Universidad de las Ciencias Informáticas

Facultad 3

Título: Componente para la configuración visual de las validaciones en el marco de trabajo Sauxe.

Trabajo de Diploma para optar por el título de Ingeniero Informático

Autor(es): *Alexander García Mariño*

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

Ing. Javier Ruiz Durán

La Habana

Junio 2011

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al <nombre área> 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 _____.

_____

Alexander García Mariño

_____ _____

Ing. Pedro Manuel Nogales Cobas Ing. Javier Ruiz Durán

DATOS DE CONTACTO

Tutor: Ing. Pedro Manuel Nogales Cobas

CEIGE, Facultad 3, Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: pmnogales@uci.cu

Tutor: Ing. Javier Ruiz Durán

CEIGE, Facultad 3, Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: aanicot@uci.cu

AGRADECIMIENTOS

Agradezco a todos los que supieron brindarme su ayuda, en especial a mis padres y mi querida esposa que siempre me apoyaron. También les agradezco a mis tutores Pedro Manuel Nogales Cobas y Javier Ruiz Durán que me ayudaron en los momentos necesarios, a mis compañeros de proyecto que estuvieron presentes en cada momento para brindarme su ayuda. A todos les entrego toda mi gratitud y decirles que siempre estaré ahí presente para lo que necesiten.

DEDICATORIA

Dedico este trabajo a mis seres queridos que son todo en mi vida.

Mis padres y mi hermano que siempre supieron confiar en mí y apoyarme en todo brindándome su amor y cariño incondicional, que estuvieron conmigo en la buenas y malas. Tienen todo mi amor y siempre están presentes en mi corazón.

Mi querida esposa, ejemplo de amor y gratitud, que lo es todo para mí, que me inspira cada minuto a ser el hombre que soy, que me hace sentir el hombre más feliz del mundo y me hace sentir necesario, le digo gracias por existir.

RESUMEN

El marco de trabajo Sauxe, desarrollado por el Centro de Informatización de la Gestión de Entidades (CEIGE), con el objetivo de soportar el desarrollo de aplicaciones web de gestión, es orientado a componentes y provee una estructura genérica para el desarrollo de aplicaciones web de gestión posibilitándole al desarrollador lograr una mayor estandarización, flexibilidad, integración y agilidad en el desarrollo del producto final. Para esto Sauxe se basa en la reutilización de componentes como IoC, Excepciones, Traza, Validaciones. Este último es el que se encarga de configurar las validaciones del lado del servidor en los sistemas que utilizan Sauxe.

La configuración de este componente se realiza sobre un XML donde se encuentran las validaciones de todos los sistemas en desarrollo sobre Sauxe, este proceso se realiza totalmente manual por lo que resulta lento y engorroso además de que no se prevé el crecimiento del XML.

Con la investigación realizada se desarrolló un componente que permite la configuración visual de las validaciones en el marco de trabajo Sauxe, brindándole a los desarrolladores una herramienta que facilite el proceso de configuración de validaciones. De esta forma tendrán una medida del crecimiento del XML, estandarizarán la declaración de las validaciones en el mismo y podrán realizar funciones como adicionar, modificar, eliminar y buscar la validación de forma visual evitándose interactuar con el XML. Este desarrollo fue basado sobre políticas de software libre, debido a esto se utilizaron tecnologías web tales como (Apache, PostgreSQL, PHP, XML).

Palabras claves: Validación, componente, marco de trabajo, configuración.

Índice de contenido

Índice de contenido

Índice de contenido

Índice de Figuras

Índice de Tablas

Introducción

El mundo actual es un mundo tecnológico, durante las últimas décadas el desarrollo de las tecnologías ha mostrado un avance agigantado en el mundo, apoyándose con mayor fuerza en una de sus ramas denominada Tecnologías de la Información y las Comunicaciones (TIC), siendo esta la de mayor participación en la modernización y eficiencia en los sectores de la sociedad.

Cuba entendiendo la necesidad de una independencia tecnológica y la importancia que tiene el desarrollo de sistemas informáticos ha empezado a desarrollar sus propios sistemas web. Con el objetivo de cumplir esta tarea surge la Universidad de las Ciencias Informáticas (UCI), que desde sus inicios desarrolla productos con el fin de informatizar todos los sectores de la sociedad.

