <precondition>
                    <validator class="UsuarioValidator" method="validarConfirmacionClave" error="EGU009"/>
                    <validator class="UsuarioValidator" method="validarRepeticionClave" error="EGU008"/>
                </precondition>
                <param name="idpersona" type="enterospos" not_null="true" null_error="EGU011" type_error="EGU012"/>
                <param name="version" type="enterospos" not_null="true" null_error="EGU013" type_error="EGU014"/>
                <param name="alias" type="letras" not_null="true" null_error="EGU015" type_error="EGU016"/>
                <param name="nombre" type="letras" not_null="true" null_error="EGU017" type_error="EGU018"/>
                <param name="papell" type="letras" not_null="true" null_error="EGU019" type_error="EGU020"/>
                <param name="sapell" type="letras" not_null="false" type_error="EGU021"/>
                <param name="ididioma" type="enterospos" not_null="true" null_error="EGU022" type_error="EGU023"/>
                <param name="idtema" type="enterospos" not_null="true" null_error="EGU024" type_error="EGU025"/>
                <param name="idespecialidad" type="enterospos" not_null="false" type_error="EGU026"/>
            </modificarusuarioAction>
```

Método pre condicional `'validarExistenciaUsuario'` que pertenece a la clase `'UsuarioValidator'`:

```
Public function validarExistenciaUsuario ()//Comprueba que el usuario
existe.

{//Obtención del usuario con el alias que se desea asignar.

      $usuario = DatUsuario::getUsuarioByAlias ($this->post
['alias']);

      If ($usuario->count () && $usuario [0] ->ci! = $this-> post
['ci'])

            Return false;

      Return true;

      }
```

Esta función se va a ejecutar antes que el método y si devuelve 'false' entonces se dispara la excepción cuyo código es el especificado en el xml de las validaciones (PREC01).

2.6 Conclusiones

Con la realización de la documentación técnica de los componentes de marco de trabajo se incrementa el entendimiento de los desarrolladores del sistema Cedrux como para los futuros clientes que utilizarán el marco de trabajo para construir otras soluciones. Además en los componentes del marco de trabajo se han realizado un conjunto de innovaciones tecnológicas en lenguaje PHP, las cuales pueden ser reutilizadas por la comunidad de PHP de la universidad y otros proyectos que tengan soluciones en común con el sistema Cedrux. Para poner en disposición esta documentación se da la tarea de construir un portal para la gestión de la información del marco de trabajo de sistema de gestión Cedrux.

3

CAPÍTULO III Portal de Documentación Técnica

3.1 Introducción

En la actualidad la socialización de la información o el conocimiento en la web se ve materializada a través de los portales. Muchos de estos son construidos y maniobrados mediante los CMS o Sistema de Gestión de Contenidos, los cuales brindan una gran cantidad de ventajas. Durante el desarrollo del sistema de gestión Cedrux se han realizadas innovaciones tecnológicas, las cuales son reutilizadas por otros proyectos de la universidad. Con la disposición de un portal, donde, los diversos desarrolladores de los distintos proyectos pueden reutilizar dichos avances, incrementaría el trabajo colaborativo en la universidad, principalmente los que desarrollan en el lenguaje PHP.

3.2 Objetivo

La documentación técnica del marco de trabajo, en el proyecto ERP- Cuba, es para el desarrollo del sistema Cedrux es de vital importancia, pues si prescindimos de esta, los encargados de desarrollar el sistema de gestión integral tendrían desconocimiento del funcionamiento de los aspectos arquitectónicos del marco de trabajo, trayendo como consecuencia retraso en los cronogramas de desarrollo e incorrecta solución y codificación de las funcionalidades de los distintos subsistemas. La disposición de la documentación técnica de los componentes del marco de trabajo, de manera práctica y específica, en un lugar de fácil acceso para todo el personal que interactúa en el proyecto ERP-Cuba, facilitaría la construcción del sistema Cedrux, la solución a errores arquitectónicos en el menor tiempo posible y ayudaría a la optimización de los distintos componentes y subsistemas.

La construcción del marco de trabajo para el sistema de gestión integral Cedrux, ha traído incontables innovaciones arquitectónicas en el lenguaje PHP, sirviendo de reutilización para la soluciones de sistema de esta índole. La habilitación del portal de gestión documental del marco de trabajo del sistema de gestión integral Cedrux,

aumentaría el trabajo colaborativo de la Universidad de las Ciencias Informáticas y de la comunidad de software libre y PHP de dicha universidad.

3.3 Estructura

3.3.1 Diseño

- En la figura 10 se muestra el encabezado, este está compuesto por un banner y la barra del menú principal de navegación el cual está compuesto por las siguientes páginas:

Figura 10: Encabezado

1. **Inicio**

   En esta página se les dará información a los usuarios de las actividades que se desarrollan en el proyecto ERP-Cuba. Noticias nacionales, internacionales y de la universidad.

2. **Arquitectura**

   Expediente de la arquitectura

   En la construcción de sistemas de software resulta elemental diseñar una arquitectura de software que soporte el funcionamiento del mismo. Pero no sólo basta con un buen diseño de la arquitectura, esta debe ser comprendida por los desarrolladores para que pueda ser gestionada, mantenida e implementada. Es por ello que se debe realizar una documentación de la arquitectura que facilite estos aspectos.

   Se propone realizar la descripción de la arquitectura mediante 7 vistas: vista de datos, vista de sistema, vista de seguridad, vista tecnológica, vista de infraestructura, vista de presentación y vista de integración. Para oficializar esta propuesta se elabora un expediente para la documentación técnica de la

arquitectura de software con el objetivo de estructurar y guiar organizacionalmente la construcción de la arquitectura del software.

