Universidad de las Ciencias Informáticas

Facultad 3

Título: Desarrollo del componente para la configuración visual de las excepciones en el marco de trabajo Sauxe.

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor: Inna Alfonso Alfonso.

Tutor(es): Ing. Pedro Manuel Nogales Cobas.

Ing. Javier Ruiz Durán.

Ciudad de la Habana, Junio de 2011.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al CEIGE de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Inna Alfonso Alfonso.

_____
Autor

Ing. Javier Ruiz Durán.

_____

Ing. Pedro Manuel Nogales Cobas.

_____

RESUMEN

En el Centro de Informatización para la Gestión de Entidades de la Universidad de las Ciencias Informáticas se está desarrollando el marco de trabajo Sauxe. La presente investigación constituye una propuesta para el desarrollo de un componente del Sauxe que permita la configuración visual de las excepciones. Actualmente en el marco de trabajo Sauxe no se está gestionando correctamente la información referente a las excepciones provocando que los desarrolladores tengan que realizar trabajos engorrosos y además se enfrenten a problemas de accesibilidad. El desarrollo de la propuesta está previamente respaldado por un estudio sobre los principales conceptos tratados en la investigación y un estudio detallado de los sistemas existentes relacionados con el tratamiento de las excepciones. Todo lo modelado en este trabajo constituye un punto de partida para la implementación de los requisitos definidos, siendo la realización del análisis y diseño del componente el primer paso. Con la implementación del sistema propuesto, se espera mejorar el control de las excepciones y brindar una aplicación capaz de agilizar el proceso de configuración visual de las mismas. Detrás de esta propuesta, el impacto radica en reducir el tiempo de desarrollo de software de las aplicaciones que se implementen sobre Sauxe, pues el componente facilitaría potencialmente la configuración de las excepciones.

Palabras claves: configuración visual, excepciones, marco de trabajo Sauxe.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En los últimos años, como consecuencia de que las tecnologías de la información y de la comunicación (TIC) se han convertido en la columna vertebral de la economía mundial, se ha puesto mayor atención a los desafíos que se impone el hombre para adaptarse a las necesidades tecnológicas del momento.
Con la evolución de las TIC, surgen herramientas que ofrecen la posibilidad de encontrar soluciones novedosas ante estos desafíos. Las aplicaciones dirigidas a la plataforma web no están excluidas de este impacto, por lo que reflejan los cambios y contradicciones de la sociedad. Sus potencialidades como el gran poder de difusión, alcance que poseen, así como la variedad de usuarios que las utilizan, hacen a este tipo de software una herramienta de alta capacidad y de uso en diversos campos; entre ellos la planificación económica y la gestión de datos.
Cuba se esfuerza en alcanzar un desarrollo informático de punta trazándose como principal misión la informatización y automatización de sus empresas. De esta ideología surge la Universidad de las Ciencias Informáticas (UCI), convirtiéndose en la entidad impulsora dentro del ámbito informático evolucionando la industria del software cubano.

Debido a la importancia de incrementar la eficiencia económica en cuanto a la planificación y el control de los recursos empresariales, se decidió crear en la UCI el Centro para la Informatización de la Gestión de Entidades (CEIGE); en el centro se desarrolla, entre otros productos, un marco de trabajo para el desarrollo de aplicaciones web de gestión orientado a componentes llamado Sauxe, que soporta el desarrollo de los sistemas y responde a las necesidades productivas del país.

Actualmente en el marco de trabajo Sauxe no se gestiona eficientemente la información referente a las excepciones. Éstas son esenciales para el correcto funcionamiento de las aplicaciones con el objetivo de impedir que realicen operaciones que no les están permitidas. El componente Excepciones de Sauxe cuenta con un fichero XML que proporciona los datos necesarios de cada excepción en todos los sistemas desarrollados sobre el marco de trabajo. La información se encuentra de forma poco organizada, pues es manejada por los desarrolladores que realizan los cambios sobre el fichero manualmente. Como consecuencia provoca un crecimiento del mismo, ya que no se controla la existencia de excepciones en desuso o la posibilidad de reinsertar alguna, siendo este un trabajo engorroso que genera pérdida de tiempo; en ocasiones no se puede llevar a cabo como requiere, ya que el desarrollador carece de la totalidad de permisos necesarios para realizar el trabajo.

A partir de la situación planteada, el **problema a resolver** queda expresado en la siguiente interrogante: ¿Cómo facilitar el proceso de configuración de las excepciones en el marco de trabajo Sauxe?
Teniendo así como **objeto de estudio**: los marcos de trabajo para aplicaciones web de gestión. Para dar solución al problema planteado, se define como **objetivo general** desarrollar una herramienta que permita la configuración visual de las excepciones en el marco de trabajo Sauxe.
Para lograr el objetivo general se trazan los siguientes **objetivos específicos**:

- Construir el marco teórico de la investigación.
- Realizar el análisis y diseño de la solución.
- Implementar la solución.

Definiendo como **campo de acción** las excepciones en el marco de trabajo Sauxe.
En la presente investigación se estable como **idea a defender:**
Si se desarrolla una herramienta que permita la configuración visual de las excepciones, se facilitaría el proceso de configuración de las excepciones en el marco de trabajo Sauxe.

Tareas de investigación:

- Descripción de los procesos de negocio a automatizar.
- Obtención de los requisitos de la herramienta.
- Definición de los escenarios arquitectónicos de la herramienta.
- Elaboración del diseño de la solución.
- Implementación de la solución.
- Validación de la solución mediante la aplicación en entornos reales.
- Documentación de la solución obtenida.

Posibles resultados:

- Documento de descripción de los procesos de negocio.
- Modelo conceptual.
- Documento de escenarios arquitectónicos.
- Documento de especificación y descripción de requisitos.
- Prototipo de interfaz de usuario.
- Herramienta para la configuración visual de las excepciones en el marco de trabajo Sauxe.

El presente trabajo de diploma quedará estructurado en tres capítulos:

Capítulo 1: Fundamentación teórica.

En este capítulo se describen los principales conceptos relacionados con el tema. Se realizará el estado del arte que incluye un estudio de software, tecnologías y herramientas utilizadas, analizando sus características, ventajas y desventajas.

Capítulo 2: Propuesta de solución.

El capítulo se centra en definir los temas de análisis y diseño de la solución, donde se plantean las características del sistema para su posterior implementación. Se examinarán cada uno de los procesos de negocio, requisitos funcionales y no funcionales y prototipos de interfaces.

Capítulo 3: Implementación y prueba.

El tercer capítulo realiza una valoración de la solución. Expone una explicación de cómo se lleva a cabo el proceso para la implementación del sistema. Se validará el diseño mediante métricas y la implementación mediante pruebas de software.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se abordan los conceptos fundamentales que son necesarios conocer para el mejor entendimiento de la temática y servirán de base para el desarrollo del trabajo. También incluye un estado del arte de las aplicaciones web de gestión, y de los marcos de trabajo más utilizados, así como sus características en el tratamiento de excepciones. Además se describen las tecnologías y herramientas utilizadas en la implementación del sistema propuesto.

1.1. Aplicaciones web de gestión

La evolución de las comunicaciones en la era digital, estuvo contrastada por el aumento de la interconexión entre estaciones de trabajo por todo el mundo, lo que hoy conocemos por internet. Con el surgimiento de la www (world wide web) aparecen las aplicaciones web, las cuales son programas que se implementan para funcionar a través de un navegador.

El área de la gestión de la información es el campo más amplio de la informática en la creación de sistemas que automatizan los procesos de una empresa. Las aplicaciones web de gestión permiten trabajar en un entorno consolidado y centralizado, con acceso seguro e inmediato a los datos y a las tareas de interés para la empresa. Por lo general estas aplicaciones están dedicadas a determinada necesidad de gestión, ya sea de la contabilidad, recursos humanos, ventas, logística, entre otros. Esta especificidad implica que la mayoría de las veces no se puedan utilizar nuevamente en otros entornos, ya que es difícil adaptarlos a nuevos procesos de gestión. Para solucionar este tipo de problema se hace uso de los marcos de trabajo que proveen la posibilidad de desarrollar soluciones genéricas.

1.2. Framework

Framework, del inglés “frame” que significa marco y “work” trabajo. Marco de trabajo es un término usado por la mayoría de los desarrolladores y que ha sido definido por los autores de la forma más explícita:

Un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas (1).

Es un conjunto de librerías y componentes de probada solvencia, junto con una documentación y metodología de uso, que permite diseñar, construir e implantar aplicaciones corporativas de forma más uniforme, rápida, y con mayor calidad (2).

De acuerdo con lo establecido por los autores consultados se puede concluir que un Framework en el desarrollo de software es una tecnología implementada tanto con objetivos específicos como generales que reúne un conjunto de características y funcionalidades, creadas para facilitar la implementación de las aplicaciones de forma organizada en la cual otro proyecto de software se encuentra trabajando. Define una filosofía de trabajo, proporciona una estructura definida la cual ayuda a crear sistemas con mayor rapidez, menos trabajo y más fáciles de mantener.

Existen muchos marcos de trabajo dedicados al desarrollo de aplicaciones web tales como:

Symfony, diseñado para perfeccionar el desarrollo de las aplicaciones web, separando la lógica de negocio, la lógica de servidor y la presentación. Facilita mecanismos y herramientas orientadas a reducir el tiempo de desarrollo de una aplicación web compleja. Automatiza las tareas más habituales, permitiendo al desarrollador dedicarse por entero a trabajar en las características únicas de cada proyecto.

Zend Framework, orientado a objetos, de desarrollo de aplicaciones web en PHP. Se basa en la simplicidad, es sumamente práctico que agiliza la creación de código. Está desarrollado de modo que cuenta con medios para realizar aplicaciones en modernas tecnologías web y servicios.

.NET Framework, provee un amplio conjunto de soluciones destinadas a necesidades generales de la programación de aplicaciones. Es el producto principal de Microsoft, y es utilizada por la mayoría de las aplicaciones creadas para el sistema operativo Windows.

Sauxe, contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor agilidad en el proceso de desarrollo. Está basado en Zend framework, utiliza en la capa de acceso a datos el Lenguaje de Consulta de Datos (DQL) que implementa Doctrine. Utiliza Ext JS en la capa de presentación, por la gran gama de componentes que se pueden reutilizar y para mostrarle al usuario una interfaz más amigable.

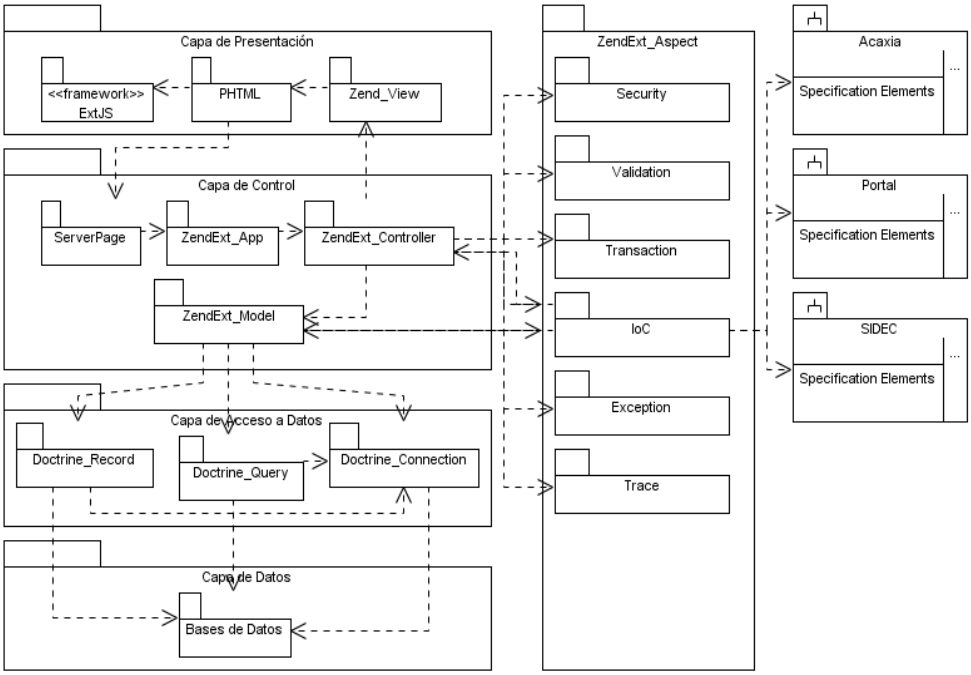


Figura 1: *Estructura del marco de trabajo Sauxe.*

1.3. Excepciones.

Una revisión de diferentes autores y sus enfoques sobre la definición del término excepción arrojó las siguientes descripciones como resultado:

Es un evento que ocurre durante la ejecución del programa que interrumpe el flujo normal de las sentencias (3).

Es cualquier situación de error o comportamiento inesperado que encuentra un programa en ejecución (4).

Los errores que se detectan en la ejecución se llaman excepciones. Están destinadas para la detección y corrección de errores. Si hay un error, la aplicación no debería morirse, se debería lanzar una excepción y resolver la situación del error. Utilizadas en forma adecuada, aumentan en gran medida la robustez de las aplicaciones (5).

Las definiciones expuestas convergen a una idea común, sobre la cual se apoyará la presente investigación.

La excepción es una situación anómala en la ejecución de un código o programa, situación que interrumpe su evolución normal y frena el cumplimiento del objetivo. Es usado el término de “excepción” por el hecho de que aunque un problema puede ocurrir lo hace con muy poca frecuencia, es decir, de manera excepcional. Generalmente se materializa en un objeto que contiene la información sobre el error, manifestando un mal funcionamiento.

La forma de configurar las excepciones varia entre los diferentes marcos de trabajo. Es importante la realización de esta configuración puesto que permite llevar un control detallado de las mismas de forma dinámica.

1.4. Estado del arte

En la actualidad existen soluciones informáticas que garantizan de manera cómoda y sencilla la configuración de las excepciones en el proceso de desarrollo de software. A continuación se hace referencia a los framework estudiados para el desarrollo de la solución, los cuales permitieron enfocar su desarrollo directamente al proceso de configuración visual de las excepciones; y se tuvo en cuenta también, como otros proyectos en la universidad han solucionado este problema.

1.4.1 Excepciones en Symfony

Al producirse una excepción en el framework Symfony, se muestra un mensaje de error que contiene información necesaria para descubrir el origen del problema que la generó. Dichos mensajes están descritos de forma clara y hacen referencia a la causa más probable del problema. La mayoría de las ocasiones ofrecen posibles soluciones para tratar los errores comunes. También muestra un enlace al sitio de Symfony para proveer más información sobre la excepción, la línea del código PHP en que se ha producido el error y la lista completa de los métodos que se han invocado, siendo posible seguir la traza de ejecución hasta la primera llamada que causó el problema (1).

Las excepciones de Symfony son de gran utilidad para depurar el funcionamiento de las aplicaciones. Los mensajes de notificación de excepciones ocurridas describen el evento útilmente pero carecen de la posibilidad de configurar esta información de manera cómoda para el desarrollador.

1.4.2 Microsoft Exception Management Application Block

Es un simple y extensible framework para guardar información de excepciones. Permitiendo anotar los detalles de la excepción a otras fuentes de datos o notificar a operadores sin afectar el código de la aplicación. Al usar el framework se puede reducir la cantidad de código escrito usualmente para el tratamiento de errores.

ESB[1] Exception Management

Los errores y excepciones pueden ocurrir en un rango de contextos y durante diferentes fases de proceso en un ESB. Existen muchas maneras de manejar excepciones en una solución de BizTalk[2], incluyendo el

1 Siglas de Enterprise Service Bus.
2 Es un servidor de administración de procesos de negocio.

uso de frameworks como el Enterprise Library. El ESB Exception Management sigue un patrón de diseño que proporciona un flexible acercamiento al monitoreo de excepción.

Los patrones consistentes para la gestión de excepciones son parte del centro de cualquier proyecto en desarrollo. Ayudan a maximizar el mantenimiento y hacen más fácil el soporte de la aplicación en el despliegue. ESB y las aplicaciones basadas en BizTalk introducen nuevos desafíos debido a la naturaleza distribuida de la infraestructura de BizTalk.