La UCI cuenta con un Centro de Informatización para la Gestión de Entidades (CEIGE) donde se desarrolla el marco de trabajo llamado Sauxe, para implementar aplicaciones web de gestión de forma eficiente.

Sauxe es un marco de trabajo orientado a componentes, de estructura genérica para aplicaciones web de gestión, logrando así una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. Actualmente cuenta con un componente para configurar las validaciones del lado del servidor, pero aun así no se gestiona eficientemente la información referente a dichas validaciones, estas validaciones se encuentran en un XML (Extensible Markup Language), que manejan diversos desarrolladores, que cuando necesitan insertar o modificar alguna validación simplemente abren el XML y manualmente introducen la información.

Las deficiencias mencionadas traen consigo algunos inconvenientes:

- ✓ Se realiza de forma manual, convirtiéndose en un proceso lento y engorroso.
- ✓ No se prevé el crecimiento del XML, debido a que se pueden insertar validaciones que ya pueden existir en el archivo XML.
- ✓ Existe un solo archivo XML para todas las validaciones de todos los sistemas desarrollados sobre el marco de trabajo Sauxe.
- ✓ No existe una vía de probar si la validación cumple con el objetivo por el que fue creada o modificada hasta el momento que sea utilizada.
- ✓ No existe un filtro para buscar una validación en caso que requiera modificación, aumentando así la complejidad del proceso, además de la pérdida de tiempo en realizar la búsqueda.

Teniendo en cuenta la situación planteada, el **problema a resolver** queda expresado de la siguiente forma: ¿Cómo optimizar el proceso de configuración de las validaciones en el marco de trabajo Sauxe?
Para ello se tendrá como **Objeto de estudio**: Marcos de trabajo para aplicaciones web de gestión.
Para resolver el problema planteado se ha propuesto como **objetivo general**: Desarrollar una herramienta que permita la configuración visual de las validaciones en el marco de trabajo Sauxe.
Para dar cumplimiento al objetivo general se trazaron los siguientes **objetivos específicos**:

- ✓ Construir el marco teórico de la investigación para la gestión de las validaciones.
- ✓ Diseñar e implementar una herramienta que permita la configuración visual de las validaciones en el marco de trabajo Sauxe.
- ✓ Validar la solución.

Identificándose como **campo de acción**: Las validaciones en marcos de trabajo para aplicaciones web de gestión.
Se tiene como **idea a defender**:
El desarrollo de un componente que permita la configuración visual de validaciones, permitirá que se optimice el proceso de configuración de las validaciones en el marco de trabajo Sauxe.
Se identificaron como **tareas de la investigación:**

- ✓ Construir el marco teórico de la investigación.
- ✓ Describir los procesos de negocio a informatizar.
- ✓ Realizar una revisión bibliográfica de las herramientas similares existentes.
- ✓ Obtener los requisitos de la herramienta.
- ✓ Definir los escenarios arquitectónicos de la herramienta.
- ✓ Realizar el diseño de la solución.
- ✓ Implementar la solución.
- ✓ Realizar pruebas de calidad a la solución.
- ✓ Documentar la solución obtenida.

Se ha determinado además que se utilizarán los métodos de investigación siguientes:

El **método histórico lógico** se utilizará para realizar una revisión histórica del desarrollo de las validaciones en los marcos de trabajo para aplicaciones web de gestión y determinar si actualmente se están desarrollados sistemas informáticos similares, tomando una posición al respecto.

El **método hipotético deductivo** se utilizará para a partir del planteamiento de la hipótesis de la investigación y siguiendo reglas de deducción poder llegar a un nuevo conocimiento acerca de los marcos de trabajos existentes que de una forma u otra gestionan las validaciones.

El **método analítico-sintético** permitió el procesamiento de la información y arribar a las conclusiones prácticas y teóricas de la investigación.

El **método modelación** para la creación de abstracciones que explican la realidad, por ejemplo, todos los modelos y diagramas presentados en la fase de diseño.

El método empírico que se utilizará en la presente investigación es la observación.

El **método de observación** se utilizará para realizar una evaluación de la situación problemática en cuestión.

El **método experimento** para establecer las diferentes pruebas de calidad con el propósito de determinar la fiabilidad del sistema y el mejoramiento de la usabilidad del mismo, a través de la detección de errores.