Modelo de desarrollo

En aras de mejorar la organización, control y rendimiento en el proceso de desarrollo de software tecnológico en la Subdirección Técnica del Centro de Desarrollo de Soluciones Integrales para la Gestión de Entidades (CESGE), se ha decidido organizar los procesos que deben desarrollarse en la misma. Con esta finalidad y luego de un estudio detallado de los modelos de procesos de desarrollo de software y estructuras organizativas existentes y basándose en las buenas prácticas de las metodologías ágiles para lograr rapidez y productividad en la construcción de tecnología, se realiza una propuesta de modelo que incluye en una primera parte, los procesos que debe llevarse a cabo y posteriormente basado en dicho modelo la propuesta de estructura organizativa con que debe contar la Subdirección Técnica para que todo funcione eficientemente y como un todo. Se tienen en cuenta además, los procesos horizontales que deben desarrollarse en la Subdirección y que van unidos al proceso productivo como es el caso de la formación académica, la gestión de recursos humanos, la calidad de los componentes desarrollados y la investigación. Destacar que este modelo no solo incluye los procesos internos que deben tener lugar en la Subdirección Técnica específicamente, describe además de manera global los macro-procesos esenciales que deben desarrollarse en el centro y en cuyo flujo de actividades la Subdirección juega un papel fundamental.

3. Marco de trabajo

3.1 Documentación

"Importancia de la documentación"

La documentación es un conjunto de información que expresa detalles de los frameworks, la forma adecuada de operar con ellos, nos permite interpretar los errores, conocer su proceso, etc., si no tenemos la documentación que nos indique que es lo que hace y como lo hace, sería muy difícil y casi imposible entender y corregir errores de los sistemas.

La documentación de los programas es un aspecto sumamente importante, tanto en el desarrollo de la aplicación como en el mantenimiento de la misma. Mucha gente no hace esto parte del desarrollo y no se da cuenta de que pierde

la posibilidad de la reutilización de parte del programa en otras aplicaciones, sin necesidad de conocerse el código al dedillo.

La documentación de un programa empieza a la vez que la construcción del mismo y finaliza justo antes de la entrega del programa o aplicación al cliente. Así mismo, la documentación que se entrega al cliente tendrá que coincidir con la versión final de los programas que componen la aplicación.

3.1.1 Caso de estudio

En el caso de estudio “Usuario” se encuentran ejemplificados todos los componentes que respaldan los aspectos arquitectónicamente significativos. Además se encuentra la ayuda al caso de estudio la cual muestra de forma detallada al lector explicaciones del trabajo del caso de uso de usuario, se aclara paso a paso todo un flujo de información del componente. Con todo ello se persigue el objetivo de lograr en los programadores del proyecto un mejor conocimiento sobre los componentes del framework y su integración a los subsistemas del proyecto, además se persigue la obtención de un aprendizaje superior sobre el marco de trabajo, su configuración y el estándar que se persigue por parte de arquitectura y por supuesto un mayor alcance sobre el lenguaje de programación que se define.

3.1.2 Especificación técnica

Las características del marco se conceptualizan de las propias definiciones de los componentes y clases de desarrollo, dándole así un perfil técnico al marco de trabajo del proyecto ERP – Cuba. A continuación se le especifican los componentes y las clases fundamentales definidos por el equipo de arquitectura de dicho proyecto:

Componente ZendExt

Este componente es una extensión del framework ZendFramework el cual reutiliza el Modelo-Vista –Controlador (MVC).

Componente IOC (Inversión de control)

Permite la integración entre componentes y subsistemas. La inversión de control se hace necesaria para gestionar las dependencias entre subsistemas y el framework.

Componente Global Concept

Permite la utilización de elementos comunes con cada definición.

Nomencladores genéricos

Permiten unificar nomencladores comunes en una aplicación, además de la gestión de los mismos. Según la definición más difundida metadatos son «datos sobre datos». Debido a que muchas veces no se tiene en cuenta la diferencia entre datos e informaciones también hay muchas declaraciones como «informaciones sobre datos», «datos sobre informaciones» y «informaciones sobre informaciones».
Los metadatos pueden describir colecciones de objetos y también los procesos en los que están involucrados, describiendo cada uno de los eventos, sus componentes y cada una de las restricciones que se les aplican. Los metadatos definen las relaciones entre los objetos, como las tuplas en una base de datos o clases en orientación a objetos, generando estructuras.

Componente Aspect

Trabaja sobre funciones programadas orientadas a objetos, la cual unifica los aspectos generales de la aplicación que luego, pueden ser configuradas para la misma. Una de las características a destacar es la separación de responsabilidades (SOC. (inv.)) Patrón de integración.

Componente ManagerException

Este componente realiza varias características funcionales como son; la unificación y el manejo de excepciones en la aplicación, en donde permite configurarlo de forma declarativa en un XML.

Componente Validator

EL componente de validación se utiliza para validar todas aquellas acciones y datos que lo requieran. Antes de ser procesados por el controlador. Este componente se utiliza declarativamente por lo que el desarrollador solo debe entrar los datos pertinentes en cada uno de los XML que se utilizan. En el XML se registran cada una de las precondiciones que se deben cumplir para que una acción sea ejecutada así como la clase y el método que ejecutará cada una de estas precondiciones. También se pueden validar parámetros, para esto se debe registrar el nombre de cada uno de los parámetros que viajan por el POST y el tipo de parámetro que es. El tipo de parámetro se consulta en el XML de expresiones.

Componente App

Sirve de inicializador de toda la configuración, los manejadores de excepciones y aspectos y de toda la gestión de la seguridad del sistema.

Componente Exportador – Importador

Permite exportar o importar a partir de un fichero XML los datos existentes de otra aplicación. Esto ocurre en un formato definido por la aplicación Cedrux. Contiene una referencia al componente IOC y Global Concept.

Componente Trazas

Las trazas permiten en el desarrollo de software crear un mecanismo de registro oficial de eventos durante un período de tiempo en particular, además de registrar datos o información sobre quien, que, cuando, donde y por que un evento ocurre para un dispositivo en particular o aplicación. Todo esto permite monitorear las actividades de la aplicación o dispositivo, donde se puede obtener una buena oportunidad para determinar eventos y tomar la acción necesaria para corregir el problema o iniciar una investigación en caso de un incidente de seguridad.

3.1.3 **Manuales**

La documentación que se entrega al cliente se divide claramente en dos categorías, interna y externa:

* Interna: Es aquella que se crea en el mismo código, ya puede ser en forma de comentarios o de archivos de información dentro de la aplicación.