Debido a la complejidad de aplicaciones ESB, el desarrollo de una solución consistente para aplicaciones de gestión de excepciones debe incluir los objetivos comunes de diseño siguientes:

- Proporcionar un estandarizado acercamiento para la detección y tratamiento de excepciones que ocurran dentro del ambiente del servidor BizTalk.
- Proporcionar patrones comunes que permitan a los procesos automatizados reaccionar y manejar las excepciones de la aplicación.
- Proporcionar un patrón de gestión de excepciones débilmente acoplado que facilite el reúso.
- Proporcionar un mecanismo común de reporte para las excepciones de la aplicación y que sus disponibles estados de mensajes puedan ser aplicados a cualquier subsistema de BizTalk.

Portal ESB Management y Visor de Mensajes de Error

La Guía de ESB es un portal web que proporciona amplias características para la gestión y notificaciones de excepciones; útil para configuración de las excepciones y posee una interfaz para el registro. En la **Figura 2** se muestra la página de inicio del portal que proporciona una apreciación global del estado de las aplicaciones en ejecución.

***Figura 2**: Portal del ESB Management.*

Los usuarios pueden seleccionar un mensaje de error desplegado en el Portal del ESB Management para ver las propiedades ambientes y estáticas del error y una lista de los mensajes originales contenidos dentro del mensaje de error, **Figura 3**.

***Figura 3**: El ESB Fault Viewer mostrando la vista Detalles de Error.*

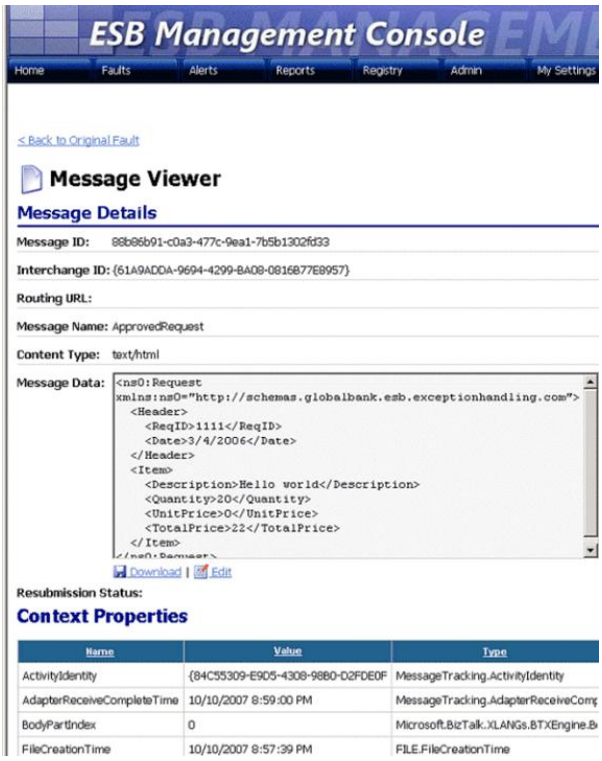


***Figura 4:** El ESB Message Viewer mostrando los detalles de error.*

1.4.3 Soluciones en la UCI

Durante la realización de la presente investigación fue necesario consultar las experiencias adquiridas en la UCI por otros proyectos. Se consultó con jefes de proyectos como ONE (Oficina Nacional de Estadísticas) el cual fue desarrollado sobre el framework .NET; y miembros de RN (Registro y Notarias) donde en el subsistema de Administración Financiera desarrollaron un intérprete de excepciones que independizaba el tratamiento y configuración de las excepciones, como el intérprete fue desarrollado en el framework .NET no fue aplicable a la solución por ser software propietario.

Una vez realizado el estudio sobre los principales marcos de trabajos existentes y teniendo en cuenta cada una de sus funcionalidades, se puede concluir que no están acorde con las necesidades de gestión de excepciones que se quiere satisfacer en el Sauxe. Algunos de estos sistemas están basados en la plataforma privativa .NET lo que resulta inviable para el desarrollo de las líneas de productos de la Dirección Tecnológica del CEIGE, debido al elevado coste que implicaría su utilización. Por tanto se decidió elaborar un componente que tenga en cuenta los elementos más importantes del diseño de Sauxe, de manera que erradique los problemas de desarrollo actuales.

1.5. Modelo de desarrollo orientado a componente.

La existencia de dos proyectos de desarrollo de software que sean iguales es casi imposible, cada uno tiene primacías, exigencias, y tecnologías muy diferentes. Sin embargo, en todos los proyectos, se debe reducir el riesgo, garantizar la predictibilidad de los resultados y entregar software en la calidad requerida a tiempo, esto se logra utilizando una metodología de desarrollo de software.

Existen metodologías tradicionales o pesadas y metodologías ágiles. Las primeras se centran fundamentalmente en controlar el proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. En el caso de las ágiles se basan en el factor humano y en el producto de software, dando mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas.

Teniendo en cuenta características especiales que presenta la UCI en cuanto a los proyectos productivos en el CEIGE se elaboró el Modelo de desarrollo orientado a componentes. Fue definido para que se tuvieran en cuenta las necesidades que presentaban cada línea, y los principales riesgos del proyecto. Está basado en buenas prácticas y principios de varias metodologías ya sean ágiles como XP y SCRUM o pesada como RUP, lo cual implica mejoría en aspectos importantes como la planeación, formación y satisfacción del equipo de trabajo.

El modelo de desarrollo orientado a componentes en general propone una solución sencilla y novedosa centrándose en el desarrollo de componentes como base tecnológica con una calidad superior en menor tiempo. Además propone dividir el trabajo y el equipo para lograr una mayor especialización de los desarrolladores. Se caracteriza por ser:

- **Centrado en la arquitectura**: La arquitectura determina la línea base, los elementos de software estructurales a partir de los elementos de la arquitectura de negocio. Interviene en la gestión de cambios y diseña la evolución e integración del producto. La arquitectura orienta las prioridades del desarrollo y resuelve las necesidades tecnológicas y de soporte para el desarrollo.
- **Orientado a componentes**: Las iteraciones son orientadas por el nivel de significancia arquitectónicas de los componentes, los mismos son abstracciones arquitectónicas de los procesos de negocio y requisitos asociados que modelan, el componente es la unidad de medición y ordenamiento de las iteraciones.
- **Iterativo e incremental**: Las iteraciones son planificadas y coordinadas con el equipo de arquitectura, los clientes y la alta gerencia. Cada iteración constituye el desarrollo de componentes, los cuales son integrados al término de la integración, permitiendo de esta manera la evolución incremental del producto.
- **Ágil y adaptable al cambio**: El desarrollo de las partes formaliza solamente las características principales de la solución, priorizando los talleres y las comunicaciones entre las personas. Los clientes y funcionales están involucrados en el proyecto y poseen parte de las responsabilidad del éxito del mismo. Los cambios son conciliados semanalmente, discutidos y aprobados.

El modelo concuerda con el propósito fundamental de una metodología especificando **quién** debe hacer **qué**, **cómo** y **cuándo** hacerlo. Define los diferentes roles involucrados y sus responsabilidades, las actividades que deben realizar y el flujo de las mismas, y los artefactos que deben ser generados; **figura 5**. La correcta aplicación del modelo proporciona en su medida la independencia tecnológica de los productos finales que es actualmente un objetivo fundamental de la UCI.

Figura 5: *Vínculo entre rol, artefacto y actividad en el proceso de desarrollo de software.*

El presente modelo de desarrollo de software cuenta con los siguientes roles (6):

- **Jefe de Línea de desarrollo**: Es la persona responsable de garantizar los cronogramas y compromisos de la línea y supervisar el proceso de desarrollo. Es quien organiza y controla el trabajo de los miembros de su línea, y controla los indicadores de eficiencia.
- **Planificador**: Es responsable de mantener actualizado el cronograma y la plantilla de Capital Humano. Es la persona que planifica y controla las tareas de los miembros del equipo, según las prioridades. También controla los planes de trabajo individuales, los horarios de trabajo y la distribución de los puestos de trabajo. Realiza las actas de las reuniones y talleres.
- **Arquitecto de sistema**: El arquitecto de sistema es quien garantiza que se cumplan las políticas y estándares definidos en la arquitectura. Toma las decisiones de integración en el proyecto y realiza la arquitectura del sistema, modera el taller de diseño.
- **Arquitecto de datos**: Es la persona encargada de construir y actualizar el Modelo de Datos, además responde por el manejo y recuperación de la información del mismo
- **Analista principal**: Es responsable de dirigir y organizar el trabajo del grupo de analistas de la Línea. Elabora el Mapa de Procesos de la Línea según los estándares. También participa en la definición y construcción de la Arquitectura de Negocio. Mantiene actualizados el seguimiento a los requerimientos de su línea de desarrollo.
- **Especialista de calidad**: Su responsabilidad es revisar, controlar las normas y estándares que establece el grupo de aseguramiento de la calidad incluyendo el proceso de desarrollo y guiar al grupo de auditoría y revisiones. Coordina el proceso de diseño de casos de prueba, coordina las pruebas de aceptación o liberación.
- **Especialista funcional**: Este rol participa en las sesiones de trabajo para identificar, describir y validar los procesos de negocio y los requisitos de software; validar desde el punto de vista funcional, los procesos de negocio y requisitos de software. Elabora Casos de Prueba según los estándares establecidos para ello.

- **Analista**: Su misión es participar en las sesiones de trabajo para identificar, describir y validar los procesos de negocio y los requisitos de software y Participar en el Taller de Diseño; elaborar la Descripción de Procesos de Negocio, Especificación de Requisitos y Casos de Prueba según los estándares establecidos para ello.
- **Desarrollador**: Diseña y construye los componentes de software de la línea.

Al realizar las actividades estos generan los artefactos siguientes (6):

- Jefe de línea de desarrollo: Plan de Iteración, Plan de Gestión de Riesgos, Plan de Trabajo individual de los integrantes de la Línea.
- Planificador: Plan de Iteración, Plan de Trabajo Individual de los profesionales, Definición del cronograma de desarrollo.
- Arquitecto de Sistema: Plan de Trabajo Individual, Diagrama de componentes, Prioridad de los componentes, Agrupación Requerimientos – Componentes, Informe de Integración.
- Arquitecto de Datos: Plan de Trabajo Individual, Modelo de Datos, Descripción del Modelo de Datos,.
- Analista Principal: Plan de Trabajo Individual, Mapa de Procesos de la Línea.
- Analista: Plan de Trabajo Individual, Modelo de procesos de negocio, Descripción de procesos de negocio, Modelo Conceptual, Prototipo de IU, Especificación de requisitos, Casos de Prueba.
- Especialista de calidad: Plan de Trabajo Individual, Plan de pruebas, Casos de Prueba, Registro de No Conformidades.
- Especialista Funcional: Plan de Trabajo Individual, Casos de Prueba, Validación de Procesos y Requisitos.
- Diseñador de LN: Diagrama de Clases, Descripción del Diseño de Clases.

1.6. Tecnologías.

A continuación se detallan las tecnologías que se definieron como política de desarrollo del centro (7), ya sean lenguajes de programación como los frameworks que componen el Sauxe.

1.6.1. Lenguajes de programación.

Los lenguajes de programación son tecnologías que nos permiten crear soluciones de software. Existen desarrollados innumerables lenguajes, que siguen el concepto de lograr la mayor abstracción posible, y facilitar el trabajo al desarrollador, aumentando la productividad.

PHP

PHP (Hypertext Pre-processor) es un lenguaje de programación usado mundialmente para la creación de sitios web dinámicos. Es una tecnología de código abierto que resulta muy útil para diseñar de forma

rápida y eficaz aplicaciones web dirigidas a bases de datos. La implementación principal de PHP es producida por The PHP Group. Publicado bajo la PHP License, considerada por la Free Software Foundation como licencia de software libre.
Es un lenguaje sencillo, de sintaxis cómoda. Su interpretación y ejecución la realiza el servidor en el cual se encuentra almacenada la página y el cliente sólo recibe el resultado de la ejecución. Dispone de una conexión nativa a los principales sistemas de base de datos utilizados actualmente tales como Postgres, MySQL, Oracle, Microsoft SQL Server, lo cual permite la creación de aplicaciones web muy robustas.

Características generales de PHP (8):

- No es un lenguaje de marcas.
- Presenta compatibilidad de interacción con los servidores de web más populares tales como Apache, XAMPP, FoxServ.
- Incorpora bibliotecas sumamente amplias de funciones integradas para realizar útiles tareas relacionadas con la web.
- Posee una amplia documentación, la cual se destaca por tener todas las funciones del sistema detalladamente explicadas y ejemplificadas en un único archivo de ayuda.
- Es un producto libre, de código abierto, lo que implica que una vez obtenido puede ser usado, copiado, estudiado, cambiado y redistribuido libremente.
- Es un lenguaje multiplataforma.
- Permite las técnicas de programación orientada a objetos.
- Capacidad de expandir su potencial utilizando gran cantidad de módulos.
- Posee tratamiento de errores.

Con el estudio de las características del lenguaje PHP podemos sintetizar algunas ventajas de su uso:
No requiere definición de tipos de variables. Las librerías facilitan en gran medida el desarrollo de las aplicaciones. Ser un producto libre y de código abierto representa una gran ventaja pues está soportado por una gran comunidad de desarrolladores que se encargan de encontrar y reparar los fallos de funcionamiento. PHP tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos ya sea Unix, Linux, Windows y Mac OS X.

Siendo PHP un lenguaje que provee al desarrollador de grandes ventajas, no está exento de desventajas potenciales que debemos tener en cuenta a la hora de utilizarlo en el desarrollo de aplicaciones:

- No posee una abstracción de base de datos estándar, sino bibliotecas especializadas.
- Por sus características promueve la creación de código desordenado y complejo de mantener.

- Todo el trabajo lo realiza el servidor, por tanto puede ser más ineficiente a medida que aumenten las solicitudes.
- La orientación a objetos es aún muy deficiente para aplicaciones grandes.

JavaScript

Es un lenguaje script e interpretado, es decir, que no requiere compilación, con una sintaxis muy parecida a la del lenguaje Java y el lenguaje C pero más sencillo de utilizar. Este es un lenguaje Case Sensitive[3] y al contrario que Java, no es exactamente un lenguaje orientado a objetos, puesto que no dispone de herencia, es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base que serían los prototipos y extendiendo su funcionalidad. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM[4]. En JavaScript se puede capturar los eventos dentro de las páginas para ejecutar alguna acción así como ser responsable de controlar algunas validaciones a nivel de cliente.

JavaScript es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Se utiliza mayormente para incorporar dinamismo, mejorar el aspecto y la funcionalidad de una página web e interactuar con los usuarios. JavaScript es soportado por la mayoría de los navegadores como Internet Explorer, Mozilla Firefox, Netscape, Opera (9).

CSS[5]

Es un lenguaje estándar creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. Se utiliza el lenguaje CSS para definir de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página. CSS es la mejor forma de separar los contenidos de su presentación y es imprescindible para crear páginas web complejas. Separar la definición de los contenidos y la definición de su aspecto, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento dando un mayor control y permite visualizar el mismo en infinidad de dispositivos diferentes, convirtiendo al HTML en un documento muy versátil y liviano (10).

Ventajas al utilizar CSS:

- Separación del contenido y presentación: permite que el desarrollador y diseñador puedan realizar sus tareas de forma independiente aunque paralela sin riesgo de interferencia en el trabajo.

3 Sensible a la diferencia entre minúsculas y mayúsculas.
4 Modelo de Objetos de Documento, DOM siglas en inglés.
5 Hojas de estilo cascada, CSS siglas en inglés.

- Flexibilidad: permite variar la apariencia con facilidad sin variar el contenido.
- Optimización de los tiempos de carga y de tráfico en el servidor: al utilizar CSS obtenemos archivos más ligeros; se reduce notablemente el tiempo de carga en el navegador.
- Precisión o elasticidad.
- Accesibilidad y estructuración.
- Limpieza del código fuente.
- Compatibilidad y continuidad.
- Estandarización frente a especificaciones propietarias: El uso del estándar CSS de la W3C (World Wide Web Consortium) evitará visualizaciones incorrectas de nuestras páginas en distintos navegadores.

XML

El Extensible Markup Language[6] es un simple formato texto-basado para representar la información estructurada: documentos, datos, configuración, libros, transacciones, facturas, y mucho más. Se derivó de un formato estándar más antiguo llamado SGML (ISO 8879) permitiendo ser más conveniente para la web (11).