En la presente tesis el contenido está estructurado por tres capítulos que se muestran a continuación:

CAPÍTULO 1. Fundamentación Teórica: Se describen los aspectos y conceptos asociados al dominio del problema de la investigación, para su mejor entendimiento. Se presenta el estado del arte de las técnicas implicadas en el objeto de estudio y el conjunto de tecnologías involucradas en el desarrollo de la propuesta.

CAPÍTULO 2. Propuesta de Solución: Se exponen los procesos de negocios y requisitos a cumplir por la herramienta incluyendo los artefactos generados durante el diseño de la misma.

CAPÍTULO 3. Implementación y Prueba: Se exponen los artefactos generados durante la implementación conteniendo también las métricas y las pruebas utilizadas para la validación de la solución propuesta.

Capítulo 1: Fundamentación teórica

1.1. INTRODUCCIÓN.

El presente capítulo constituye el marco teórico de la investigación a realizar. En él se definen una serie de conceptos necesarios para entender el objetivo fundamental del trabajo, profundizando en temas como las validaciones en los marcos de trabajo, haciendo énfasis en los principales marcos de trabajo que existen en el mundo. Se describen las tecnologías que serán utilizadas durante la implementación.

1.2. APLICACIONES WEB DE GESTIÓN.

En el transcurso de los años las necesidades de optimización de los procesos de una o varias empresas, llevó a pensar en una solución para la gestión de la información de los procesos de empresas. Así se comenzaron a informatizar los procesos de las empresas surgiendo como solución las aplicaciones web de gestión que se encuentran dentro de un concepto más amplio que son las aplicaciones web.

Las aplicaciones web son uno de los activos principales de cualquier empresa. Debido a que dan servicio a usuarios finales (por ejemplo aplicaciones de e-banking o e-commerce) como si se trata de aplicaciones corporativas (una intranet), las aplicaciones web manejan información cuya confidencialidad, disponibilidad e integridad es fundamental para las empresas. (1)

En otros términos, una aplicación web es un sistema web que permite a los usuarios ejecutar lógica de negocio a través de un navegador, o lo que es lo mismo: modificar el estado del negocio. Su arquitectura general es la de un sistema Cliente - Servidor. El protocolo principal de comunicación en una aplicación web es HTTP y funciona normalmente desconectado, es decir, el cliente hace una petición al servidor, este la procesa y le devuelve el resultado, terminando la comunicación entre ellos. (2)

Las ventajas más significativas de las aplicaciones web son:

Compatibilidad Multiplataforma: una misma versión de la aplicación puede correr sin problemas en múltiples plataformas como Windows, Linux, Mac, etc.

Actualización: las aplicaciones web siempre se mantienen actualizadas y no requieren que el usuario deba descargar actualizaciones y realizar tareas de instalación.

Acceso inmediato y desde cualquier lugar: las aplicaciones basadas en tecnologías web no necesitan ser descargadas, instaladas y configuradas. Además pueden ser accedidas desde cualquier computadora conectada a la red en donde se accede a la aplicación.

Menos requerimientos de hardware: este tipo de aplicación no consume (o consume muy poco) espacio en disco y también es mínimo el consumo de memoria RAM en comparación con los programas instalados localmente. Tampoco es necesario disponer de computadoras con poderosos procesadores ya que la mayor parte del trabajo se realiza en el servidor en donde reside la aplicación.

Seguridad en los datos: los datos se alojan en servidores con sistemas de almacenamiento altamente fiables y se ven libres de problemas que comúnmente sufren los ordenadores de usuarios comunes como virus y roturas de disco. (3)

De lo expresado al inicio de este epígrafe se concluye que las aplicaciones web de gestión tienen gran importancia dentro de cualquier entidad, debido a que estas aplicaciones web manejan cierta cantidad de información que necesita ser validada desde el momento en que es introducida, para que no ocurran daños irreparables dentro de dichas entidades.