* Externa: Es aquella que se escribe en cuadernos o libros, totalmente ajena a la aplicación en si. Dentro de esta categoría también se encuentra la ayuda electrónica.

La guía técnica

En la guía técnica o manual técnico se reflejan el diseño del proyecto, la codificación de la aplicación y las pruebas realizadas para su correcto funcionamiento. Generalmente este documento esta diseñado para personas con conocimientos de informática, generalmente programadores.

El principal objetivo es el de facilitar el desarrollo, corrección y futuro mantenimiento de la aplicación de una forma rápida y fácil.

La guía de uso

Es lo que comúnmente llamamos el manual del usuario. Contiene la información necesaria para que los usuarios utilicen correctamente la aplicación.

Este documento se hace desde la guía técnica pero se suprimen los tecnicismos y se presenta de forma que sea entendible para el usuario que no sea experto en informática.
Un punto a tener en cuenta en su creación es que no debe hacer referencia a ningún apartado de la guía técnica y en el caso de que se haga uso de algún tecnicismo debe ir acompañado de un glosario al final de la misma para su fácil comprensión.

La guía de instalación

Es la guía que contiene la información necesaria para implementar dicha aplicación. Dentro de este documento se encuentran las instrucciones para la puesta en marcha del sistema y las normas de utilización del mismo.
Dentro de las normas de utilización se incluyen también las normas de seguridad, tanto las físicas como las referentes al acceso a la información.

3.1.4 Video tutorial

El video tutorial del caso de estudio le permitirá obtener una mayor comprensión de cómo se trabaja con el marco de trabajo a través de una explicación guiada.

3.2 Libros

Tendrá acceso a descargar libros de interés que tratan de arquitectura. Entre los libros que aparecen a su disposición se encuentran:

1. Manual de instalación de las herramientas.
2. Php_manual.es.

3.3 Release

Fuente

Se facilita el código fuente a aquellos usuarios que los lideres del proyecto ERP-Cuba den autorización para fortalecer el trabajo colaborativo y en equipo. Así se fomentará la colaboración entre los proyectos de la universidad de las Ciencias Informáticas.

Versión

El marco de trabajo se encuentra en su primera versión. Integrado por 11 componentes. Con la cual se ha realizado la prueba piloto en 6 entidades del país. Presenta un instalador destinado para ser ejecutado sobre el sistema operativo Linux.

4. Foro

Foro debate para que de su opinión sobre temas referentes al marco de trabajo. Puede dar su opinión en los temas existentes o crear un nuevo tema de debate.

5. UCIPedia

Vínculo a la wikipedia de la universidad**.**

6. Salir

Salir de la sesión de usuario.

El menú secundario se encuentra en el lateral derecho del sitio. Esta cuenta con varias secciones, una de estas secciones solo podrá observarlas como usuario figura 11 y 12:

Invitado:

- ✓ Enlaces a otras páginas: desde donde podrá acceder a la intranet, el sitio primavera, el sitio de Software libre y al portal del SEI.
- ✓ Inicio de sesión: los usuarios iniciaran sesión como usuarios autenticados
- ✓ En línea: muestra la cantidad de usuarios que están conectados al sitio y el user de los autenticados.

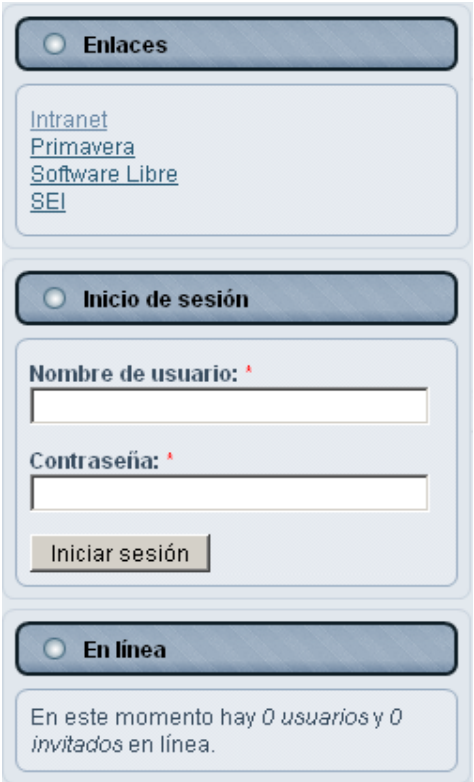


Figura 11: lateral derecho sin autenticarse

Usuario:

Podrá observar las mismas secciones que el invitado y la sección Grupo de eventos.

- Grupo de eventos: Tendrá información de los eventos a desarrollarse en el proyecto ERP-Cuba, conferencias, talleres, etc.
- Encuesta: Dara su opinión acerca del sitio.

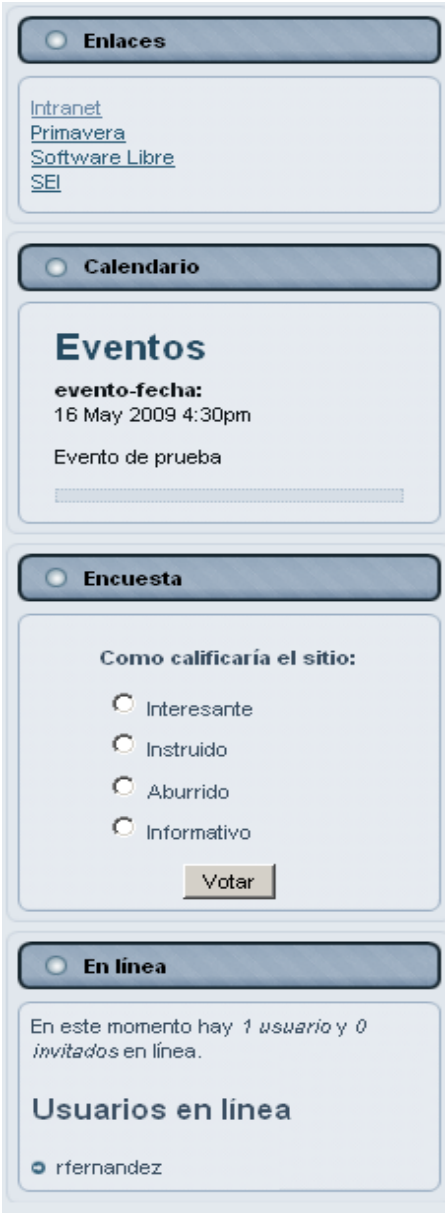


Figura 12: Lateral derecho autenticado

En la parte inferior del sitio se muestra el contenido de cada uno de los menús y submenús que se encuentran en el encabezado, en la figura 13 se muestra el contenido de la página de inicio.