El estándar permite definir la gramática de lenguajes específicos, por lo que el XML no es solo un lenguaje en particular, sino también un metalenguaje cuya particularidad más importante es que no posee etiquetas prefijadas con anterioridad, permitiendo describir otros lenguajes de marcado y definir lenguajes de presentación propios en dependencia del contenido del documento. El marcado extensible se basa en dos postulados:

- El marcado debe describir la estructura de un documento y otros atributos, en lugar de especificar el proceso que se realizará en ella, como marcado descriptivo necesita hacerse sólo una vez, y será suficiente para el procesamiento futuro.
- El incremento debe ser cuidadoso para que las técnicas disponibles en el procesamiento de objetos rigurosamente definidos como los programas y bases de datos puedan ser utilizados para el procesamiento de documentos XML.

El XML tiene como objetivos de diseño destacar la simplicidad, generalidad y usabilidad del estándar. Aunque el diseño de XML se centra en los documentos, es ampliamente utilizado para la representación de las estructuras de datos arbitrarios, por ejemplo en los servicios web.XML no ha nacido para su aplicación en la web únicamente, sino que se propone como un estándar para el intercambio de

6 Lenguaje de Marcado Extensible, XML siglas en inglés.

información estructurada entre diferentes plataformas. XML se desarrolló para proporcionar una flexibilidad y consistencia que no se podía alcanzar con HTML.

1.6.2. Marcos de trabajo y librerías

Zend Framework

Es un marco de trabajo de alta calidad para el desarrollo de aplicaciones y servicios web, es de código abierto y está diseñado para PHP.

Zend es un framework basado en la arquitectura MVC (Modelo-Vista-Controlador). Brinda soluciones para construir sitios web modernos, robustos y seguros. Posee un bajo acoplamiento entre sus componentes, lo que posibilita la utilización de los mismos a conveniencia, aunque todos estos en conjunto conforman un potente y extensible framework. Cuenta con clientes para el acceso a WS (12).

Presenta también las siguientes características (13):

- Módulos para manejar archivos en formato de documento portátil PDF, canales de sindicación de noticias RSS, y presenta el módulo Zend_Service para servicios web que soporta APIs de Amazon, Flickr, Yahoo.
- Módulos para la autenticación que poseen robustas clases, filtrado y validación de entradas de datos.
- Proporciona un sistema de caché dividido en front-end y back-end, de forma que se puedan almacenar en caché diferentes datos como resultados de funciones, páginas completas, y que esta información se almacene en archivos, en memoria, en base de datos, etc.
- Incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para consultar su base de datos, sin tener que escribir ninguna consulta SQL.
- Provee una solución para el acceso a base de datos que balancea el ORM con eficiencia y simplicidad.
- Proporciona una capa de acceso a base de datos, construida sobre PDO pero ampliándola con diferentes características, haciendo extremadamente simple la interacción con estas, sin necesidad de escribir ninguna consulta SQL.
- Permite convertir estructuras de datos PHP a JSON[7] y viceversa.
- Zend Framework posibilita la integración con PhPUnit para el desarrollo de pruebas unitarias a la aplicación que se desarrolle con el mismo.
- Cuenta con una completa documentación y pruebas de alta calidad.

7 Es un formato ligero para el intercambio de datos, JSON siglas en inglés.

El empleo de Zend Framework y su extensión Zend Ext provee de ventajas pues cuenta con componentes que facilitan y acortan el proceso de implantación de aplicaciones Web, aprovechando a su vez las ventajas de los servicios web.

Doctrine

Doctrine Framework es un potente y completo sistema de Mapas de Relaciones de Objetos (Object Relational Mapper, ORM por sus siglas en inglés) para PHP 5.2 o superior que incorpora una Database Abstraction Layer (capa de abstracción a base de datos). Brinda la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir, convertir clases creadas siguiendo las pautas del ORM, a tablas de una base de datos (14). Posee la habilidad de escribir opcionalmente las preguntas de la base de datos utilizando la programación orientada a objetos debido a que doctrine utiliza el patrón Active Record para manejar la base de datos. Esto les brinda una alternativa poderosa a diseñadores de SQL que mantiene un máximo de flexibilidad sin requerir la duplicación del código innecesario (14). Cuenta con su propio lenguaje DQL (Doctrine Query Language) a través del que permite crear el modelo en la sintaxis específica que propone Doctrine y luego generar toda la base de datos (15).

Ext JS

Ext JS es un framework de desarrollo de software que facilita las herramientas necesarias para la creación de aplicaciones web. El modelo del componente Ext JS mantiene bien estructurado el código incluso las aplicaciones más grandes pueden mantenerse fácilmente. Se puede extender los componentes predefinidos para satisfacer las necesidades de la solución, y las extensiones pueden ser encapsuladas justamente dentro de los componentes. Como resultado, el equipo de desarrollo puede desarrollar incluso aplicaciones más grandes sin consultar otros códigos. Proporciona una colección-enciclopédica de interface de usuario con un elegante tema de inicio. Tener en cuenta que Ext JS 2.2 tiene dos tipos de licencias, Lesser General Public License (LGPL) y la comercial, esta última es obligatoria si se desea obtener soporte.

Ext JS ofrece un rango extraordinario de interface de usuario. Posee una amplia librería con gran variedad de componentes web como ventanas, pestañas, formularios, paneles, barras de herramientas, menús, menús jerárquicos de árbol y más, semejante a aplicaciones de escritorio. Dentro de su librería contiene componentes para el manejo de datos, lectura de XML, lectura de datos JSON. Posee recursos de aprendizaje y una documentación API detallada, y regularmente mantenida.

Su diseño y arquitectura brindan las características esenciales de una librería completa para construir interfaces de usuario intuitivas, amigables y flexibles:

- Puede trabajar por medio de otras librerías JavaScript usando adaptadores.
- Sirve de puente entre las librerías JS más usadas (Prototype, JQuery, YUI[8]). Debido a que se inició como una extensión de YUI ésta presenta una cierta ventaja de compatibilidad respecto a las otras dos.
- Funcionalidades e interfaces controladas por eventos. Esto permite que en cada interacción del usuario con la aplicación, Ext JS intervenga en la presentación de la respuesta que da la aplicación al usuario, ya sea con intervención del servidor web o no.
- Brinda soporte para comunicar datos de forma asíncrona con el servidor.

Estas son solo unas razones por lo que Ext JS es la principal elección de los desarrolladores en el mundo.

Ya sea en Internet Explorer o el último navegador Cromo, las aplicaciones Ext JS tienen el mismo comportamiento, es decir, no importa el navegador donde la aplicación sea ejecutada. Ext JS framework soporta la mejoría de los navegadores incluyendo (16):

- Internet Explorer 6+.
- Safari 3+.
- Opera 9+.
- Mozilla Firefox 1.5+.

1.7. Herramientas

A continuación se hace una breve descripción de las herramientas que ayudarán al cumplimiento del objetivo general de este trabajo. Las mismas fueron definidas como política de desarrollo del centro (7), pues buscar otras herramientas conllevaría a un gasto innecesario de tiempo y de recursos.

Visual Paradigm

Es una herramienta CASE[9] ampliamente utilizada en el mundo del software que permite a los profesionales modelar sus diseños. Utiliza UML[10] como lenguaje de modelado y soporta el ciclo de vida completo del desarrollo de software: análisis y diseño, construcción, pruebas y despliegue. Permite el modelado colaborativo con Subversion y la integración de aplicaciones empresariales a las bases de datos. Es capaz de realizar ingeniería tanto directa como inversa, posibilita generar código para lenguajes como PHP. Posee un generador de informes automático para la documentación del proyecto en varios formatos como HTML, PDF. La herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto (17).

8 Interfaz de Usuario de Yahoo, YUI siglas en inglés.
9 Ingeniería de Softw are Asistida por Computadora, CASE siglas en inglés.
10 Lenguaje Unificado de Modelado, UML siglas en inglés.

PostgreSQL

Los servidores de bases de datos surgen producto de la necesidad de manejar grandes y complejos volúmenes de datos, al tiempo que requieren compartir la información con un conjunto de clientes de una manera segura. Los SGBD[11] proveen herramientas de administración completas que simplifican la tarea de la configuración, seguridad, creación y gestión de bases de datos. Estos sistemas deben proporcionar mecanismos de comunicación con otras plataformas que actúen también como clientes o servidores de datos.

PostgreSQL, inicialmente llamado Postgres, fue creado por el profesor Michael Stonebraker de la Universidad de Berkeley. Es un sistema de gestión de base de datos relacional orientada a objeto que marcó el inicio de las tecnologías objeto-relacional dentro de las bases de datos. Es distribuido bajo la licencia Berkeley Software Distribution (BSD). La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.

Propiedades (18):

- Atomicidad, asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.
- Consistencia, asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.
- Aislamiento, asegura que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generarán ningún tipo de error.
- Durabilidad, asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

PostgreSQL tiene soporte total para disparadores, vistas, procedimientos almacenados, almacenamiento de objetos de gran tamaño. Permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin que existan bloqueos. Posee un lenguaje procedural propio denominado PL/PgSQL. Contiene extensiones propias del estándar SQL para realizar consultas sobre la base de datos. Garantiza también las dependencias entre objetos, control para transacciones y concurrencia. Brinda soporte SQL/XML, soporte de autenticación, soporte total del Modelo Relacional de Bases de datos, soporte nativo para SSL, permite la exportación en formato XML e incluye nuevos tipos de datos: UUIDs, ENUMs y arreglos de tipos compuestos (19).

Ventajas:

11 Sistemas de gestión de bases de datos.

A través de más de quince años de desarrollo se le han ido incorporando un conjunto de mejoras, que lo convierten en un sistema de base de datos con una arquitectura probada, de sólida reputación por su integridad de los datos y exactitud. Se destaca en ejecutar consultas complejas y de gran tamaño. Facilita la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. PostgreSQL siempre tiene en cuenta una característica deseable en las bases de datos: la integridad referencial, gracias a la que se garantiza que una entidad o registro siempre se relacione con otras entidades que existen en la base de datos.

Como toda herramienta de software libre PostgreSQL tiene entre otras ventajas las de contar con una gran comunidad de desarrollo en Internet, su código fuente está disponible sin costo alguno. Fue diseñado para ambientes de alto volumen, por lo que escala fácilmente. Es ejecutado en la mayoría de los principales sistemas operativos: Linux, Unix, BSDs, Mac OS, Beos, Windows. Posee una documentación muy bien organizada, pública y libre, es altamente adaptable a las necesidades del cliente.

Apache

Es un software que permite crear en un ordenador, de una sencilla y rápida forma, un servidor de Protocolos de Transferencia de Hipertexto (HTTP), el cual se encarga de transferir los hipertextos, páginas web o páginas HTML, textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música. El Apache HTTP Server es un servidor robusto, de múltiples características y funcionalidades, una herramienta gratuita y de código abierto, lo que proporciona transparencia en todo el proceso de instalación.

Principales características con las que cuenta (20):

- Configuración basada en un poderoso archivo (`httpd.conf`).
- Soporte del protocolo HTTP y autenticación basada en la web.
- Soporte de host virtuales: Apache es uno de los primeros servidores web en soportar tanto host basados en IP como host virtuales.
- Integración de Perl.
- Soporte de scripts PHP: Apache ofrece un amplio soporte de PHP utilizando el módulo `mod_php`.
- Soporte de servlets de Java: Puede ejecutar servlets de Java utilizando el premiado entorno Tomcat con Apache.
- Servidor proxy integrado.
- Estado del servidor y adaptación de registros Apache le da una gran cantidad de flexibilidad en el registro y la monitorización del estado del servidor. El estado del servidor puede monitorizarse mediante un navegador web.

- Soporte de Server Side Includes (SSI), Secured Sockect Layer (SSL), Common Gateway Interface (CGI).
- Soporte de FastCGI.

Dadas las características se puede decir que un servidor Apache es altamente configurable y de diseño modular. Posibilita que los administradores de sitios web puedan elegir los módulos que serán incluidos y ejecutados en el servidor. Permite personalizar los mensajes de errores, la creación y gestión de logs[12], y la negociación de contenido, de este modo es posible tener un mayor control sobre lo que sucede en el servidor. Está diseñado para ser un servidor web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. La aplicación es prácticamente universal ya que permite ejecutarse en múltiples sistemas operativos como Windows, Novell NetWare y Mac OS X.

Subversion

Antes de existir las herramientas para controlar las versiones de su proyecto, era común que los desarrolladores pasaran trabajo con la gestión de los cambios en el código, de ahí que sea indispensable contar con una herramienta que facilite el control de versiones.

Subversion es una herramienta de control de versiones utilizada principalmente en los proyectos de desarrollo de software. La herramienta es libre y de código fuente abierto, se emplea para manejar los ficheros y directorios a través del tiempo haciendo posible la recuperación en caso de una perdida inminente, la recuperación se lleva a cabo mediante la salva existente en el repositorio[13] central. El repositorio es como un servidor de ficheros ordinario, excepto porque guarda todos los cambios hechos a sus ficheros y directorios. Subversion puede acceder al repositorio a través de la red, por lo que es usado por desarrolladores que se encuentran en distintas estaciones de trabajo.

Una característica importante de Subversión es que los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo.

Presenta también las siguientes características (21):

- Mantiene versiones no sólo de archivos, sino también de directorios
- Mantienen versiones de los metadatos asociados a los directorios.
- Además de los cambios en el contenido de los documentos, se mantiene la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre.

12 Es un registro oficial de eventos durante un rango de tiempo en particular.
13 Es un sitio centralizado donde se almacena y mantiene información, habitualmente bases de datos o archivos informáticos.

- Atomicidad de las actualizaciones, una lista de cambios constituye una única transacción o actualización del repositorio, esta característica minimiza el riesgo de que aparezcan inconsistencias entre distintas partes del repositorio.
- Soporte tanto de ficheros de texto como de binarios.
- Mejor uso del ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos.

Tener en cuenta que Subversion ayuda a medir la calidad y cantidad del trabajo realizado en una unidad de tiempo. Esta visibilidad instantánea nos permite observar la productividad del equipo de trabajo, así como los beneficios a escala administrativa para un líder de grupo.

Zend Studio

Zend Development Environment o Zend Studio, es un completo entorno de desarrollo integrado (IDE[14]) para el lenguaje de programación PHP. Fue diseñado para usarse con el lenguaje PHP, sin embargo también ofrece soporte básico para otros lenguajes web como HTML, JavaScript y XML. Está escrito en Java, y está disponible para las plataformas Microsoft Windows, Mac OS X y GNU/Linux.

Características generales que posee (22):

- Fácil instalación, soporte técnico, asociación de archivos, mecanismos de actualización y documentación.
- No requiere instalación previa de PHP ni del entorno de ejecución de Java.
- Instalación de barras de herramientas para navegadores como Internet Explorer o Mozilla Firefox de manera opcional.
- Integración del manual de PHP.
- Integración del Zend Framework: personalización del marco de proyecto, integración de herramientas Zend, generación de código del MVC, estándar de formato de código y código ayuda.
- Refactorización PHP; fácil renombramiento para archivos de clases, funciones y variables.
- Generación de código PHP: implementa funciones override, asistente para elementos PHP.
- Soporte JavaScript, ayuda para las librerías AJAX.
- Depuración PHP: integra depuración PHP/JavaScript.
- Administración de servicios web, comunicación SSL.
- Soporta HTML & CSS. Resaltado de sintaxis, completamiento de código, etiqueta de cierre automática, detección de errores de sintaxis en tiempo real.
- Soporta la ejecución de consultas SQL y navegación en bases de datos

14 Entorno de Desarrollo Integrado, IDE siglas en inglés.

- Sistemas remoto y virtuales: Permite la depuración en servidores remotos, VMware Workstation integration, SSH, FTP, SFTP.