1.3. VALIDACIONES

La validación tiene numerosos significados ligeramente diferentes dependiendo de la empresa y del contexto. Normalmente implica que un plan de aseguramiento de la calidad se ha puesto en su lugar para que un producto o proceso no pueda estar equivocado después de que haya sido validado. (4)

Consiste en comprobar que tanto el algoritmo como el programa cumplen la especificación del problema. También es el proceso de comprobar la precisión de los datos; conjunto de reglas que se pueden aplicar a un control para especificar el tipo y el intervalo de datos que los usuarios pueden especificar. (5)

De acuerdo a lo establecido por los autores consultados se puede concluir que las validaciones no son más que, el proceso de garantizar que un producto se ajusta a las necesidades definidas por el usuario, siendo la manera de comprobar si los datos introducidos cumplen con ciertas condiciones o limitaciones. Dentro del concepto de validaciones que es muy amplio, se encuentran las validaciones de datos que se describen a continuación.

La validación de datos garantiza la corrección y precisión de todos los valores de datos de la aplicación y puede diseñarse utilizando distintos enfoques: código de interfaz de usuario, código de aplicación o restricciones de bases de datos.

Hay varios tipos de validación de datos:

- ✓ **Validación del tipo de datos:**

  Una de las formas más sencillas de validación de datos que consiste en comprobar el tipo de datos. Este tipo de validación de datos responde a preguntas tan simples como "¿Es alfabética la cadena?" y

"¿Es numérico el número?". Normalmente, validaciones tan simples se pueden controlar con la interfaz de usuario de la aplicación. (6)

- ✓ **Comprobación del intervalo:**

Como ampliación del tipo sencillo de validación, la comprobación del intervalo garantiza que el valor proporcionado esté entre los valores máximo y mínimo permitidos. Por ejemplo, un código de servicio con tipo de datos "character" sólo puede admitir caracteres alfabéticos de la A-Z; el resto de caracteres no sería válido. Al igual que ocurre en el caso de la validación del tipo de datos, la interfaz de la aplicación puede proporcionar normalmente la validación de intervalo necesaria aunque, como alternativa de diseño, se puede crear una regla de empresa para controlar validaciones de intervalo. (6)

- ✓ **Comprobación del código:**

La comprobación del código es un poco más complicada y requiere normalmente una tabla de búsqueda. Por ejemplo, supongamos que la aplicación calcula los impuestos sobre ventas correspondientes únicamente a determinados códigos de estados. Será necesario crear una tabla de validación que contenga códigos de estados sujetos a impuestos que estén autorizados. Esta tabla de validación puede formar parte de una regla de empresa, o se puede implementar directamente en la base de datos a efectos de búsqueda mediante consulta. (6)

- ✓ **Validación compleja:**

A veces, una validación sencilla de búsqueda y de campo no es suficiente. Por ejemplo, se considera el caso de una petición de asistencia sanitaria que tiene un importe facturado de 123,57 dólares, pero cuyo importe permitido puede depender de una acumulación variable anual con un límite de 1.500 dólares (sin superar directiva de duración máxima de 100.000 dólares). En esta situación, la validación de datos va más allá de la pantalla de entrada de datos inmediata y consiste también en una evaluación minuciosa de cómo se ha de pagar la petición basándose en los límites de la directiva y en las acumulaciones anual y de vida. Este tipo de validación de datos compleja de varios archivos se suele controlar mejor con reglas de empresa basadas en procedimientos. (6)

1.4. MARCOS DE TRABAJO.

Muchas de las personas que se involucran en el desarrollo de software principalmente los desarrolladores de aplicaciones web, desde sus inicios se enfrentan a disímiles conceptos de los cuales deben tener conocimiento a la hora de desarrollar sus soluciones, unos de estos conceptos es el de marco de trabajo.

Los marcos de trabajo son colecciones de patrones de diseño, interfaces, código real, y así sucesivamente. Ellos conforman un sistema que proporciona cierto apoyo a los programadores. Lo mejor es la práctica de uso y explotación de los marcos existentes y subsistemas en lugar de crear algo parecido a ti mismo. (7)

También se podría decir que un marco de trabajo, en el desarrollo de software es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. (8)

De acuerdo con lo establecido por los autores consultados se puede concluir que los marcos de trabajo son una estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

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

A continuación se explican algunos de los marcos de trabajos existentes y como tratan estos marcos de trabajos las validaciones.