Figura 13: Cuerpo del sitio

3.3.2 Funcional

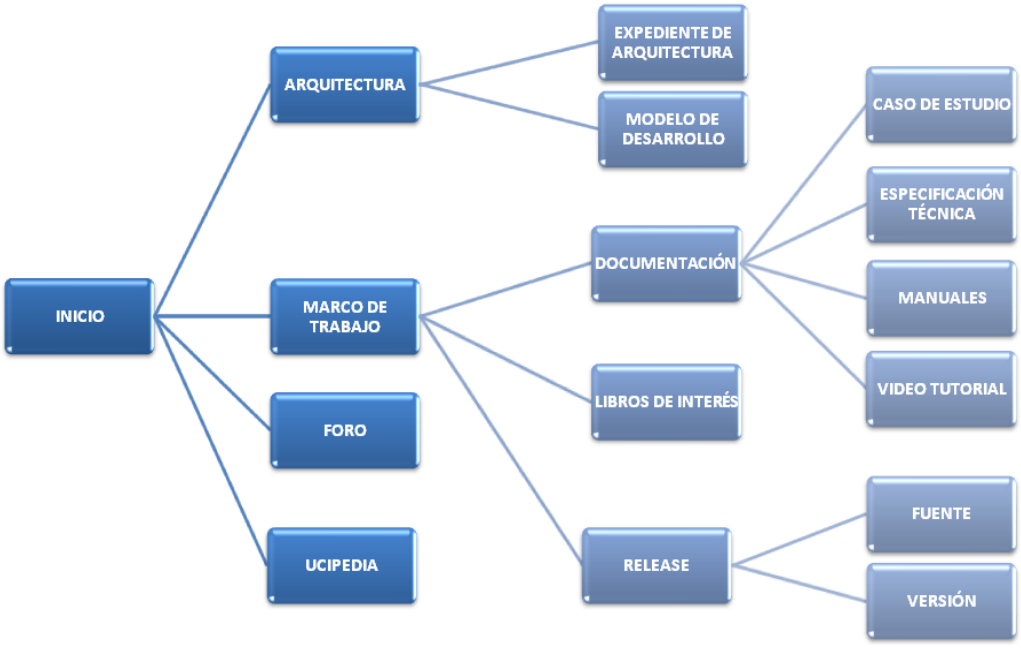


Figura 14: Mapa de navegación del sitio

3.4 Políticas de navegación

Para navegar por el sitio debe cumplir con las políticas de seguridad que se plantean a continuación.

Como invitado

7. Acceso a ver la información.
8. Acceso a encuesta.

Como usuario autenticado

1. Acceso a ver la información
2. Acceso a encuesta.
3. Puede realizar una solicitud para poder descargar la documentación del marco de trabajo.
4. Acceso al calendario de eventos.
5. Acceso al foro.

Como administrador

1. Acceso a ver toda la información.

2. Acceso a modificar toda la información.
3. Acceso a insertar información.
4. Acceso a editar la información.
5. Acceso a eliminar la información.

3.5 Conclusiones

El portal para la disposición de la documentación técnica del marco de trabajo del sistema Cedrux permitirá compartir los conocimientos e innovaciones tecnologías realizadas en dicho proyecto, con el resto de los proyectos que se dedican a implementar soluciones de la índole del sistema de gestión Cedrux, incrementando a su vez el trabajo colaborativo de la universidad. Aporta nuevos conocimientos para la comunidad de PHP de la universidad. Sirve de apoyo para el desarrollo en las líneas de desarrollo del proyecto ERP - Cuba.

Conclusiones generales

El actual marco de trabajo del proyecto Cedrux, fusionado bajo tecnología totalmente libre (PHP, postgresql, apache). Posee el desarrollo de tecnologías propias basadas en otros frameworks como ZendFramework, Doctrine, y que unifica temas que actualmente están en desarrollo y que son totalmente novedosos en las aplicaciones de gestión similares. Por otra parte este marco tecnológico soporta actualmente una aplicación de mas de 11 componentes (los cuales responden a más de 1000 requisitos de negocio),garantizando las transacciones de negocio, la integración de los componentes de negocio, la exportación e importación de información, el tratamiento de las excepciones a nivel de negocio, las validaciones de negocio, la trazabilidad de las operaciones, el uso de los aspectos generales la colaboración entre subsistemas y la integración con una aplicación externa (Reporteador dinámico). En fin no da dado problemas hasta ahora en las 6 entidades en que esta desplegado el producto nacional Cedrux (ERP cubano).

Resumiendo, dicho marco tiene una gran importancia a nivel de país, siendo una tecnología novedosa que surge para garantizar una de las tareas más Importantes que tiene nuestro país y que va dirigido al control y planificación de los recursos, enmarcado sobra la soberanía tecnología que estamos buscando, el mismo perfeccionará y cambiará las dimensiones actuales de desarrollo de software de gestión en tecnologías libres.

La realización de este trabajo de diploma facilitó el cumplimiento de los objetivos trazados para la solución de la problemática planteada. Se adquirieron conocimientos a partir de un análisis del estado del arte sobre la documentación técnica de los marcos de trabajo, la utilización del trabajo colaborativo como medio de socialización de la información y el conocimiento, así como las herramientas para la puesta en práctica del mismo.

Se realizó un análisis de la disponibilidad de la documentación técnica del marco de trabajo para las líneas de desarrollo del proyecto ERP-Cuba, estructurando la documentación en el repositorio central. Se realizó un estudio del funcionamiento de los componentes que conforman el marco de trabajo del sistema de gestión Cedrux, definiendo un conjunto de aspectos para la documentación técnica de los que conforman la primera versión del marco de trabajo.