Por lo que sus ventajas principales serian:

- Fácil integración de Java en su código utilizando las características del completado de Código y define/especifica Jars adicionales o carpetas de Clase que pueden utilizarse para el completado de códigos.
- Integración del uso y completado de código personalizado de Zend Framework y vista de la lista de las funciones del framework desde la Visualización de Funciones PHP.
- Visualización los eventos de Zend Platform en una ventana de lista de eventos personalizada y dedicada. Aumenta la productividad con: Soporte PHP 5 completo, Analizador de Código, carpeta de Código, completado de Código, coloreado de Sintaxis, Administrador de Proyecto, Editor de Código, Depurador de gráficos y asistentes.
- Documentación del código de forma más sencilla, aplicaciones, y proyectos con PHPDocumentor, la herramienta de documentación estándar para PHP.
- Simplifica el despliegue con la integración FTP y SFTP de forma tal que permita a los programadores en forma segura subir y descargar archivos de proyectos de modo transparente hacia y desde servidores remotos.

Mozilla Firefox

Mozilla Firefox es un producto desarrollado por la Corporación Mozilla. Se trata de un innovador y práctico navegador open source[15], que está en renovación constante. Se basa en el motor de renderizado Gecko, el cual se encarga de procesar el contenido de las páginas Web, desarrolladas en su mayor parte utilizando el lenguaje C++.

Algunas de las características más conocidas del Mozilla Firefox (23):

- Navegación privada: Navega por internet sin dejar una sola huella.
- Gestor de contraseñas: Recuerda las contraseñas sin ver ninguna ventana emergente.
- Barra de direcciones alucinante: Busca los sitios que favoritos en segundos (sin recordar URLs imposibles).
- Súper velocidad: Ver las páginas web mas rápido, usando menos memoria del ordenador.
- Antiphishing y Antimalware: Protección más avanzada contra los malware de la red. Firefox te protege de virus, gusanos, troyanos y spyware. Si de forma accidental entras en una web atacante, se te notificará que el sitio es sospechoso y el por qué no es seguro.

15 código abierto.

- Restaurar sesión: En caso de cierre inesperado vuelve a lo que hacías sin perder nada.
- Marcar páginas: Marca, busca y organiza los sitios web fácil y rápidamente.
- Personalización fácil: Posee muchos complementos dan la libertad de sentir un navegador hecho por el usuario.
- Pestañas: Mas funcionalidad con el uso de las pestañas a través del ratón.
- Personas: Cambia instantáneamente la apariencia muchos temas ligeros.

Mozilla Firefox por ser un práctico navegador es utilizado por los desarrolladores del centro CEIGE; se puede concluir mencionando algunas de las principales ventajas de su uso:
Las mejoras en el navegador en cuando a la velocidad del motor de JavaScript han dado como resultado un mejor rendimiento. Incorpora bloqueo de ventanas emergentes y posee compatibilidad con estándares abiertos.
La misión del proyecto Mozilla es preservar la elección y la innovación en Internet.

1.8 Conclusiones parciales

Se construyó el marco teórico de la investigación estudiándose los principales conceptos y definiciones que son esenciales para la investigación. La investigación que se realizó en el presente capítulo sobre cómo algunos marcos de trabajo realizan la configuración de las excepciones, arrojó las siguientes conclusiones:

- Zend framework provee los medios para el lanzamientos de las excepciones ocurridas en las aplicaciones desarrolladas sobre la plataforma, pero carece de una herramienta que permita gestionar la información sobre las excepciones.
- Las herramientas que permiten la gestión de la información sobre las excepciones no se encuentran disponibles para su utilización por ser software propietario, no ajustándose así a las necesidades del CEIGE.

Por lo tanto con el anterior análisis y valoración se propone la construcción de una herramienta que permita la configuración visual de las excepciones, la cual automatizará dicha configuración en todos los sistemas desarrollados sobre Sauxe.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

En el siguiente capítulo se parte de la fundamentación teórica de la solución y mediante la aplicación del modelo de desarrollo se comenzó la modelación del negocio describiendo los procesos de negocio. En este capítulo se describen las características que debe tener el sistema, a través de los requisitos funcionales y no funcionales. A partir de la especificación de los requisitos se elabora el modelo de diseño donde se describen las principales clases del diseño que serán utilizadas posteriormente en la implementación del componente.

2.1 Descripción de procesos de negocio

Un proceso de negocio es un conjunto de actividades en orden específico a lo largo del tiempo, con un principio y un fin, que permiten crear un producto o servicio final. El objetivo es guiar a los desarrolladores para la obtención de sistemas que se ajusten a las necesidades de los clientes. La identificación de los procesos de negocio es la actividad en la que se obtendrá una visión global de la entidad, mediante entrevistas y encuestas, con los desarrolladores. La descripción de estos tiene como objetivo entender el funcionamiento del componente que se desarrolla, comprender los problemas actuales e identificar las mejoras, es una vía natural para la determinación y captura de requerimientos.

2.1.1 Descripción del proceso de negocio: Adicionar excepción

La figura muestra el diagrama que identifica el proceso Adicionar Excepción que tiene como objetivo adicionar una nueva excepción a un subsistema ya existente. Partiendo de la necesidad que tiene el desarrollador de adicionar una nueva excepción, éste consulta el fichero XML de configuración de las excepciones buscando la ubicación para la introducción de los datos. Posteriormente guarda los cambios realizados sobre el fichero.

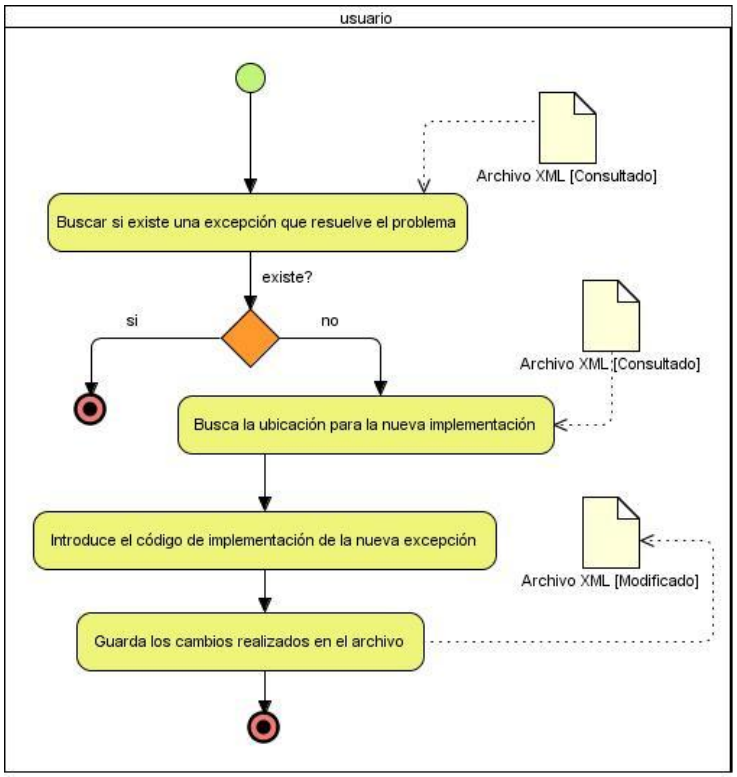


***Figura 6**: Diagrama del proceso Adicionar Excepción.*

2.1.2 Descripción del proceso de negocio: Modificar excepción

La figura muestra el diagrama que identifica el proceso Modificar Excepción que tiene como objetivo modificar una excepción ya existente en un subsistema. Partiendo de la necesidad del desarrollador de modificar los datos de una excepción en uso, éste consulta el fichero XML de configuración de las excepciones, busca la ubicación exacta donde se encuentran los datos de la excepción, introduce modificaciones en el fichero y guarda los cambios realizados.

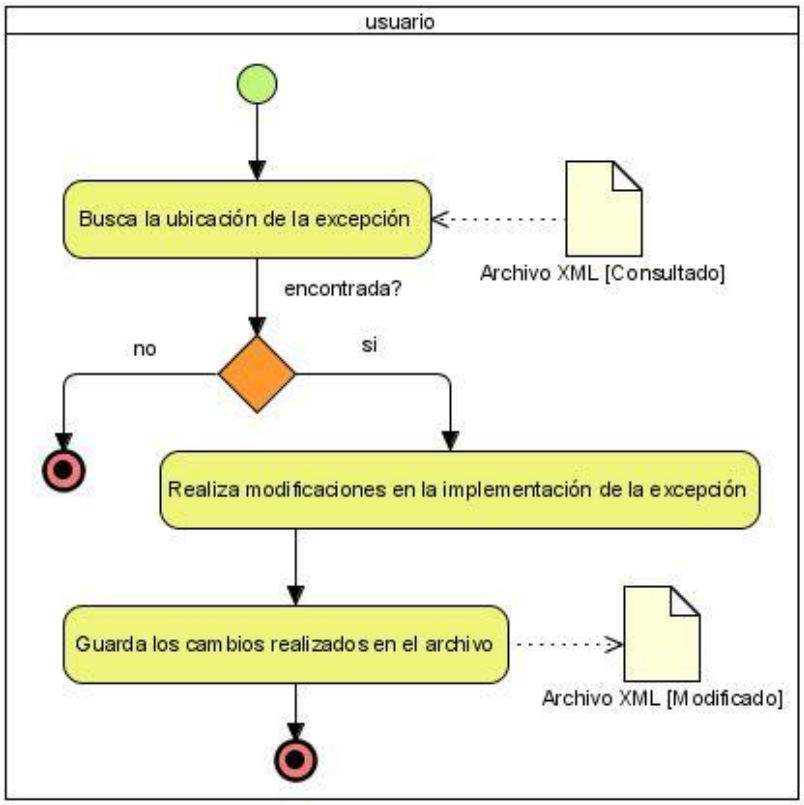


***Figura 7**: Diagrama del proceso Modificar Excepción.*

2.2 Especificación de los requisitos

En la ingeniería de software, un requerimiento es una necesidad documentada sobre la funcionalidad, el contenido o la forma de un servicio o producto. Es una condición o capacidad que debe ser conformada por el software en desarrollo. La especificación de requisitos es un proceso básico para garantizar la calidad del software y por lo general se realiza entre analistas e interesados en lenguaje natural.

2.2.1 Requisitos funcionales

Los requisitos funcionales especifican las capacidades o condiciones que la aplicación debe ser capaz de cumplir. Estos incluyen los procedimientos que los desarrolladores podrán ejecutar, así como las acciones ocultas que realizará la aplicación. Para la identificación de requisitos funcionales se tomó como punto de partida los procesos de negocio descritos anteriormente, además del análisis de las principales necesidades de los desarrolladores con el fin de determinar las capacidades o condiciones que debe satisfacer el componente.

Técnicas de elicitación de requisitos utilizadas

La elicitación o captura de requisitos es la actividad que se considera como el primer paso en un proceso de ingeniería de requisitos. Existen muchas técnicas disponibles para tal propósito, para obtener los requisitos funcionales del componente en cuestión, se utilizaron las siguientes técnicas:

Entrevista: es la más utilizada, y es prácticamente inevitable en cualquier desarrollo pues es una de las formas de comunicación más naturales entre personas; estructuradas por preguntar cómo y a quién, favoreciendo el contacto directo y la validación. Las entrevistas no se deben improvisar, por lo que se debe ir con ella redactada para que tenga éxito (24).

Tormenta de ideas: Reunión de varios interesados en la que todos expresan sus ideas sobre el problema y su solución. La forma de llevarla a cabo es que cada participante diga su idea sin ser interrumpido por otro. Al finalizar la sesión de lluvia de ideas se puede hacer una recolección de ideas sin duplicidad (24).

Requisitos funcionales identificados:
R1: Listar Excepción.
R2: Adicionar Excepción.
R3: Modificar Excepción.
R4: Eliminar Excepción.
R5: Probar Excepción.
R6: Buscar Excepción.

Las aplicaciones deben intentar anticiparse a las necesidades del usuario y no esperar a que el usuario tenga que buscar la información, recopilarla o invocar las herramientas que va a utilizar; por esta razón se tratan de diseñar interfaces de usuario que van mas allá de ser un simple vínculo entre el humano y la máquina.
Para el mejor entendimiento y desarrollo de la implementación del componente se diseñan las principales vistas tratando de satisfacer las exigencias del usuario final o desarrollador, buscando disminuir la complejidad. Estas vistas parten de la necesidad de creación del componente que en un principio no se ajustan a todos los requisitos que se proponen, es sólo durante su desarrollo que se alcanzan los

prototipos finales de los que va a disponer el sistema. Se elaboraron los prototipos de interfaz con la intención de simular las funcionalidades que necesita el componente. Cada una de ellas permitió obtener una idea de cómo quedará estructurada la solución.

Requisito funcional: Listar Excepción

Tabla 1: *Descripción del requisito funcional Listar Excepción.*

Conceptos tratados	**Conceptos**	**Atributos**
	Excepción.	Código, mensaje, descripción, nombre, tipo, identificador.
Precondiciones	**Precondiciones**	**Pre-requisito**
	Existencia de al menos un subsistema.	No procede.
Descripción	Para listar excepciones el desarrollador debe seleccionar el subsistema del cual desea visualizar las excepciones, a continuación el sistema muestra en la interfaz con la lista de las excepciones existentes.	
Validaciones	No procede.	
Post-condiciones	No procede.	
Post-requisito	No procede.	

Prototipo de interfaz de usuario: Listar Excepción

En la vista principal a la derecha se encuentra el árbol de subsistemas que posee el Sauxe como marco de trabajo, a los cuales se les va a gestionar las excepciones.

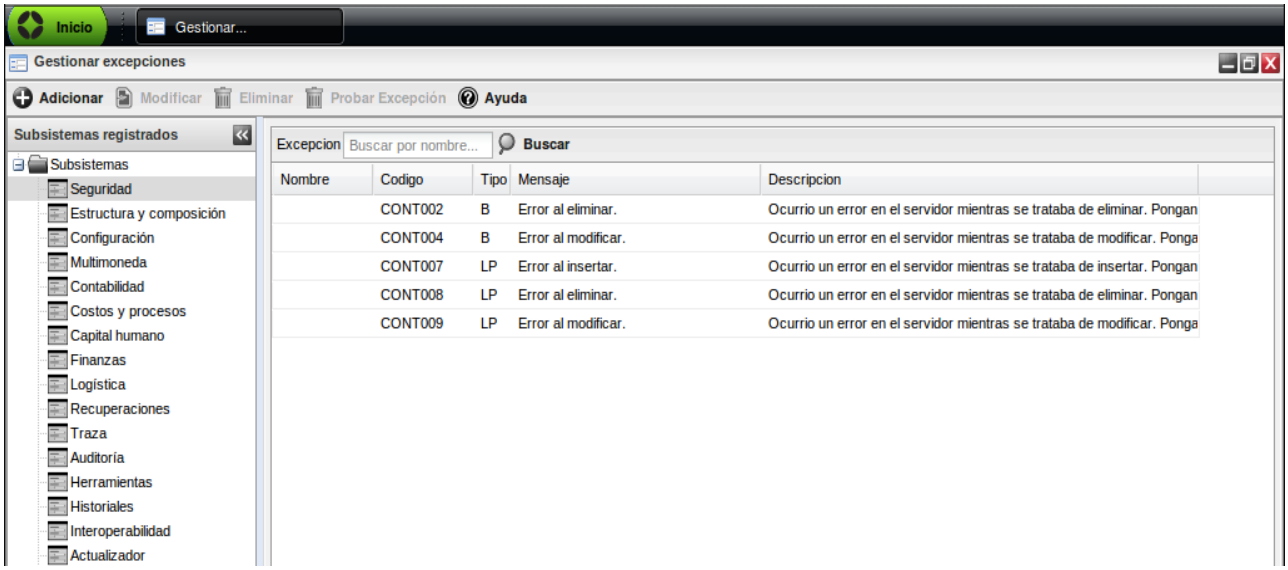


***Figura 8**: Prototipo de interfaz Listar Excepción.*

Requisito funcional: Adicionar Excepción

Tabla 2: *Descripción del requisito funcional Adicionar Excepción.*

Conceptos tratados	**Conceptos**	**Atributos**
	Excepción.	Código, mensaje, descripción, nombre, tipo, identificador.
Precondiciones	**Precondiciones**	**Pre-requisito**
	Existencia de al menos un subsistema.	No procede.
Descripción	Para adicionar una excepción el desarrollador debe especificar el subsistema al cual se adicionará la excepción y especificar el nombre de la excepción, el código que la identificara durante su empleo, el mensaje que se mostrara al lanzarse la excepción, debe especificarse el tipo de excepción y opcionalmente dar una descripción. El sistema deberá detectar posibles errores en los datos especificados por el desarrollador y notificar al mismo para su corrección. En caso de que no existan errores en los datos el sistema confirma que se adicionó correctamente la excepción y automáticamente muestra el listado de excepciones actualizado. El sistema debe permitir la cancelación de esta acción.	
Validaciones	No se permite la creación de excepciones con el mismo código de una ya existente. No se permiten valores nulos en los datos.	
Post-condiciones	El sistema actualiza la lista de excepciones.	
Post-requisito	Listar Excepción.	