1.4.1. Symfony

Es un marco de trabajo que simplifica el desarrollo de una aplicación web mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Está basado en el clásico patrón de diseño web conocido como arquitectura MVC (Modelo-Vista-Controlador). Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. También, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. (9)

Las validaciones de formularios en Symfony se pueden realizar en el lado del servidor y/o en el lado del cliente. La validación en el servidor es obligatoria para no corromper la base de datos con datos incorrectos. La validación en el lado del cliente es opcional y debe realizarse de forma manual con JavaScript, pero carecen de la posibilidad de configurar esta información de manera cómoda para el desarrollador.

1.4.2. Zend Marco de trabajo (ZF)

Es un marco de trabajo de código abierto utilizado para el desarrollo de aplicaciones web y servicios web para PHP 5. Para su implementación se usa código 100% orientado a objetos. La estructura de los componentes de ZF es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado. A menudo se refiere a este tipo de diseño como "use-at-will" (uso a voluntad). (10)

Aunque se pueden utilizar de forma individual, los componentes de la biblioteca estándar de Zend Marco de trabajo conforman un potente y extensible marco de trabajo de aplicaciones web al combinarse. ZF ofrece un gran rendimiento y una robusta implementación MVC, una abstracción de base de datos fácil de usar, y un componente de formularios que implementa la prestación de formularios HTML, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos. (10)

El ZF realiza el tratamiento de validaciones mediante el componente interno Zend_Validate que proporciona un conjunto de validadores. También proporciona un validador con un simple mecanismo de encadenamiento por el cual múltiples validadores pueden ser aplicados a un simple dato en un orden definido por el usuario. El validador examina la entrada de los datos con respecto a algunos requisitos y produce un resultado booleano si la entrada es correcta, de no ser así el validador, además, puede proporcionar información acerca de los requisitos de la entrada, pero carece de la posibilidad de configurar esta información de manera cómoda para el desarrollador.

1.4.3. ZendExt Marco de trabajo (ZE)

El Marco de Trabajo ZendExt se ha estructurado de manera tal que facilite la reutilización de los diferentes componentes y que sea de fácil entendimiento para todos los programadores que desarrollen sobre el mismo. ZE está formado por los componentes: ZendExt_App, ZendExt_Aspect, ZendExt_ADT, ZendExt_Cache, ZendExt_ExpImp, ZendExt_MVC, ZendExt_Exception, ZendExt_FastResponse, ZendExt_GlobalConcept, ZendExt_IoC, ZendExt_Nomencladores, ZendExt_Trace, ZendExt_Validation, ZendExt_Portal, ZendExt_TransactionManager.

ZE utiliza EXT para implementar la capa de presentación, se apoya en Zend, una extensión de Zend Marco de trabajo para el desarrollo de la lógica del negocio y para la gestión de los datos, utiliza Doctrine.

También cuenta con un componente de transacciones mediante el cual serán salvados automáticamente los datos de modificaciones e inserciones, es decir, ya no será necesaria la implementación por parte del programador de las consultas de inserción, éste solamente deberá programar la obtención de los campos de la presentación y el ZE será el responsable de que los mismos sean guardados en la base de datos.
Ventajas:

- ✓ Los identificadores de cada tupla de las tablas serán generados automáticamente en la base de datos como una secuencia.
- ✓ El antiguo método de try y catch para el lanzamiento de excepciones desaparece, en el nuevo marco con registrar las excepciones en el manager de excepciones, el marco de trabajo será capaz de realizar un tratamiento óptimo de las mismas.
- ✓ No será necesario en cada método de inserción de información a la base de datos ejecutar el método correspondiente en la clase del negocio correspondiente, el marco de trabajo será capaz de que una vez tomados los datos de la presentación salvarlos en la base de datos directamente.