Se obtuvo el manual de instalación del marco de trabajo para la utilización por parte de los interesados en implementar soluciones integrales sobre el marco en cuestión.

Con el objetivo de incrementar el trabajo colaborativo en la Universidad de las Ciencias Informáticas y socializar las innovaciones tecnológicas realizadas en el lenguaje PHP durante el desarrollo del sistema, se construyó un portal para la disposición de la documentación técnica del marco de trabajo del sistema de gestión Cedrux.

Recomendaciones

Se recomienda para la validación y utilización futura de este trabajo:

- ✓ Agregar y actualizar toda la documentación técnica del marco de trabajo en el repositorio.
- ✓ Poner a disposición, el portal para la disposición de la documentación técnica del marco de trabajo para la comunidad de desarrollo de PHP, y los interesados en reutilizar el marco de cuestión.
- ✓ Empezar la documentación de los componentes que se forman parte de la segunda versión del marco de trabajo.
- ✓ Realizar un módulo para la solicitud de reutilización del marco de trabajo.

Bibliografía

ABCdatos [En línea]. - http://www.abcdatos.com/webmasters/programa/z5030.html.

aferve.com [En línea]. - 8 de Febrero de 2009. - 15 de Abril de 2009. - http://www.aferve.com/noticias/8-proyectos-opensource/2055-ezpublish-402.

Altamar Johanna Recursos para webmaster [En línea] // Webnova. - 13 de Abri de 2009. - http://www.webnova.com.ar/articulo.php?recurso=509.

Altamiranda M.Sc. Junior PLATAFORMA GFORGE [En línea]. - 20 de Abril de 2009. - http://sistemas.fsl.fundacite-merida.gob.ve/docman/view.php/16/116/gforge.pdf.

Alvarez Sara desarrolloweb.com [En línea]. - 6 de Julio de 2006. - 20 de Diciembre de 2008. - http://www.desarrolloweb.com/articulos/importancia-documentacion.html.

Buetas Carlos pirineos [En línea]. - 25 de Enero de 2009. - 5 de Febrero de 2009. - http://www.infopirineo.com/blog/cms-4-comparativa-de-cms-drupal-joomla-ezpublish-wordpress-xoops-typo3.

eFaber Efaber soluciones inteligentes [En línea]. - 1 de Marzo de 2009. - 9 de Abril de 2009. - http://www.efaber.net/empresa.

Franco Miguel Aula 2.1 [En línea]. - 10 de enero de 2008. - 18 de Marzo de 2009. - http://www.aula21.es/aula/spip.php?article6.

Fundación Gabriel Piedrahita Uribe Eduteka Fundación Gabriel Piedrahita Uribe [En línea]. - proiectus, 1 de Mayo de 2009. - 10 de Mayo de 2009. - http://www.eduteka.org/quienes.php3.

Itcole Itcole [En línea] // Innovative Technologies for Collaborative Learning and Knowledge Building. - 2007. - 20 de Marzo de 2009. - http://bscl.fit.fraunhofer.de/.

Ledo María Vidal y Concepción Baez Carlos Mario Herramientas para el trabajo colaborativo o sistema de gestión de contenidos [En línea]. - 4 de Marzo de 2008. - 13 de Febrero de 2009. - http://bvs.sld.cu/revistas/ems/vol22_3_08/ems13308.htm.

Lidia Fuentes José M. Troya y Antonio Vallecillo Lección 1 Desarrollo de Software Basado en Componentes [Libro]. - Málaga, España : Dept. Lenguajes y Ciencias de la Computación. Universidad de Málaga..

Montejava [En línea]. - http://www.montejava.es/joomla.asp.

reinisch DOCUMENTACIÓN TÉCNICA [Libro]. - España : [s.n.].

TechneZero TechneZero [En línea]. - 5 de Febrero de 2009. - http://www.technezero.com/productos/cms.htm.

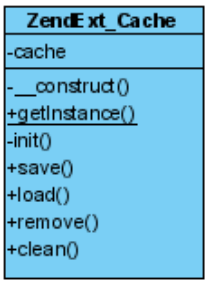
Wikipedia la enciclopedia libre [En línea]. - 18 de Marzo de 2009. - http://es.wikipedia.org/wiki/Trabajo_colaborativo.

Wikipedia La enciclopedia libre [En línea]. - 25 de Enero de 2009. - http://es.wikipedia.org/wiki/Groupware.

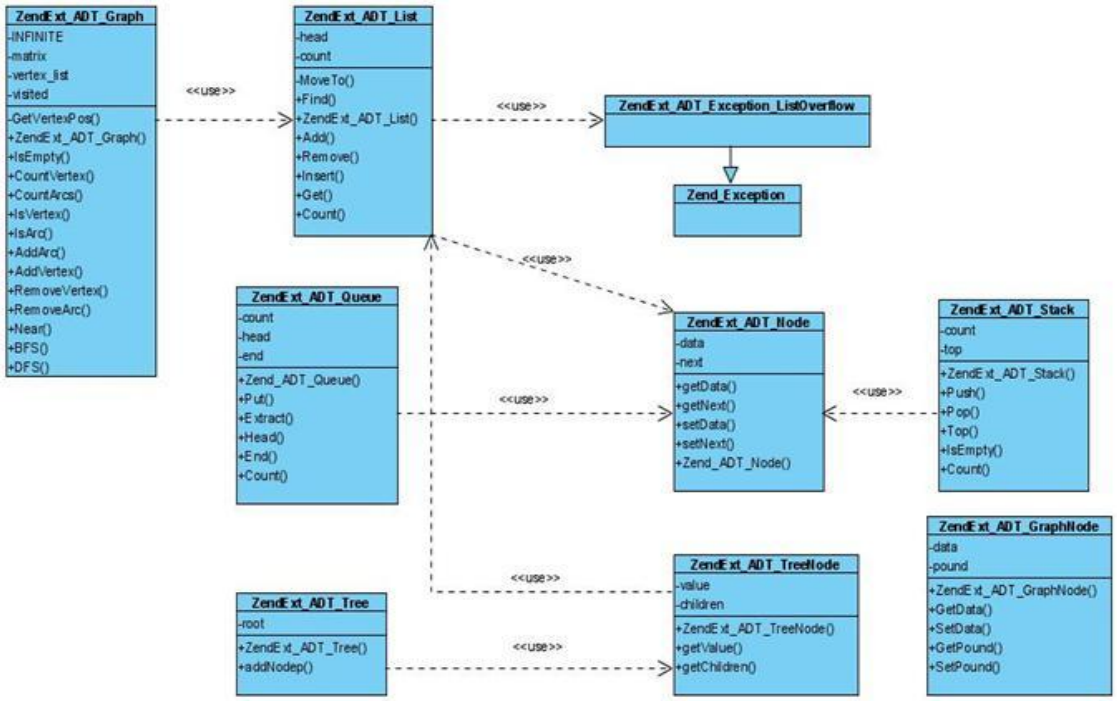
xsto.info [En línea]. - 2003-2009. - http://www.xsto.info/index.php/xs2info/productos/ez_publish/presentacion.