Prototipo de interfaz de usuario: Adicionar Excepción

***Figura 9**: Prototipo de interfaz Adicionar Excepción.*

Requisito funcional: Modificar Excepción

***Tabla 3:** Descripción del requisito funcional Modificar Excepción.*

Conceptos tratados	**Conceptos**	**Atributos**
	Excepción.	Código, mensaje, descripción, nombre, tipo, identificador.
Precondiciones	**Precondiciones**	**Pre-requisito**
	Existencia de al menos un subsistema. Existencia de al menos una excepción.	Adicionar Excepción.
Descripción	Para modificar una excepción el desarrollador debe especificar el subsistema y la excepción que desea modificar. El sistema mostrará un formulario con los datos que posee actualmente la excepción, el desarrollador podrá modificarlos especificando el nombre de la excepción, el mensaje que se mostrará al	

	lanzarse la excepción, el tipo de excepción y opcionalmente dar una descripción. Este requisito no permitirá modificar el código de la excepción puesto que es un dato único. El sistema deberá detectar posibles errores en los datos entrados por el desarrollador y notificar al mismo para su corrección. En caso de que no existan errores en los datos el sistema confirma que se modificó correctamente la excepción y automáticamente muestra el listado de excepciones actualizado. El sistema debe permitir la cancelación de esta acción.
Validaciones	No se permiten valores nulos en los datos.
Post-condiciones	El sistema actualiza la lista de excepciones.
Post-requisito	Listar Excepción.

Prototipo de interfaz de usuario: Modificar Excepción

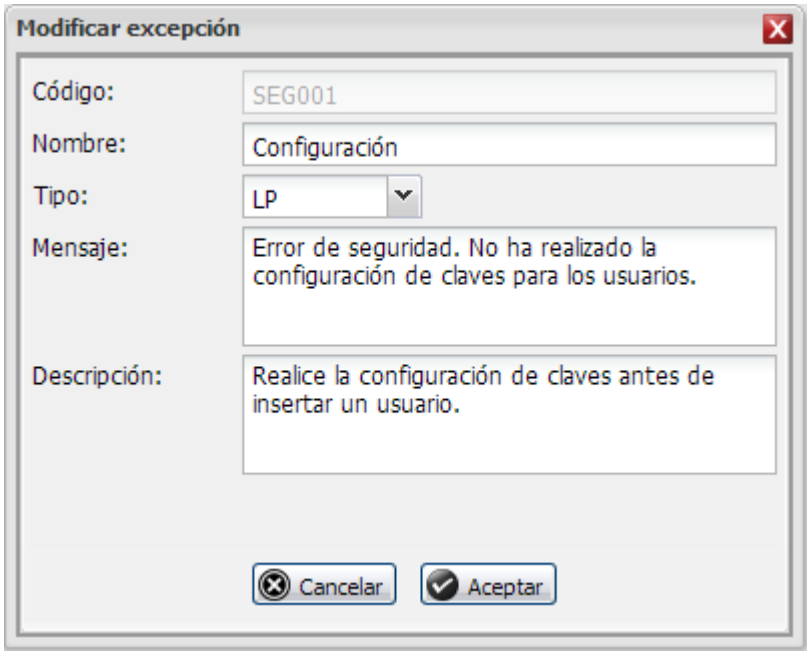


***Figura 10**: Prototipo de interfaz Modificar Excepción.*

Requisito funcional: Eliminar Excepción

Tabla 4: *Descripción del requisito funcional Eliminar Excepción.*

Conceptos tratados	**Conceptos**	**Atributos**
	Excepción.	Código, mensaje, descripción, nombre, tipo, identificador.
Precondiciones	**Precondiciones**	**Pre-requisito**
	Existencia de al menos un subsistema. Existencia de al menos una excepción.	Adicionar Excepción.
Descripción	Para eliminar una excepción el desarrollador debe especificar de los subsistemas y la excepción que desea eliminar. El sistema solicita la confirmación del desarrollador para realizar esta acción, en caso de ser confirmada la acción el sistema elimina la excepción especificada. El sistema debe permitir la cancelación de esta acción.	
Validaciones	No procede.	
Post-condiciones	El sistema actualiza la lista de excepciones.	
Post-requisito	Listar Excepción.	

Prototipo de interfaz de usuario: Eliminar Excepción

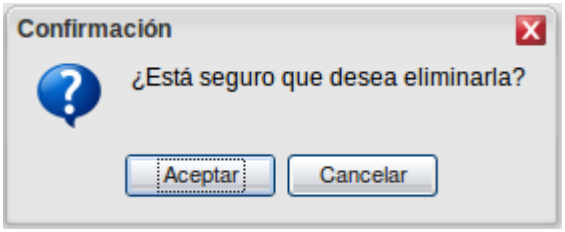


Figura 11*: Prototipo de interfaz Eliminar Excepción.*

Requisito funcional: Probar Excepción

***Tabla 5**: Descripción del requisito funcional Probar Excepción.*

Conceptos tratados	**Conceptos**	**Atributos**
	Excepción.	Código, mensaje, descripción, nombre, tipo, identificador.
Precondiciones	**Precondiciones**	**Pre-requisito**
	Existencia de al menos un subsistema. Existencia de al menos una excepción.	Adicionar Excepción.
Descripción	Para probar una excepción el desarrollador debe especificar de los subsistemas y la excepción que desea probar. El sistema anuncia la ejecución de la acción con un mensaje de alerta. El sistema ejecuta la prueba lanzando la excepción especificada.	
Validaciones	No procede.	
Post-condiciones	No procede.	
Post-requisito	No procede.	

Prototipo de interfaz de usuario: Probar Excepción

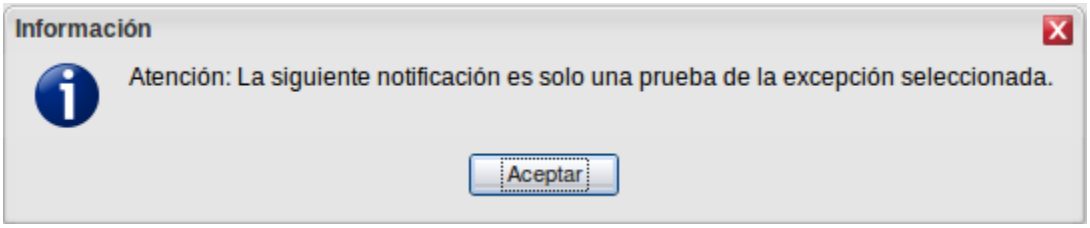


***Figura 12**: Prototipo de interfaz Probar Excepción.*

Requisito funcional: Buscar Excepción

Tabla 6: *Descripción del requisito funcional Buscar Excepción.*

Conceptos tratados	**Conceptos**	**Atributos**
	Excepción.	Código, mensaje, descripción, nombre, tipo, identificador.
Precondiciones	**Precondiciones**	**Pre-requisito**
	No procede.	No procede.
Descripción	Para buscar una excepción el desarrollador debe especificar dentro de cual subsistema desea buscar. Introduce el criterio de búsqueda en el campo de texto indicado para la realización de la acción. El sistema arroja los resultados listando las coincidencias según la búsqueda realizada.	
Validaciones	No se permiten valores nulos en el campo de búsqueda.	
Post-condiciones		
Post-requisito	No procede.	

Prototipo de interfaz de usuario: Buscar Excepción

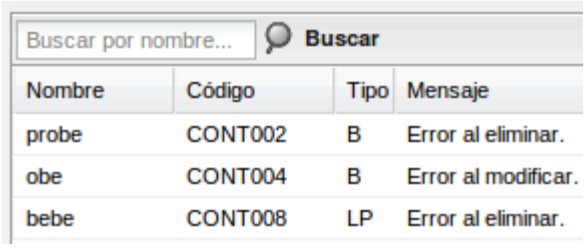


Figura 13*: Prototipo de interfaz Buscar Excepción.*

2.2.2 Requisitos no funcionales

Los requisitos no funcionales son cualidades o propiedades que el producto final debe poseer. Con los requisitos no funcionales se especifican propiedades del sistema que tienen que ver con el rendimiento, restricciones de ambiente y desarrollo, dependencias de plataforma y mantenimiento.

Los requisitos no funcionales que debe poseer el componente a desarrollar son los establecidos por el centro CEIGE al inicio del proceso de desarrollo, a continuación son descritos algunos de los más importantes.

Usabilidad

El componente será usado por los desarrolladores los cuales poseen los conocimientos necesarios para trabajar con el componente.

Rendimiento

Los tiempos de respuesta y velocidad de procesamiento de la información serán rápidos, no mayores de 5 segundos para las actualizaciones y 20 para las recuperaciones.

Seguridad

Autenticación y Autorización (Contraseña de acceso). Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos. El acceso se realizará a través del Sistema de Gestión Integral de Seguridad (ACAXIA). La atención al sistema incluyendo el mantenimiento de las bases de datos, así como la salva de la información, se realizará de forma centralizada por el administrador.

Software

Para el cliente:

- Navegador Mozilla Firefox 3.0 o superior.
- Sistema operativo Windows 98 o superior o Linux.

Para el servidor:

- Sistema operativo Linux en cualquiera de sus distribuciones.
- Un servidor Apache 2.0 o superior con módulo PHP 5.0 disponible. Este debe estar configurado con la extensión “pgsql” incluida.
- Un servidor de base de datos PostgreSQL 8.3 o superior.

Hardware

Para el cliente:

- Requerimientos mínimos: Procesador Pentium III a 1GHz con 256Mb de memoria RAM.

- Tarjeta de red.

Para el servidor:

- Requerimientos mínimos: Procesador Pentium IV a 2GHz de velocidad de procesamiento y 1Gb de memoria RAM.
- Al menos 40Gb de espacio libre en disco duro.
- Tarjeta de red.

2.3 Modelo conceptual

Con la construcción del modelo conceptual se logra una mejor comprensión del dominio del problema; el modelo conceptual no es más que una representación visual de los conceptos u objetos del mundo real que son significativos para el problema o el área que se analiza; representando las clases conceptuales, no los componentes de software. Puede verse como un modelo que comunica a los interesados, cuáles son los términos importantes y cómo se relacionan entre sí (25).

Finalmente con esta actividad se concreta el dominio del problema y los requisitos con una perspectiva de clasificación por objetos. Para desglosar el dominio del problema hay que identificar los conceptos y las asociaciones referentes al mismo que se consideran importantes.

El diagrama expresa gráficamente como el Sauxe está compuesto por subsistemas los cuales pueden tener excepciones o no; el usuario que en este caso será un desarrollador se encargará de gestionar las excepciones del subsistema al cual pertenece.

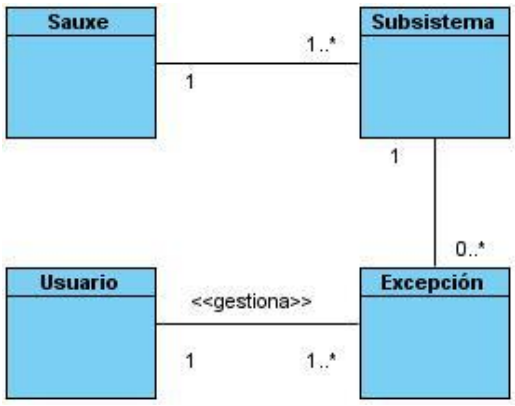


***Figura 14**: Modelo Conceptual.*

2.4 Patrones

2.4.1 Patrón arquitectónico Modelo-Vista-Controlador

El patrón Modelo-Vista-Controlador (MVC) describe una forma, muy utilizada para aplicaciones web, de organizar el código de una aplicación según los datos, la interfaz y la lógica con el fin de poder reutilizar componentes fácilmente. MVC sugiere la separación del software en tres estratos Modelo, Vista y Controlador:

Modelo: Es la representación de la información que maneja la aplicación. El modelo en sí son los datos del sistema que proveen información al usuario o a la aplicación. Representa las estructuras de datos o clases que contienen las funciones para la extracción, modificación o eliminación de la información.

Vista: Es la representación grafica del modelo disponible para la interacción con el usuario. Una vista puede ser una página web o una parte de una página, el HTML y CSS enviados en el navegador.

Controlador: Actúa como intermediario entre el Modelo, la Vista y cualquier otro recurso necesario para generar una página. Es la capa encargada de manejar y responder las solicitudes del usuario, procesando la información necesaria y modificando el modelo si es necesario.

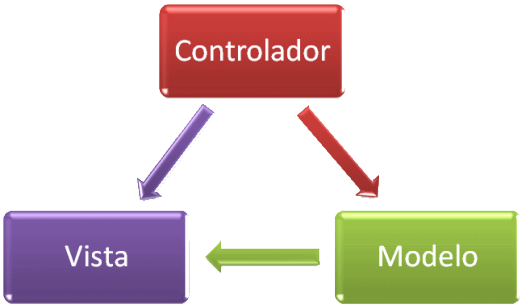


***Figura 15**: Patrón arquitectónico Modelo-Vista-Controlador.*

2.4.2 Patrones de diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software (26).

Los patrones describen un problema que ocurre constantemente en el entorno y describen la solución básica de manera genérica para que ésta pueda ser utilizada en reiteradas ocasiones, evitando hacer lo mismo varias veces.

Patrón de diseño **Singleton o solitario**: Es un patrón creacional que garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Se debe usar cuando:

- Existe la necesidad de que una clase se instancie una sola vez de modo que todos los clientes puedan acceder a esa única instancia desde un punto conocido.
- La instancia única se puede ampliar mediante subclases y los clientes deben ser capaces de utilizar las instancias de las subclases sin tener que modificar su código.

2.5 Modelo de diseño

En el modelo de diseño, se modela el sistema para que soporte todos los requisitos: los funcionales, los no funcionales. Las abstracciones del modelo de diseño tienen una correspondencia directa con los elementos físicos del ambiente de implementación, con el diseño se crea una entrada para la actividad de implementación futura ya que se capturan los conceptos de las clases, interfaces y sistemas que serán necesarios para la solución.

2.5.1 Diagramas de clases del diseño

Se llama clase a la base para la ejecución de un programa, que determinará la funcionalidad del objeto, estableciendo los datos y métodos que dispondrá. Una clase es un artefacto de modelado que describe un conjunto de objetos con atributos, operaciones y relaciones.

Un diagrama de clases es un diagrama estático que representa la estructura que sigue un sistema basándose en los principales componentes que la integran, sus clases y las relaciones entre ellas. Estos diagramas son utilizados durante el proceso de análisis y diseño de los sistemas con el objetivo de modelar los aspectos estáticos y definir una solución.

A continuación el diagrama de clases propuesto fue concebido teniendo en cuenta las características del patrón arquitectónico Modelo-Vista-Controlador (MVC) el cual se describe en el epígrafe anterior y se aplica a continuación.