Es importante mencionar que ZE utiliza como patrón arquitectónico el patrón modelo-vista-controlador (MVC), este marco de trabajo en su nueva configuración define que la conexión a la base de datos será configurada en un XML almacenado en la carpeta de recursos comunes del proyecto, además tiene como propósito insertar la programación orientada a aspectos así como la gestión de las validaciones, la cual se realizará con la implementación del patrón de diseño validators. La configuración de este patrón se realiza sobre un archivo XML, donde se encuentran todas las validaciones de todos los sistemas que estén en desarrollo sobre el marco de trabajo, actualmente dicha configuración se realiza totalmente manual lo que resulta engorroso el trabajo y conlleva a la pérdida de tiempo, a la aparición de pequeños errores ya que este proceso se haría manualmente, además de que no se prevé el crecimiento del archivo XML, debido a que se pueden insertar validaciones que ya pueden existir en el XML, también se dificulta la búsqueda de alguna validación en caso de que requiera modificación aumentando así la complejidad del proceso.
A partir del estudio del análisis realizado a los marcos de trabajo existentes se observó que ninguno proporcionaba un componente visual para gestionar la información referente a las validaciones de manera eficiente. Debido a lo antes planteado se decidió implementar un componente para la configuración visual de las validaciones en el marco de trabajo Sauxe garantizando así que no haya pérdida de tiempo innecesariamente además de que se gestione la información referente a las validaciones de manera eficiente utilizando como guía de desarrollo el modelo de desarrollo orientado a componentes seleccionado por el CEIGE.

1.5. MODELO DE DESARROLLO ORIENTADO A COMPONENTES.

Un modelo de procesos del software es una descripción simplificada de un proceso del software que presenta una visión de ese proceso. Estos modelos pueden incluir actividades que son parte de los procesos y productos de software y el papel de las personas involucradas en la ingeniería de software. (11)

Una metodología es el modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Esta sistematización indica como se dividirá un gran proyecto en módulos mas pequeños llamados etapas, y las acciones que corresponden a cada una de ellas, ayudan a definir entradas y salidas para cada una de las etapas y, sobre todo, normaliza el modo en que se administrará el proyecto. Entonces una metodología son los procesos a seguir sistemáticamente para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado. (12)

A diferencia de una metodología que indica cómo hay que obtener los distintos productos parciales y finales, un modelo de desarrollo nos brinda los pasos a seguir para alcanzar el proyecto final, además de brindarnos los artefactos necesarios para documentar el producto.

Para el desarrollo de esta herramienta se utilizó un modelo de desarrollo orientado a componentes, seleccionado por el Centro de Informatización de la Gestión de Entidades (CEIGE). Este es un modelo de desarrollo orientado a las necesidades y artefactos generados durante el proceso de desarrollo de cualquier producto en el centro CEIGE. Es una combinación de diferentes metodologías de las cuales se ha tomado lo que sería más conveniente para llevar a término un proyecto. Este modelo de desarrollo permitirá la generación de artefactos de vital importancia en el análisis y el diseño como son:

- ✓ Modelo de proceso de negocio.
- ✓ Descripción de procesos de negocio.
- ✓ Modelo conceptual.
- ✓ Prototipo de interfaz de usuario.
- ✓ Especificación de requisitos.
- ✓ Modelo de datos.
- ✓ Diagrama de clases.
- ✓ Descripción del diseño de clases.
- ✓ Diagrama de componente.
- ✓ Diagrama de despliegue.

- ✓ Casos de prueba. (13)

Características del modelo:

- ✓ Se utilizan solamente los artefactos necesarios para documentar el producto.
- ✓ Se basa en la reutilización de componentes.
- ✓ Se desarrollan partes pequeñas y se ensambla después el producto.
- ✓ Los flujos se integran a través de la arquitectura de software y de negocio. Todos los flujos de desarrollo de cada fase se integran siempre detrás de la arquitectura de software y de negocio.
- ✓ Existen áreas dedicadas a tareas específicas y especializadas en temas específicos.
- ✓ Se hacen pruebas continuas sobre los componentes y/o productos y los cambios se hacen a tiempo. Antes de poner un componente en el repositorio se hacen pruebas unitarias, y cuando se va a utilizar como parte de otro producto se hacen pruebas de integración, al igual que antes de liberar el producto también. Todo esto demuestra que se está probando en todo el proceso de desarrollo.
- ✓ Las áreas de proceso están especializadas, en temas de apoyo a la producción que es el elemento fundamental de la Subdirección y llevan unido al proceso productivo procesos tales como Investigación, Formación, Gestión del capital humano y calidad.
- ✓ Es un método muy estructurado que funciona bien con gente de poca experiencia.
- ✓ Reduce los riesgos ya que:
    - Provee visibilidad sobre el progreso a través de sus nuevas versiones.
    - Provee retroalimentación a través de la funcionalidad mostrada.
    - Permite atacar los mayores riesgos desde el inicio.
- ✓ La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software.
- ✓ En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.
- ✓ Preocupación por el aprendizaje de los desarrolladores. (13)

1.6. TECNOLOGÍAS UTILIZADAS.

Después de haber realizado un análisis de las necesidades encontradas en la gestión de las validaciones en el marco de trabajo Sauxe se realizó un estudio de las tendencias y tecnologías actuales posibles a emplear para adoptar la tecnología que aporte mayores ventajas en la elaboración de la solución. Estas tecnologías serán descritas en este epígrafe.