Anexos

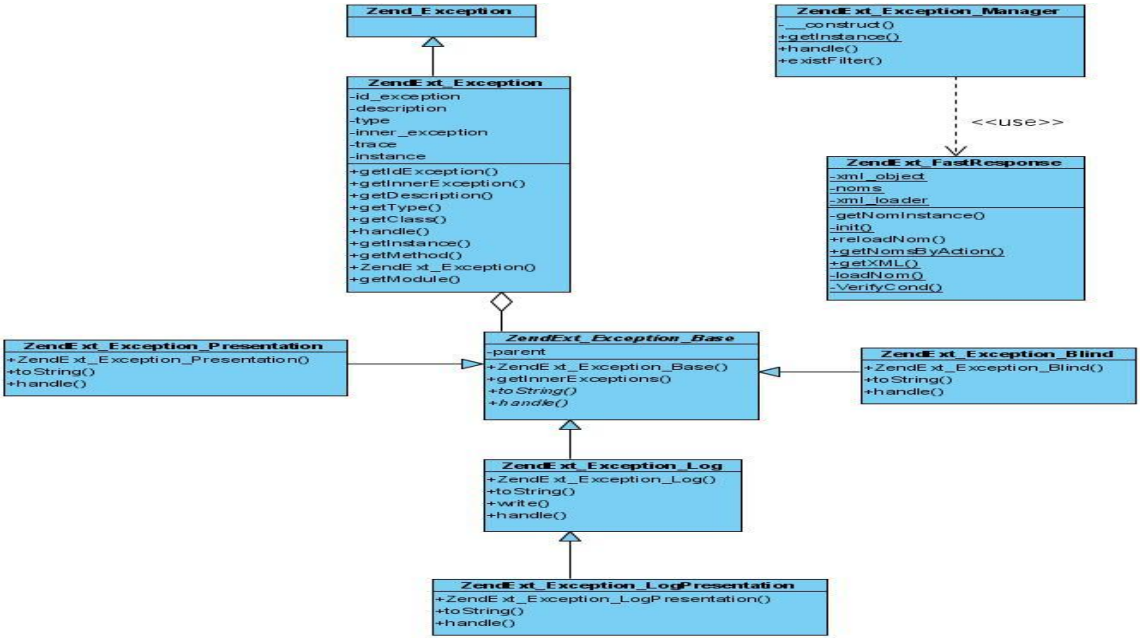
Anexo 1: Diagrama de clases componente ZendExt_Cache



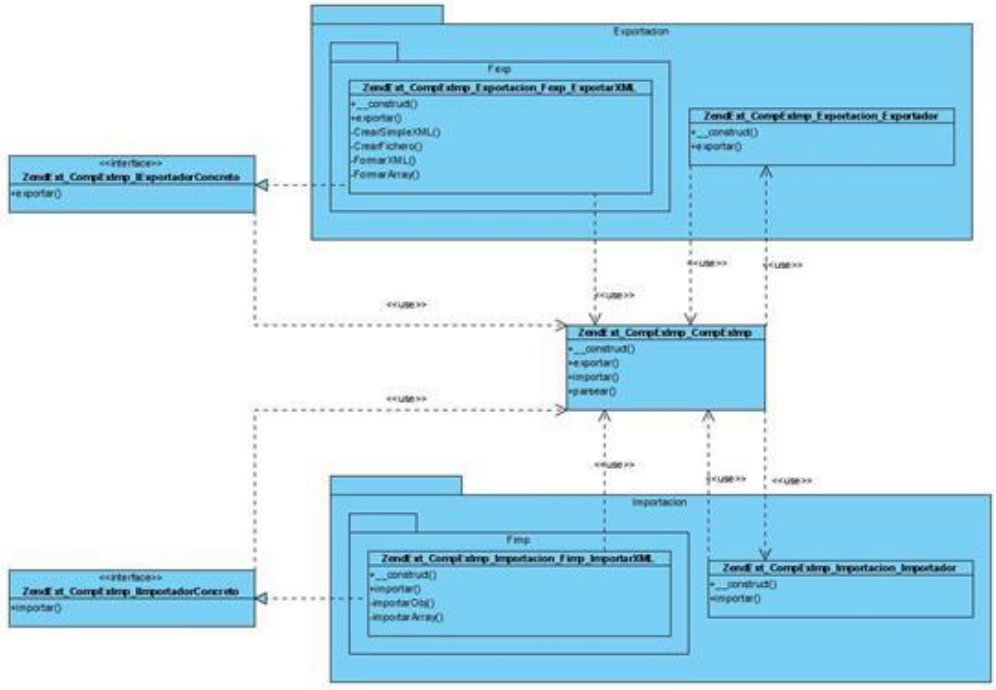
Anexo 2: Diagrama de clases componente ZendExt_ADT



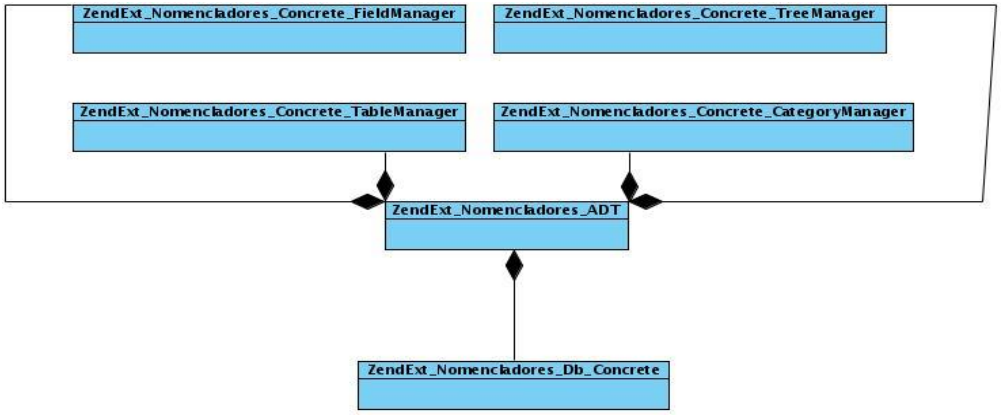
Anexo 3: Diagrama de clases componente ZendExt_Exception



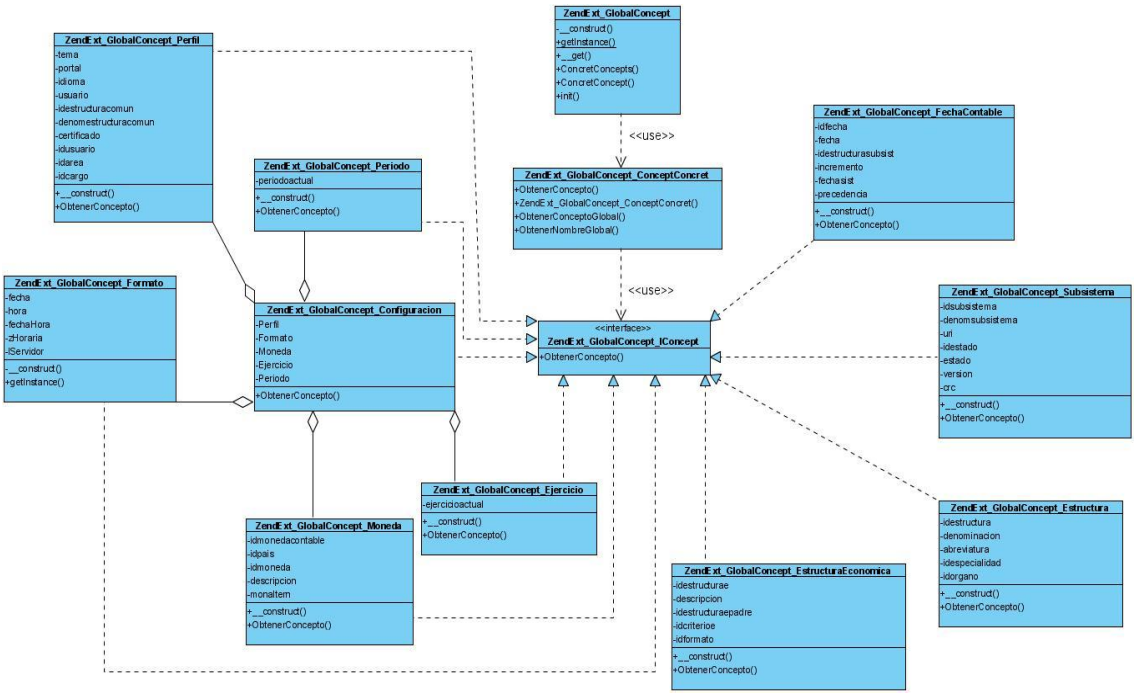