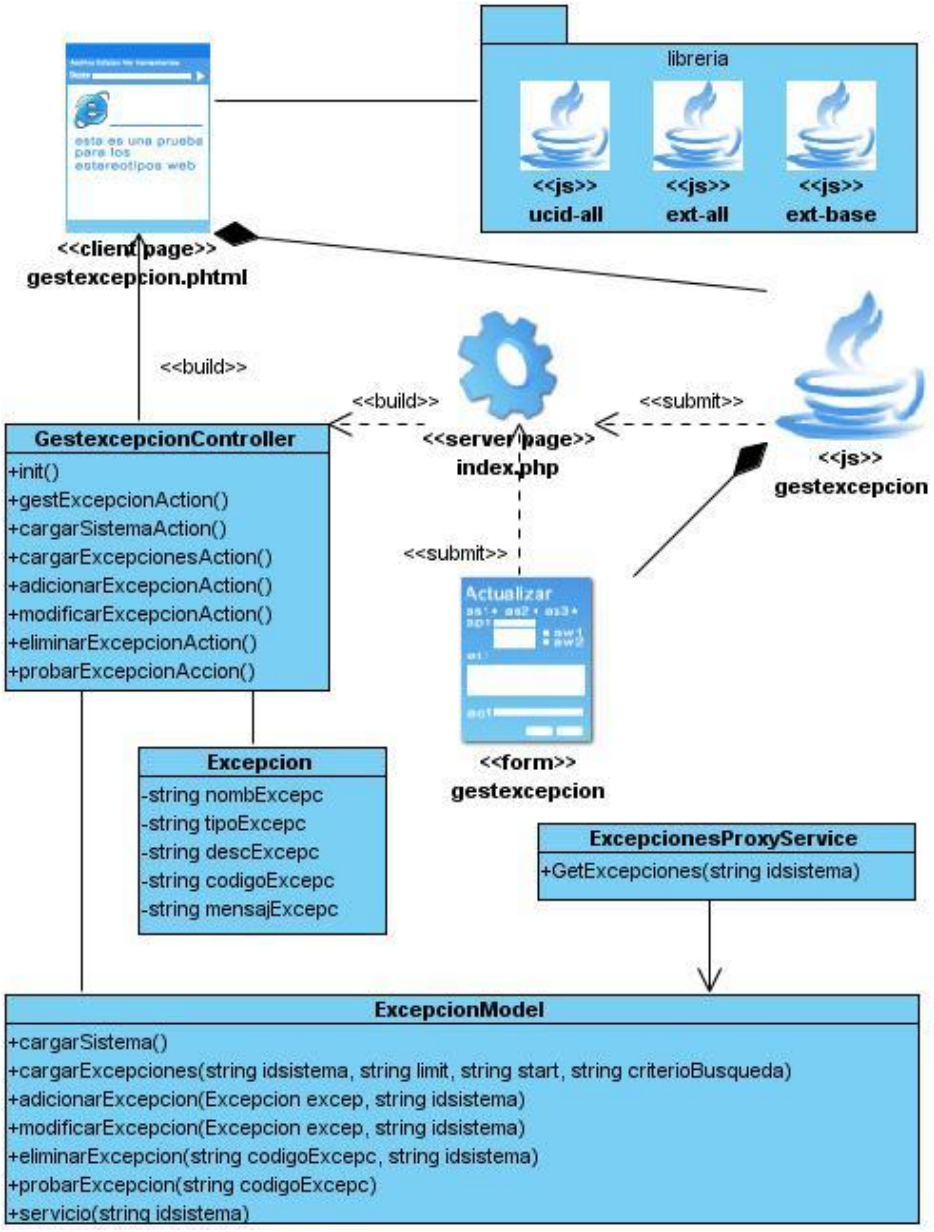


***Figura 16**: Diagrama de clases del diseño.*

El diagrama muestra como la página cliente junto a librerías y scripts componen la vista con la que interactúa el usuario; a través de esta se envían los datos a la clase controladora que es intermediaria entre la vista y el modelo. Este último realiza la lógica del componente y contiene las funciones para la extracción de datos.

2.6 Conclusiones parciales

A lo largo del capítulo se abordaron temas de interés correspondientes a las etapas de análisis y diseño del componente para la configuración visual de las excepciones en el marco de trabajo Sauxe. Ilustrando el proceso con los principales artefactos que propone el modelo orientado a componentes. En este capítulo se han generado los diagramas de proceso, el modelo conceptual con los conceptos más importantes de la herramienta que se quiere desarrollar. Se describió la solución propuesta mediante los requisitos funcionales para así tener una guía de lo que se quiere implementar. Se elaboró el diagrama de clases compuesto por las clases de diseño y sus relaciones, aplicando los patrones de diseño definidos para la solución.

Los objetivos planteados para la realización del presente capítulo han sido cumplidos, ya que resulta en el análisis y diseño de un componente para la configuración visual de las excepciones. Una vez concluido puede darse paso a los flujos de implementación y prueba del mismo.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

El desarrollo de un software es algo complejo y son innumerables las posibilidades de errores, por lo que éste ha de ir acompañado de alguna actividad que garantice la calidad. En el presente capítulo se aborda sobre los puntos más importantes en la implementación del componente para la configuración visual de las excepciones, se describen los estándares de codificación para una mejor legibilidad del código y también la configuración del sistema para su ejecución en un ambiente del mundo real con el diagrama de despliegue. En el capítulo se realiza la validación a la solución propuesta, de manera que se puede comprobar la eficiencia de las clases y operaciones utilizadas para dar respuesta a los distintos requisitos. Para ello se siguieron las métricas de diseño TOC y RC; y pruebas de implementación de caja blanca y caja negra, asegurando así la calidad del software.

3.1 Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y lenguaje o lenguajes de programación utilizados y cómo dependen los mismos unos de otros (27).

3.1.1 Diagrama de componentes

El diagrama de componentes representa la estructura física del sistema, su agrupación por paquetes y muestra las dependencias entre estos.

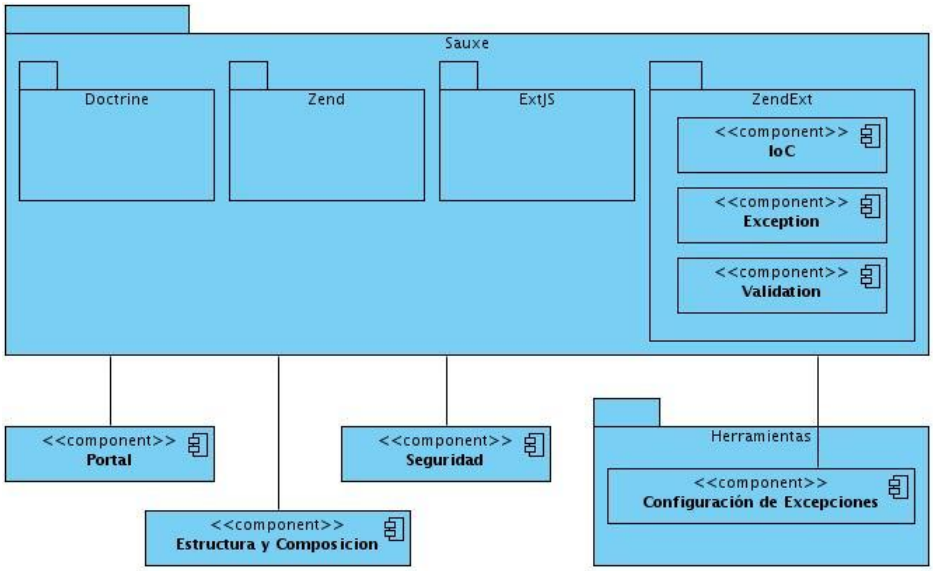


***Figura 17**: Diagrama de componentes.*

3.1.2 Diagrama de despliegue

Los diagramas de despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos (28).

Un diagrama de despliegue describe la configuración del sistema para su ejecución en un ambiente del mundo real. Representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Para el despliegue se deben tomar decisiones sobre los parámetros de la configuración, funcionamiento, asignación de recursos, distribución y concurrencia.

El desarrollador desde su estación de trabajo podrá acceder al sistema que estará desplegado en el servidor web. Dicho servidor se conectará al servidor de bases de datos que es necesario para el funcionamiento interno del marco de trabajo Sauxe. A continuación se muestra el diagrama de despliegue confeccionado.

***Figura 18**: Diagrama de Despliegue.*

3.1.3 Estándares de codificación

Los estándares de codificación son reglamentos de la programación que están enfocadas a la estructura y apariencia física del programa con el objetivo de facilitar la lectura, comprensión y mantenimiento del código. Un estándar de codificación comprende todos los aspectos de la generación de código. Los programadores deben implementar un estándar de forma prudente que tienda siempre a lo práctico. La legibilidad del código fuente influye directamente en el entendimiento que puedan tener los desarrolladores del equipo de trabajo, pues todo software tiene que someterse constantemente a mantenimiento y mejora de sus funcionalidades. El mejor método para lograr que un grupo de desarrolladores mantenga un código de calidad es establecer un estándar de codificación sobre el cual se realizarán revisiones rutinarias.

Los Estándares utilizados en la codificación fueron los siguientes:

- Notación Húngara: Esta convención se basa en definir prefijos para cada tipo de datos y según el ámbito de las variables. También es conocida como notación REDDICK (por el nombre de su creador). La idea de esta notación es la de dar mayor información al nombre de la variable, método o función definiendo en ella un prefijo que identifique su tipo de dato y ámbito.

- Notación PascalCasing: Es como la notación húngara pero sin prefijos. En este caso, los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula.
- Notación CamelCasing: Es parecido al Pascal-Casing con la excepción que la letra inicial del identificador debe estar en minúscula.

Nomenclatura de las clases

Los nombres de las clases comenzarán con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing.

El nombre de las clases estará dado según la función que cumplan, es decir se le agregará al nombre original un prefijo o sufijo que describa el tipo de clase que es la misma. A continuación se listan los prefijos y sufijos determinados para cada tipo de clase.

- Clases controladoras: A las clases controladoras se le adiciona el sufijo “Controller”, ejemplo: GestexcepcionController.
- Clases modelos: A las clases pertenecientes al modelo de negocio se le adiciona el sufijo “Model”, ejemplo: ExcepcionModel.

Nomenclatura de las funciones

Los nombres a emplear para las funciones se escribirán con minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing.

En el caso particular de las funciones de las clases controladoras se les adiciona el sufijo “Action”, ejemplo: adicionarExcepcionAction.

Nomenclatura de las variables

Los nombres a emplear para las variables se escribirán con la notación húngara, en caso de que sea un nombre compuesto se empleará notación CamelCasing.

El nombre de las variables estará dado según la notación húngara, es decir se le agregará al nombre original un prefijo que describa el tipo de dato. A continuación se listan los prefijos a utilizar para cada tipo de dato.

- Arreglos: “arr”
- Objetos: “obj”

- Enteros: “int”
- Cadena: “str”
- Float: “flt”
- Boolean: “boo”

Nomenclatura de los comentarios

Los comentarios deben ser lo bastante claros y precisos de forma tal que se entienda el propósito de lo que se está desarrollando.

Es de gran importancia que tanto clases como métodos sean correctamente comentados para obtener un código fuente más legible y reutilizable, de manera que se pueda dar mantenimiento de manera más fácil a lo largo del tiempo. Antes de la declaración de una clase o método se escribirá una breve descripción donde se explique el propósito del mismo. Este comentario tendrá la siguiente estructura:

- En las clases

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

- En los métodos

```
/**
* Nombre de la función *
* Descripción *
* @author * (en caso de que no sea el autor de la clase)
* @param *(los parámetros que se le pasan a la función con su descripción)
* @throws *(en caso de que dispare una excepción)
* @return *(se pone lo que devuelve la función y un comentario)
/*
```

3.2 Métricas de diseño

El IEEE *Standard Glossary of Software Engineering Terms* [IEEE93] define métrica como una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado.

Las métricas del software permiten medir de forma cuantitativa la calidad de los atributos internos del producto, esto permite al ingeniero evaluar la calidad durante el desarrollo del sistema. Son varios los puntos de vista relacionados con la calidad del software. El objetivo de este epígrafe es desarrollar una evaluación del diseño propuesto para el componente de configuración visual para las excepciones en el marco de trabajo Sauxe. Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del software con medidas que pueden ayudar al desarrollador a juzgar la calidad de un diseño a nivel de componente.

Un aspecto importante a tener en cuenta en la evaluación del diseño, es la creación de métricas básicas inspiradas en el estudio de la calidad del diseño orientado a objetos. Los atributos de calidad que se tienen en cuenta son:

- **Responsabilidad**: Es la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- **Complejidad de implementación**: Es la complejidad de implementación que posee una estructura de diseño de clases.
- **Reutilización**: Es el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- **Acoplamiento**: Es el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.
- **Complejidad del mantenimiento**: Es el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- **Cantidad de pruebas**: Es el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (componente, modulo, clase, conjunto de clases, etc.) diseñado.

Las métricas seleccionadas como instrumento para evaluar la calidad del diseño del componente para la configuración visual de las excepciones en el marco de trabajo Sauxe y su relación con los atributos de calidad son las siguientes:

- Tamaño Operacional de Clase (TOC): Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

Tabla 7: *Atributos de calidad evaluados por la métrica TOC.*

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

Se definieron los siguientes criterios y categorías de evaluación para los atributos de calidad anteriores:

Tabla 8: *Criterios de evaluación para la métrica TOC.*

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

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

Tabla 9: *Atributos de calidad evaluados por la métrica RC.*

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Aumento del RC provoca aumento del Acoplamiento de la clase.
Complejidad de mantenimiento	Aumento del RC provoca aumento de la complejidad del mantenimiento de la clase.

Reutilización	Aumento del RC provoca disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Aumento del RC provoca aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Se definieron los siguientes criterios y categorías de evaluación para los atributos de calidad anteriores:

Tabla 10: *Criterios de evaluación para la métrica RC.*

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

3.2.1 Resultados obtenidos de la aplicación de la métrica TOC

Luego de aplicarse la métrica de diseño TOC se obtuvieron resultados que permiten evaluar el diseño propuesto de calidad aceptable teniendo en cuenta que el 50% de las clases empleadas en el sistema poseen 1 operación lo que conlleva a evaluaciones positivas de los atributos de calidad involucrados (responsabilidad, complejidad de implementación y reutilización).

Tabla 11: *Instrumento de evaluación de la métrica TOC.*

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
GestexcepcionController	9	Media	Media	Media
ExcepcionModel	7	Media	Media	Media
ExcepcionesProxyService	1	Baja	Baja	Alta
Excepcion	1	Baja	Baja	Alta

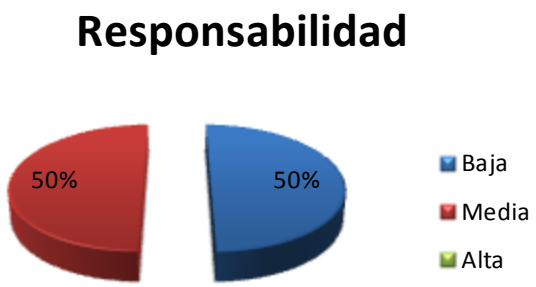


***Figura 19**: Resultados de la evaluación de la métrica TOC para el atributo Responsabilidad.*

***Figura 20**: Resultados de la evaluación de la métrica TOC para el atributo Reutilización.*

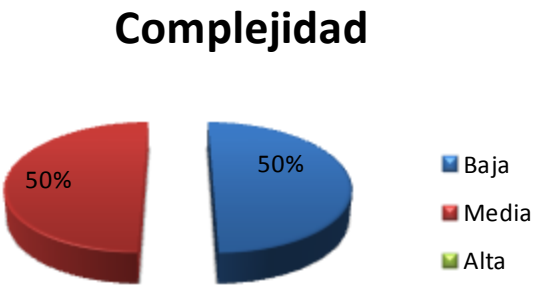


***Figura 21**: Resultados de la evaluación de la métrica TOC para el atributo Complejidad.*

3.2.2 Resultados obtenidos de la aplicación de la métrica RC

Luego de aplicarse la métrica de diseño RC se obtuvieron resultados que permiten evaluar el diseño propuesto de calidad aceptable teniendo en cuenta que el 75% de las clases empleadas en el sistema poseen 1 o menos dependencias con otras clases lo que conlleva a evaluaciones positivas de los atributos de calidad involucrados (acoplamiento, complejidad de mantenimiento, cantidad de pruebas y reutilización).