1.6.1. Lenguajes de programación

Un lenguaje de programación puede ser utilizado para controlar el comportamiento de una computadora, este consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Un lenguaje de programación permite a uno o más programadores especificar de manera precisa: sobre qué datos una computadora debe operar, cómo deben ser estos almacenados, transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural. (14)

PHP

PHP es un lenguaje de programación usado normalmente para la creación de páginas web dinámicas. Es un acrónimo recursivo que significa "PHP Hypertext Pre-processor". Es conocido como una tecnología de código abierto que resulta muy útil para diseñar de forma rápida y eficaz aplicaciones web dirigidas a bases de datos. Es un potente lenguaje de secuencia de comandos diseñado específicamente para permitir a los programadores crear aplicaciones en la web con distintas prestaciones de forma rápida. Su interpretación y ejecución se realiza en el servidor en el cual se encuentra almacenada la página y el cliente sólo recibe el resultado de la ejecución. Cuando el cliente hace una petición al servidor para que le envíe una página web, enriquecida con código PHP, el servidor interpretará las instrucciones mezcladas en el cuerpo de la página y las sustituirá con el resultado de la ejecución antes de enviar el resultado a la computadora del cliente.

Además es posible utilizarlo para generar archivos PDF, Flash o JPG, entre otros. Permite la conexión a numerosas bases de datos de forma nativa tales como PostgreSQL, MySQL, Oracle, ODBC, IBM DB2, Microsoft SQL Server y SQLite, lo cual permite la creación de Aplicaciones web muy robustas.

PHP tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos tales como UNIX, Linux, Windows y Mac OS X, y puede interactuar con los servidores de web más populares.

Ventajas:

- ✓ Es un lenguaje multiplataforma.
- ✓ Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL.
- ✓ Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- ✓ Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- ✓ Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- ✓ Permite las técnicas de Programación Orientada a Objetos.
- ✓ Biblioteca nativa de funciones sumamente amplia e incluida
- ✓ No requiere definición de tipos de variables.
- ✓ Tiene manejo de excepciones.

Desventajas:

- ✓ 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

JavaScript es un lenguaje de scripts desarrollado por Netscape para incrementar las funcionalidades del lenguaje HTML. Sus características más importantes son:

- ✓ JavaScript es un lenguaje interpretado, es decir, no requiere compilación. El navegador del usuario se encarga de interpretar las sentencias JavaScript contenidas en una página HTML y ejecutarlas adecuadamente.
- ✓ JavaScript es un lenguaje orientado a eventos. Cuando un usuario pincha sobre un enlace o mueve el puntero sobre una imagen se produce un evento. Mediante JavaScript se pueden desarrollar scripts que ejecuten acciones en respuesta a estos eventos.
- ✓ JavaScript es un lenguaje orientado a objetos. El modelo de objetos de JavaScript está reducido y simplificado, pero incluye los elementos necesarios para que los scripts puedan acceder a la información de una página y puedan actuar sobre la interfaz del navegador.
- ✓ JavaScript es soportado por la mayoría de los navegadores como Internet Explorer, Netscape, Opera, Mozilla Firefox, entre otros.

1.6.2. Marco de Trabajo Sauxe

El desarrollo de la herramienta se realizará utilizando el marco de trabajo Sauxe, desarrollado por el Departamento de Tecnología del CEIGE, Sauxe es un marco de trabajo orientado a componentes, de estructura genérica para aplicaciones web de gestión, para lograr así una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo.