Anexo 4: Diagrama de clases componente ZendExt_ExpImp



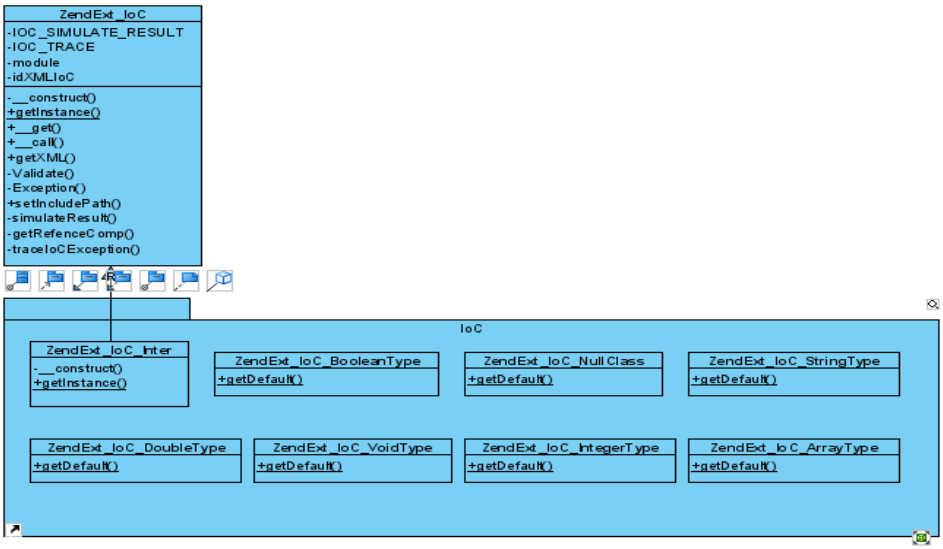
Anexo 5: Diagrama de clases componente ZendExt_Nom



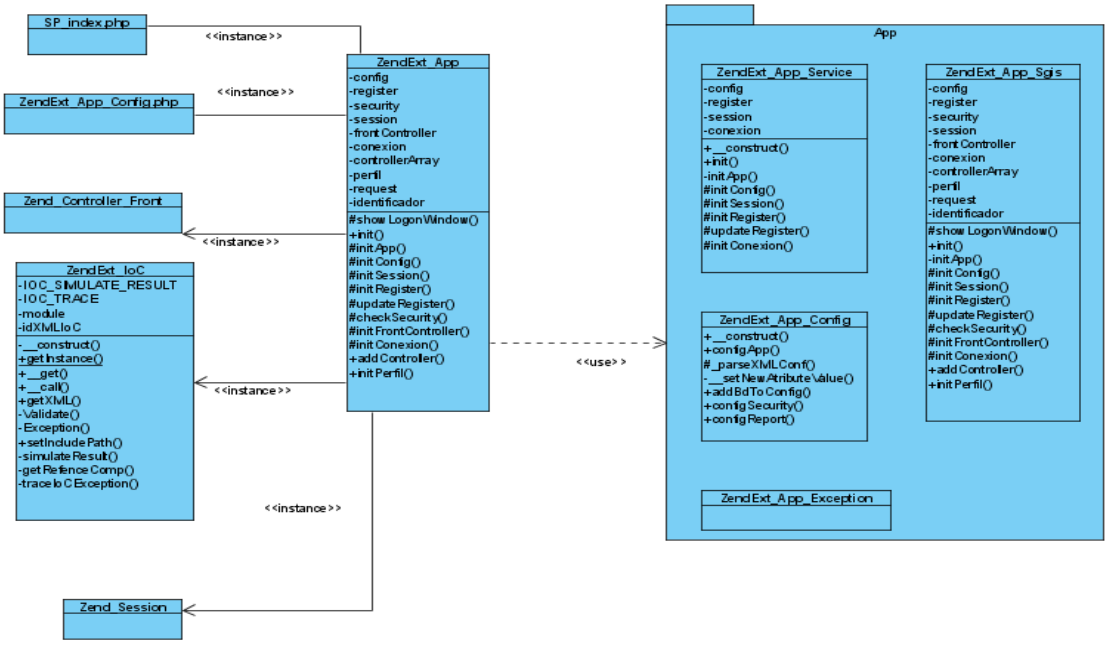
Anexo 6: Diagrama de clases componente ZendExt_GlobalConcept



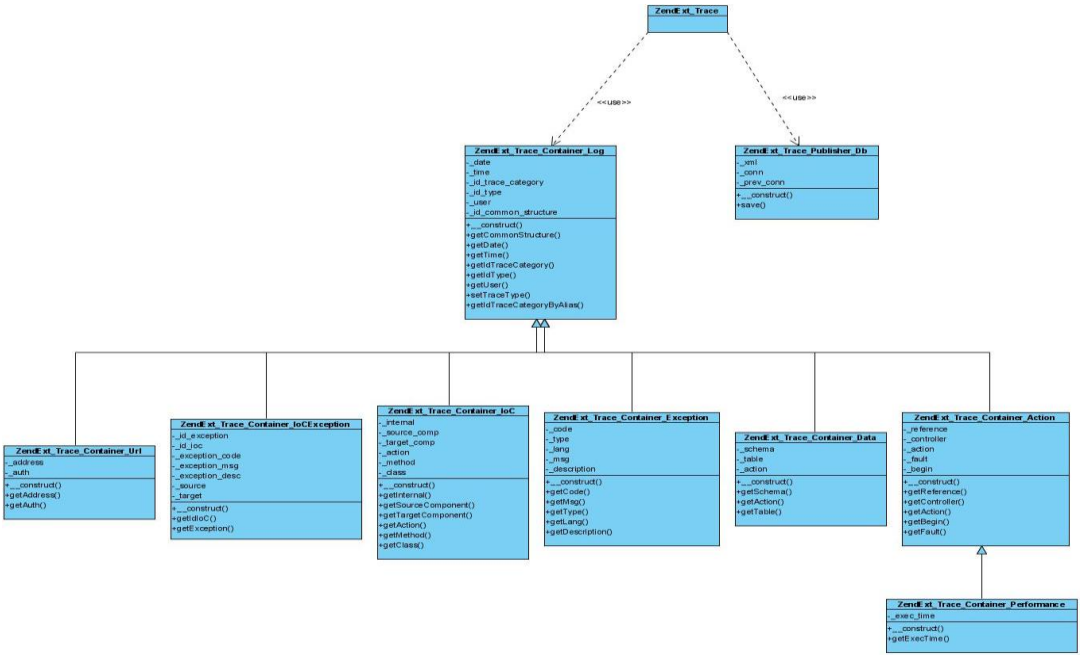
Anexo 7: Diagrama de clases componente ZendExt_IoC