Tabla 12: *Instrumento de evaluación de la métrica RC.*

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad de Mantenimiento	Reutilización	Cantidad de pruebas
GestexcepcionController	5	Alto	Alta	Baja	Alta
ExcepcionModel	1	Bajo	Baja	Alta	Baja
ExcepcionesProxyService	1	Bajo	Baja	Alta	Baja
Excepcion	0	Ninguno	Baja	Alta	Baja

Figura 22*: Resultados de la evaluación de la métrica RC para el atributo Acoplamiento.*

***Figura 23**: Resultados de la evaluación de la métrica RC para el atributo Complejidad de mantenimiento.*

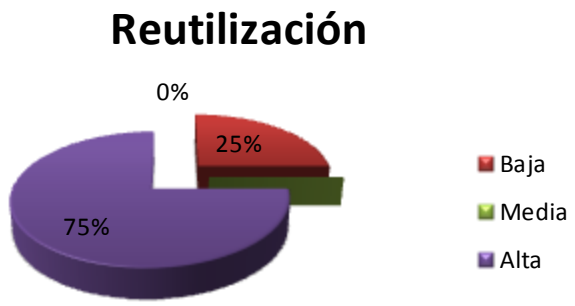


***Figura 24**: Resultados de la evaluación de la métrica RC para el atributo Reutilización.*

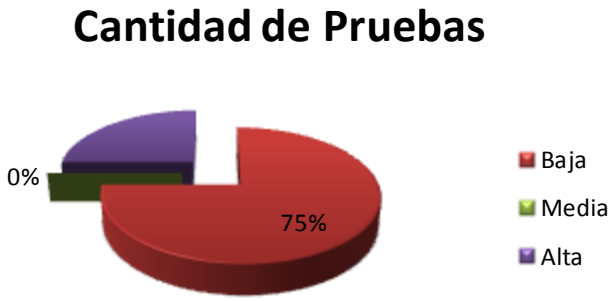


***Figura 25**: Resultados de la evaluación de la métrica RC para el atributo Cantidad de pruebas.*

3.2.3 Matriz inferencia de indicadores de calidad

Con la matriz inferencia de indicadores de calidad se representan los atributos de calidad y las métricas que se utilizaron para medir la calidad del diseño del componente. La matriz demuestra si los resultados de las relaciones entre atributos y métricas son positivos o negativos a partir de una escala numérica. En la escala si los resultados son positivos se le asigna el valor de 1, si son negativos toma valor 0 y si no existe relación es considerada como nula y es representada con un guión simple "-". Luego se puede obtener un resultado general para cada atributo promediando todas sus relaciones no nulas.

Tabla 13: *Rango de valores para la evaluación de la relación Atributo/Métrica.*

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

Tabla 14: *Resultados de la evaluación de la relación Atributo/Métrica.*

Atributo/Métrica	TOC	RC	Promedio
Responsabilidad	1	-	1
Complejidad de Implementación	1	-	1
Reutilización	1	1	1
Acoplamiento	-	1	1
Complejidad de Mantenimiento	-	1	1
Cantidad de pruebas	-	1	1

Resultados obtenidos de la evaluación de los atributos de calidad:

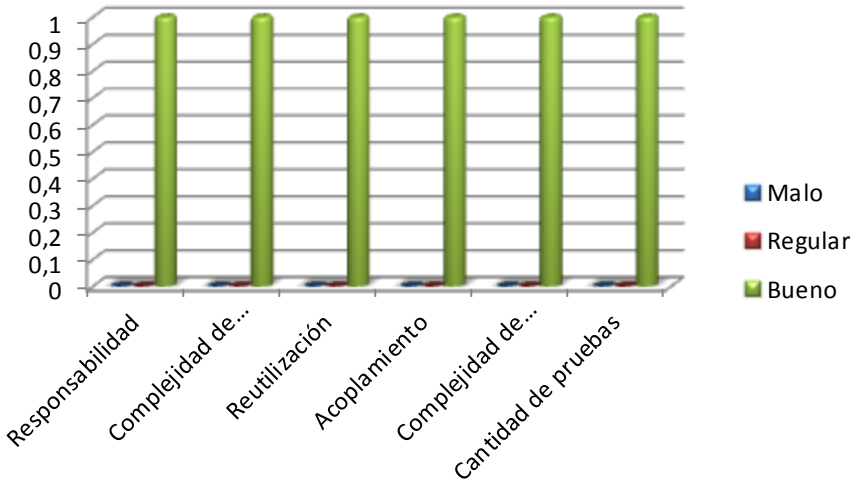


Figura 26*: Matriz de cubrimiento.*

3.3 Pruebas de software

Las pruebas de software forman parte de una etapa imprescindible durante el proceso de desarrollo del software, pues permiten detectar y corregir el máximo de errores posibles antes de ser liberado el producto final. Por lo que el éxito de las mismas incide directamente sobre la percepción de calidad del usuario final. Cuando se considera que un componente está terminado se realizan las pruebas sistemáticas. A continuación se describe como se le realizaron las pruebas funcionales y estructurales al componente de configuración visual de las excepciones.

3.3.1 Pruebas estructurales o de caja blanca

Las pruebas estructurales o de caja blanca permiten diseñar casos de prueba que se centren en obtener una medida de la complejidad lógica del diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. También garantiza que en los casos de prueba obtenidos a través del camino básico se ejecute cada sentencia del programa al menos una vez. Para esto primero se procede a enumerar las sentencias del código y a partir del mismo se construye el grafo de flujo asociado.

```
function cargarExcepcionesAction($idsistema)
{
    $dirsistema =  $this->integrator->seguridad->getRutasistema($idsistema);1
    $limit = $this->_request->getPost('limit');1
    $start = $this->_request->getPost('start');1
    $dirsistema = str_replace('/', DIRECTORY_SEPARATOR, $dirsistema);1
    $direccion = $dirsistema.DIRECTORY_SEPARATOR."comun".DIRECTORY_SEPARATOR."recurs

    if (!file_exists($direccion))2
    {
        $dom = new DOMDocument('1.0', 'UTF-8');3
        $xml = $dom->createElement("excepciones",'');3
        $dom->appendChild($xml);3
        $files = fopen ( $direccion, "a+");3
        $f = fputs($files, $dom->saveXML());3
        $f = fclose($files);3
    }
    $xml = simplexml_load_file($direccion);4
    $excepciones = $xml->children();4
    $cantf = count($excepciones);4
    $arrayExcepciones = array();4
    $nombre='nombre';4
    $cantt = 0;4

    for($cont=(int)$start; $cont<(int)$start+(int)$limit && $cont<$cantf; $cont++)5
    {
        $item = array();6
        $item['nombre']= (string)($excepciones[$cont]->attributes()->$nombre);6
        $item['codigo']= $excepciones[$cont]->getName();6
        $item['tipo']= (string)$excepciones[$cont]->tipo;6
        $item['mensaje']= (string)($excepciones[$cont]->es->mensaje);6
        $item['descripcion']= (string)($excepciones[$cont]->es->descripcion);6
    }
    $result =  array('cantidad_filas' => $cantt, 'datos' => $arrayExcepciones);7
    echo json_encode($result);7
    return;7
}
```

***Figura 27**: Código fuente de la funcionalidad Cargar Excepción.*

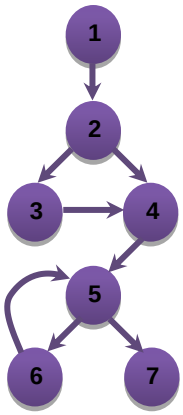


***Figura 28**: Grafo de flujo asociado a la funcionalidad Cargar Excepción.*

La complejidad ciclomática es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez (29). A continuación se realiza el cálculo de la complejidad ciclomática de la funcionalidad Cargar Excepción mediante tres fórmulas descritas, las cuales deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad sea correcto.

1. V (G) = (A – N) + 2

Donde “A” es la cantidad total de aristas y “N” la cantidad total de nodos.

V (G)= (8-7)+2

V (G)=3

2. V (G) = P + 1

Donde “P” es la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G)=2+1

V (G)=3

3. V (G) = R

Donde “R” es la cantidad total de regiones, para cada formula “V (G)” representa el valor del cálculo.

V (G)=3

Con la correcta realización del cálculo de complejidad se determinó el número de posibles caminos por donde el flujo debe circular y el número de casos pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. El cálculo arrojó como valor 3 por lo que seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo:

- Camino básico #1: 1-2-3-4-5-7.
- Camino básico #2: 1-2-4-5-7.
- Camino básico #3: 1-2-4-5-6-5-7.

Para cada camino se realiza un caso de prueba:

- Caso de prueba para el Camino básico #1:

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: El identificador del sistema será una cadena.
Condición de ejecución: El fichero XML no se encuentra creado aun.
Entrada: N/A.
Resultados esperados: Creación del fichero XML.

- Caso de prueba para el Camino básico #2:

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: El identificador del sistema será una cadena.
Condición de ejecución: El fichero XML se encuentra creado y está vacío.
Entrada: N/A.
Resultados esperados: Se carga el fichero XML vacío.

- Caso de prueba para el Camino básico #3:

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: El identificador del sistema será una cadena.
Condición de ejecución: El fichero XML se encuentra creado y contiene datos.
Entrada: N/A.
Resultados esperados: Se carga el fichero XML mostrando los datos que contiene.

Al aplicar los casos de prueba descritos anteriormente se comprobó que el flujo de trabajo de la función es correcto pues cumplió con las condiciones necesarias de la prueba.

3.3.2 Pruebas funcionales o de caja negra

La aplicación fue probada por el Departamento de Calidad del CEIGE (Figura 29 y 30 de los Anexos), donde se comprobó el correcto funcionamiento de la misma a partir de los diseño de caso de prueba (Tablas 16 – 21 de los Anexos).

3.4 Conclusiones parciales

Con el desarrollo de este capítulo fueron expuestos los artefactos generados durante la implementación del componente para la configuración visual de las excepciones, tales como el diagrama de componentes y los estándares de codificación. Se pudo apreciar como el modelo de implementación constituye una entrada fundamental para las pruebas de software. Se describieron las pruebas estructurales y funcionales realizadas que permitieron comprobar el correcto funcionamiento del sistema. Se realizó una validación del diseño mediante la aplicación de métricas para la evaluación de atributos de calidad, lo cual permitió valorar el diseño propuesto de muy bueno dado los resultados obtenidos.

CONCLUSIONES

El desarrollo de los objetivos posibilitó arribar a las siguientes conclusiones:

- La aplicación de las métricas de calidad del diseño arrojaron resultados satisfactorios para cada uno de los atributos evaluados.
- El componente fue probado por el grupo de calidad del CEIGE donde se verificó que el sistema cumple con las características de funcionamiento esperadas demostrando así la calidad del software.
- El componente permitirá agilizar el proceso de configuración de las excepciones en los sistemas desarrollados sobre Sauxe, tarea que hasta el momento está siendo realizada por los desarrolladores manualmente. Como resultado se dotará a los desarrolladores del CEIGE con un marco de trabajo más a fin a sus necesidades.
- Con la automatización de este proceso se reducirá el tiempo de desarrollo del software en el Sauxe y por tanto aumentará la producción de software.

Siendo así el componente para la configuración visual de las excepciones una mejor opción para el desarrollo sobre el marco de trabajo Sauxe, pues cumple con los lineamientos de producción de software de la UCI, logrando la libertad tecnológica del producto.

RECOMENDACIONES

Ningún sistema es estático en el tiempo debido a que la evolución del negocio y la tecnología así lo exigen, es por ello que se recomienda:

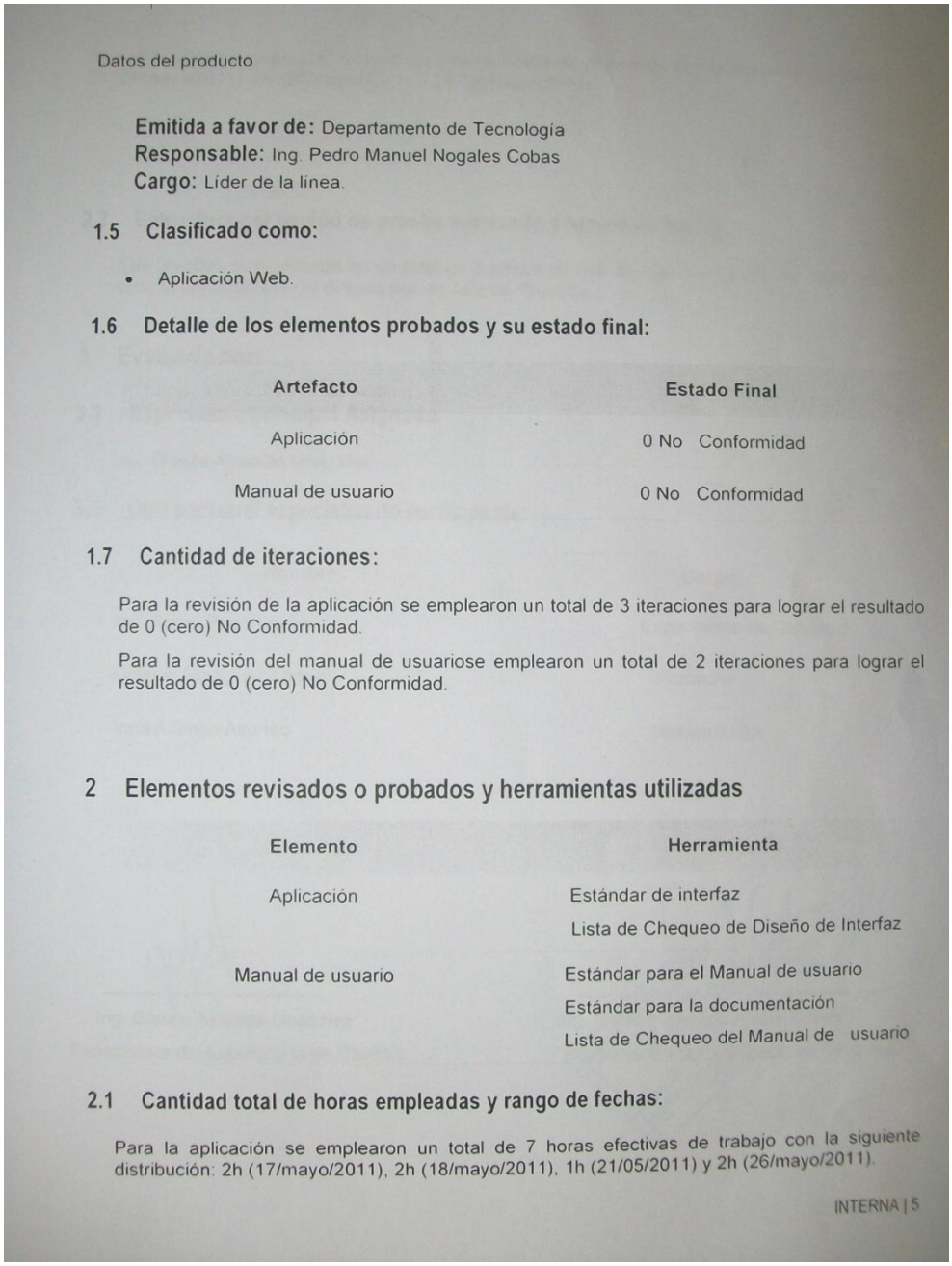
- Utilizar el presente trabajo como bibliografía para posibles investigaciones referentes al tema desarrollado en el mismo.
- Se recomienda que el resultado propuesto en la investigación sea implantado como uso cotidiano e introducido en el desarrollo de las aplicaciones web de gestión que se desarrollen sobre el Marco de Trabajo Sauxe.
- Continuar perfeccionando la herramienta analizada a partir de los nuevos requisitos que puedan surgir como resultado de su explotación.