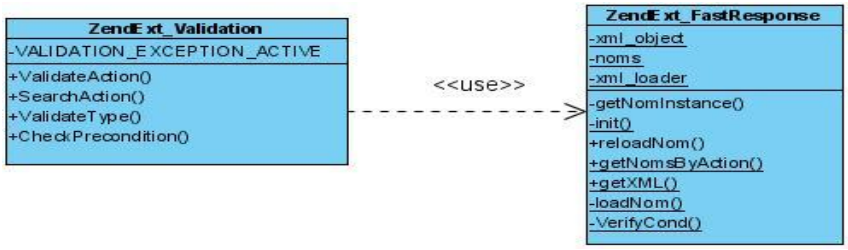
Anexo 8: Diagrama de clases componente ZendExt_App



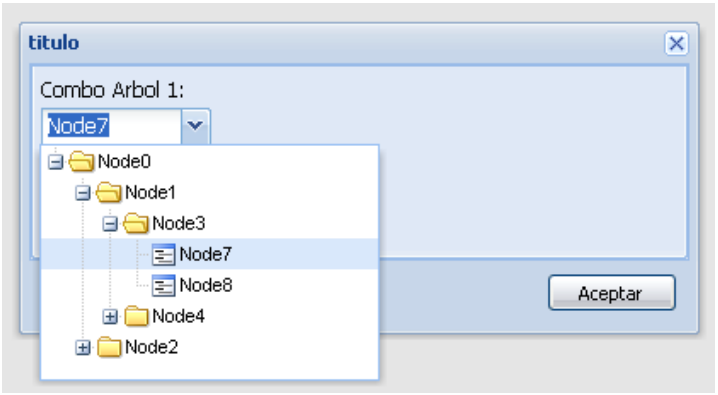
Anexo 9: Diagrama de clases componente ZendExt_Trace



Anexo 10: Diagrama de clases componente ZendExt_Validation



Anexo 11: Componente combo-iframe-árbol



Anexo 12: Componente ventana-iframe.