REFERENCIAS

1. **Potencier, Fabien y Zaninotto, François.** *Symfony la guía definitiva.* s.l. : Apress, 2008. 978-1590597866.
2. **Casas Rescalvo, Esther.** *Framework de desarrollo de Código Abierto.* [En línea] [Citado el: 12 de enero de 2011.] http://www.opensourceworldconference.com/papers/Dia22/Sala%204/Casas_235.pdf.
3. **Felix Isidro, Pablo.** Tecnologico. [En línea] [Citado el: 03 de 11 de 2010.] http://www.mitecnologico.com/Main/ExcepcionesDefinicion.
4. **Microsoft.** MSDN. [En línea] [Citado el: 02 de 12 de 2010.] http://msdn.microsoft.com/es-es/library/ms173161%28VS.80%29.aspx.
5. **Equipo Arquitectura del ERP Cedrux.** 2008.
6. **Producción, Equipo de.** Modelo de Desarrollo orientado a componentes del proyecto ERP-CUBA.
7. **Pupo, Yanisleydi Cañete.** Libro de Ayuda del Marco de Trabajo Sauxe, En su versión 2.0. Ciudad de la Habana : s.n., 2010.
8. **PHP Group.** PHP. [En línea] 2001. [Citado el: 20 de 10 de 2010.] http://www.php.net/manual/es/intro-whatis.php.
9. Territorio PC. [En línea] 2001. [Citado el: 27 de 11 de 2010.] http://www.territoriopc.com/javascript/tutorial_javascript_introduccion.php.
10. **Eguíluz Pérez, Javier.** Introducción a CSS. *Librosweb.es.* [En línea] [Citado el: 17 de 12 de 2010.] http://www.librosweb.es/css.
11. **Liam R. E. Quin.** W3C. [En línea] [Citado el: 16 de Diciembre de 2010.] http://www.w3.org/standards/xml/core.
12. *Secure Programming with the Zend-Framework.* **Esser, Stefan.** Amsterdam : s.n., 2009. Dutch PHP Conference.
13. [En línea] http://www.techtastico.com/.../zend-framework-una-introduccion/.
14. **Core Team.** Doctrine, PHP Object Persistence Libraries and More. [En línea] [Citado el: 20 de 12 de 2010.] http://www.doctrine-project.org/.
15. **Pérez Mata, Manel.** TecnoRetales. [En línea] 03 de 09 de 2009. [Citado el: 14 de 12 de 2010.] http://www.tecnoretales.com/programacion/que-es-doctrine-orm/.
16. **Sencha Inc.** Sencha. [En línea] 2006. [Citado el: 06 de 01 de 2011.] http://www.sencha.com/products/extjs/.
17. Visual Paradigm. [En línea] [Citado el: 24 de 11 de 2010.] http://www.visual-paradigm.com/product/vpuml/.
18. **Grupo Global de Desarrollo de PostgreSQL.** PostgreSQL. [En línea] 1996. [Citado el: 11 de 01 de 2011.] http://www.postgresql.org/about/press/presskit83.html.es.
19. **Sánchez Barriento, Manuel.** QDiario Magazines. [En línea] 06 de 02 de 2008. [Citado el: 10 de 01 de 2011.] http://www.aplicacionesempresariales.com/version-83-de-postgresql.html.
20. **Kabir, Mohammed J.** *La biblia Servidor Apache2.* s.l. : Anaya.
21. **Collins-Sussman, Ben, W. Fitzpatrick, Brian y Pilato, C. Michael.** Version Control with Subversion. [En línea] 2002. [Citado el: 17 de 01 de 2011.] http://svnbook.red-bean.com/nightly/es/svn-book.pdf.

22. **PHP Company.** Zend The PHP Company. [En línea] [Citado el: 09 de 01 de 2011.] http://www.zend.com/en/products/studio/features.
23. MozillaEs. [En línea] [Citado el: 18 de 01 de 2011.] http://www.mozilla-europe.org/es/firefox/features/.
24. **Pressman, Roger S.** *Ingeniería de Software, un enfoque práctico.* 2002.
25. **Larman, Craig.** *UML y Patrones.*
26. **Tedeschi, Nicolás.** MSDN. [En línea] [Citado el: 24 de 3 de 2011.] http://msdn.microsoft.com/es-es/library/bb972240.aspx.
27. umed.net. [En línea] [Citado el: 3 de 10 de 2010.] http://www.eumed.net/libros/2009c/584/RUP%20Diseno%20e%20implementacion%20del%20sistema.htm.
28. **Marca Huallpara, Hugo Michael y Quisbert Limachi, Nancy Susana.** Universidad Salesiana de Bolivia. [En línea] [Citado el: 19 de 03 de 2011.] http://virtual.usalesiana.edu.bo/web/practica/archiv/despliegue.doc.
29. **Pérez Ramírez, Oilede y García López, Yonisley.** Desarrollo del Componente Trabajador del Subsistema Capital Humano del sistema Cedrux. Habana : s.n., 2009.
30. **PHP Company.** Zend The PHP Company. [En línea] [Citado el: 06 de 10 de 2010.] http://www.zend.com/en/community/framework.
31. **CollabNet.** Open Source Software Engineering Tools. [En línea] 2004. [Citado el: 24 de 01 de 2011.] http://tortoisesvn.tigris.org/.
32. **Proyecto mozilla.** Mozilla europe. [En línea] [Citado el: 07 de 02 de 2011.] http://www.mozilla-europe.org/es/firefox/3.0/releasenotes/.
33. **Fronckowiak, John.** IBM. [En línea] 01 de 07 de 2008. [Citado el: 07 de 01 de 2011.] http://www.ibm.com/developerworks/web/library/wa-aj-extjs/.
34. **Duque, Orlando.** elpimiento.cl. [En línea] 10 de junio de 2009. [Citado el: 20 de 01 de 2011.] http://www.elpimiento.cl/index2.php?option=com_content&do_pdf=1&id=13596.
35. **Martínez Madrid, Natividad.** Departamento de Ingeniería Telemática. [En línea] [Citado el: 23 de 11 de 2010.] http://www.it.uc3m.es/tsirda/material/Tema10.pdf.
36. **Eguiluz, Javier.** Symfony.es. [En línea] [Citado el: 28 de enero de 2011.] http://www.symfony.es/que-es-symfony/.
37. **Boda, Pedro, y otros, y otros.** *Zend Framework Manual en español.* 2009.
38. **Küng, Stefan, Onken, Lübbe y Large, Simon.** [En línea] [Citado el: 03 de 02 de 2011.] http://tortoisesvn.net/docs/nightly/TortoiseSVN_es/.
39. **Gracia, Joaquin.** IngenieroSoftware. [En línea] 27 de mayo de 2005. [Citado el: 25 de 3 de 2011.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

ANEXOS



Datos del producto

Emitida a favor de: Departamento de Tecnología
Responsable: Ing. Pedro Manuel Nogales Cobas
Cargo: Líder de la línea.

1.5 Clasificado como:

- Aplicación Web.

1.6 Detalle de los elementos probados y su estado final:

Artefacto	Estado Final
Aplicación	0 No Conformidad
Manual de usuario	0 No Conformidad

1.7 Cantidad de iteraciones:

Para la revisión de la aplicación se emplearon un total de 3 iteraciones para lograr el resultado de 0 (cero) No Conformidad.

Para la revisión del manual de usuariose emplearon un total de 2 iteraciones para lograr el resultado de 0 (cero) No Conformidad.

2 Elementos revisados o probados y herramientas utilizadas

Elemento	Herramienta
Aplicación	Estándar de interfaz
	Lista de Chequeo de Diseño de Interfaz
Manual de usuario	Estándar para el Manual de usuario
	Estándar para la documentación
	Lista de Chequeo del Manual de usuario

2.1 Cantidad total de horas empleadas y rango de fechas:

Para la aplicación se emplearon un total de 7 horas efectivas de trabajo con la siguiente distribución: 2h (17/mayo/2011), 2h (18/mayo/2011), 1h (21/05/2011) y 2h (26/mayo/2011).

INTERNA | 5

Figura 29: *Acta de Liberación del producto.*

UCI | CIG-CAL-DI : ACTA DE LIBERACIÓN

Para el manual se emplearon un total de 5 horas efectivas de trabajo con la siguiente distribución: 2h (24/mayo/2011), 2h (26/mayo/2011) y 1 h (30/mayo/2011).

2.2 Estructura del equipo de prueba empleado y turnos de trabajo:

Las pruebas se realizaron en un total de 3 turnos de trabajo, con 1 probador en cada turno y toda la actividad estuvo dirigida por un Jefe de Pruebas.

3 Evaluado por:

3.1 Especialista principal Asignado:

Ing. Giselle Almeida González

3.2 Otro personal especializado participante:

Nombre	Cargo
Ing. Ilianni Pupo Leyva	Especialista de Calidad
Orlienis Rodíguez López	Probador
Inna Alfonso Alfonso	Desarrollador

Ing. Giselle Almeida González
Especialista del Laboratorio de Calidad

Ing. Pedro Manuel Nogales Cobas
Responsable por el Equipo de Desarrollo

INTERNA | 6

Figura 30: *Acta de liberación del producto.*

Tabla 15: *No conformidades detectadas en la aplicación.*

Fecha	Probador	Elemento	Etapa de detección	No	No conformidad
17 de mayo de 2011	Orlienis Rodríguez Lopez	Aplicación	Primera Iteración	1	Al adicionar y modificar un nombre permite caracteres especiales.
17 de mayo de 2011	Orlienis Rodríguez Lopez	Aplicación	Primera Iteración	2	El campo buscar presenta problemas, al presionar Enter se queda el cursor al final del campo.
17 de mayo de 2011	Orlienis Rodríguez Lopez	Aplicación	Primera Iteración	3	Los nombres de los campos (adicionar, modificar, etc.) no deben estar en negrita y en el nombre del campo “Probar Excepción”, excepción debe empezar con minúscula.
18 de mayo de 2011	Orlienis Rodríguez Lopez	Aplicación	Segunda Iteración	1	En los DCP se debe especificar el mensaje que va a mostrar la aplicación.
18 de mayo de 2011	Orlienis Rodríguez Lopez	Aplicación	Segunda Iteración	2	Especificar en el DCP el escenario de prueba Aplicar.
18 de mayo de 2011	Orlienis Rodríguez Lopez	Aplicación	Segunda Iteración	3	La descripción del escenario de prueba no es centrada y el nombre del requisito debe estar enumerado.
21 de mayo de 2011	Orlienis Rodríguez Lopez	Aplicación	Tercera Iteración	1	Adicionar en el DCP de eliminar, el escenario de prueba "Eliminar excepción estando en uso."
26 de mayo de 2011	Orlienis Rodríguez Lopez	Aplicación	Tercera Iteración	2	Al adicionar o modificar dejando campos vacíos, el sistema debe mostrar un mensaje especificando el error.
26 de mayo de 2011	Orlienis Rodríguez Lopez	Aplicación	Tercera Iteración	3	Cuando vamos a modificar y no modificamos ningún dato, el sistema no debe mostrar el mensaje de que los campos fueron modificados.

Tabla 16: *Caso de prueba Buscar excepción.*

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1. Buscar excepción.	Buscar una excepción en el subsistema seleccionado.	EP 1.1: Buscar excepción.	– Se introduce el criterio de búsqueda. – Presionar el botón Buscar. – El sistema mostrará el resultado de la búsqueda.

Tabla 17: *Caso de prueba Adicionar excepción.*

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1. Adicionar excepción.	Adicionar una nueva excepción al subsistema seleccionado.	EP 1.1: Adicionar excepción correctamente.	– Se introducen correctamente los datos. – Se presiona el botón Aceptar. – Se presiona el botón Aceptar de la ventana de información que muestra el mensaje:" La excepción fue insertada satisfactoriamente.".
		EP 1.2: Adicionar excepción incorrectamente.	– Se introducen datos incorrectos. – Se presiona el botón Aceptar. – Se muestra el mensaje de "Valor incorrecto".
		EP 1.3: Adicionar excepción dejando campos vacíos.	– Se introducen los datos dejando campos vacíos. – Se presiona el botón Aceptar. – Se muestra el mensaje de "Campo obligatorio."
		EP 1.4: Adicionar excepción registrando el mismo código de uno ya registrado.	– Se introducen los datos registrando un código ya existente. – Se presiona el botón Aceptar. – Se presiona el botón Aceptar de la ventana de información que muestra el mensaje: "La excepción no fue insertada, ya existe una excepción con ese código.".
		EP 1.5: Cancelar.	– Se introducen los datos o no. – Se presiona el botón Cancelar.
		EP 1.6: Aplicar.	– Se introducen los datos. – Se presiona el botón Aplicar. – Se presiona el botón Aceptar de la ventana de información que muestra el mensaje:" La excepción fue insertada satisfactoriamente.". – Se muestra el formulario con los campos vacíos.

Tabla 18: *Caso de prueba Eliminar excepción.*

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1. Eliminar excepción.	Eliminar una excepción al subsistema seleccionado.	EP 1.1: Eliminar excepción.	– El sistema muestra el mensaje: "Está seguro que desea eliminarla?". – Presionar el botón Aceptar.
		EP 1.2: Cancelar.	– El sistema muestra el mensaje: "Está seguro que desea eliminarla?". – Presionar el botón Cancelar.

Tabla 19: *Caso de prueba Listar excepción.*

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1. Listar excepción.	Listar las excepciones de un subsistema seleccionado.	EP 1.1: Listar excepción.	– El sistema mostrará las excepciones que contiene el subsistema.

Tabla 20: *Caso de prueba Modificar excepción.*

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1. Modificar excepción.	Modificar una nueva excepción al subsistema seleccionado.	EP 1.1: Modificar excepción correctamente.	– Se introducen correctamente los datos. – Se presiona el botón Aceptar. – Se presiona el botón Aceptar de la ventana de información que muestra el mensaje:" La excepción fue modificada satisfactoriamente.".
		EP 1.2: Modificar excepción incorrectamente.	– Se introducen datos incorrectos. – Se presiona el botón Aceptar. – Se muestra el mensaje de "Valor incorrecto.".
		EP 1.3: Modificar excepción	– Se introducen los datos dejando campos vacíos. – Se presiona el botón Aceptar.

	dejando campos vacíos.	– Se muestra el mensaje de “Campo obligatorio.”
	EP 1.4: Cancelar.	– Se introducen los datos o no. – Se presiona el botón Cancelar.
	EP 1.5: Modificar una excepción que fue eliminada.	– Se introducen los datos. – Se presiona el botón Aceptar. – Se presiona el botón Aceptar de la ventana de información que muestra el mensaje:” No se modificó la excepción, ya fue eliminada.”.

Tabla 21: *Caso de prueba Probar excepción.*

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1. Probar excepción.	Prueba una excepción del subsistema seleccionado.	EP 1.1: Probar excepción.	– El sistema muestra el mensaje de notificación: “Atención: La siguiente notificación es solo una prueba de la excepción seleccionada.”. – Presionar el botón Aceptar. – El sistema muestra el mensaje de la excepción. – Presionar el botón Aceptar.

GLOSARIO

Actividad: Trabajo dentro de un proceso que será desempeñado por una combinación de recursos humanos y computacionales.

API: La interfaz de programación de aplicaciones es un conjunto de funciones residentes en bibliotecas.

Artefactos: Productos tangibles del proyecto que son creados, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.

Componente: Unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio.

Eficiencia: Es el logro de los objetivos y metas con el mínimo de los recursos y tiempo. Es el resultado del mejor aprovechamiento de los recursos utilizados para la realización de las actividades que se prevén a fin del cumplimiento de una meta o acción determinadas.

Funcionalidad: Coherencia entre las necesidades detectadas y los resultados que se obtienen con el uso del material. Es lo que un producto puede hacer. Probar la funcionalidad significa asegurar que el producto funciona tal como estaba especificado.

Herramienta: es un objeto elaborado a fin de facilitar la realización de una tarea mecánica.

Método: Conjunto de instrucciones a las que se les da un determinado nombre de tal manera que sea posible ejecutarlas en cualquier momento sin tenerlas que reescribir sino usando sólo su nombre.

Métrica: Medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado.

Proceso: Un proceso se define como un conjunto de tareas, actividades o acciones interrelacionadas entre sí que, a partir de una o varias entradas de información, materiales o de salidas de otros procesos, dan lugar a una o varias salidas también de materiales (productos) o información con un valor añadido.

Reusabilidad: Capacidad de un componente y un subsistema para ser usado por otras aplicaciones en otros escenarios. Esta minimiza la duplicación de componentes así como el tiempo de implementación.

Reutilización: Acción de volver a utilizar los bienes los bienes o productos. Ver Reusabilidad.

Sistema: Es un conjunto organizado de objetos o partes interactuantes e interdependientes, que se relacionan formando un todo unitario y complejo.

Software: se refiere al equipamiento lógico o soporte lógico de una computadora digital, y comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica.

Software libre: Es la denominación del software que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, cambiado y redistribuido libremente.

Soporte: acciones que permiten mantener disponibles los recursos de hardware o software que necesita un producto de software.

Subsistemas: En la misma definición de sistema, se hace referencia a los subsistemas que lo componen, cuando se indica que el mismo está formado por partes o objetos que forman el todo.

Usuario: El usuario es la persona que consume o usa el producto, bien o servicio.

Validación: Confirmación mediante el suministro de evidencia objetiva de que se han cumplido los requisitos para una utilización o aplicación especifica prevista.