Es un producto desarrollado sobre tecnologías libres como el lenguaje PHP, gestor de base de datos Postgres, servidor web Apache, etc. Su administración centralizada de todos los aspectos necesarios a tener en cuenta en el desarrollo de aplicaciones de gestión lo convierte en un producto de punta en esta rama. Permite realizar un número de funcionalidades que hace esta arquitectura muy aplicable para cualquier entorno web en PHP. Permite la gestión de multi-entidad y con esta la compartimentación de la información de cada una de ellas.

Sauxe está compuesto por varios marcos de trabajo, los cuales serán descritos a continuación, estructurados en niveles o capas como se puede apreciar en la siguiente figura.

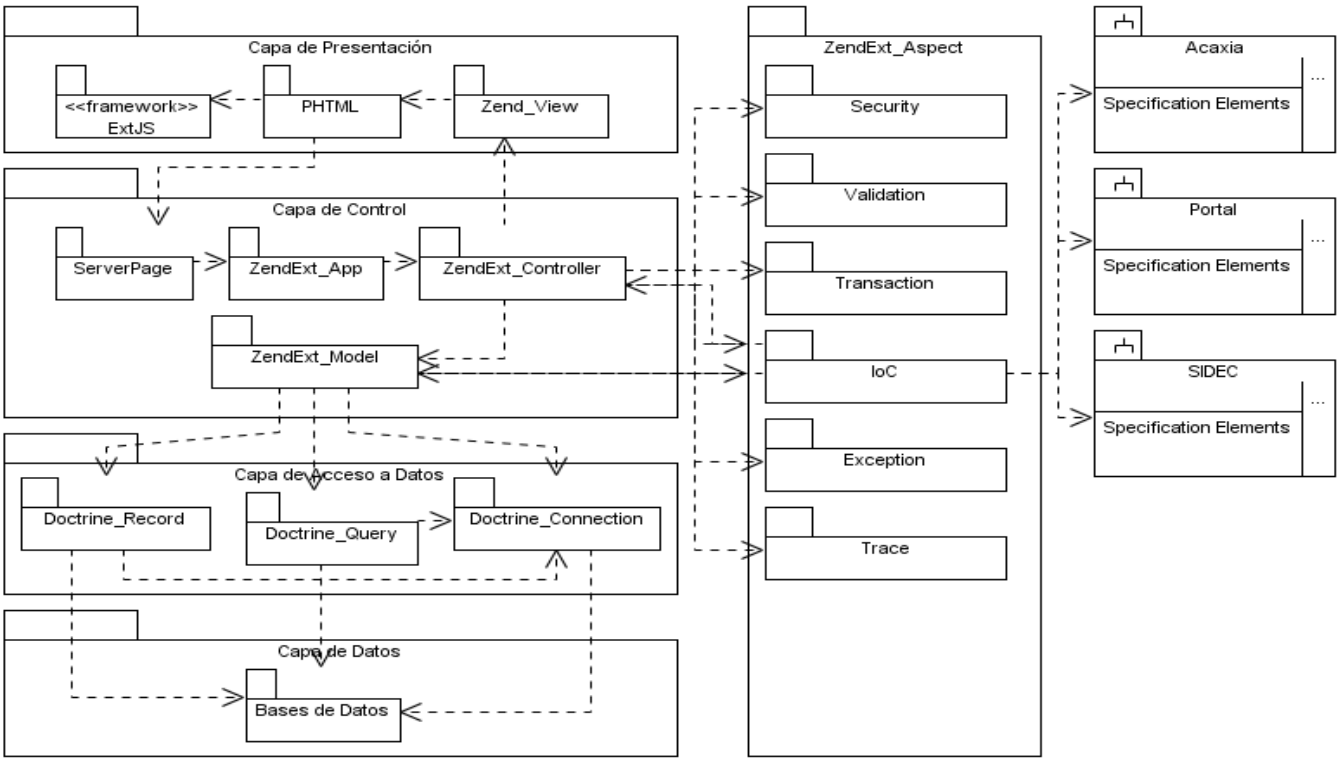

Ilustración 1 Estructura del marco de trabajo Sauxe.

Zend Marco de trabajo

Es un marco de trabajo para desarrollo de aplicaciones Web y servicios Web con PHP. Brinda soluciones para construir sitios web modernos, robustos y seguros. Además es Open Source y trabaja con PHP 5. Presenta entre otras las siguientes características:

- ✓ Proporciona un sistema de caché dividido en frontend y backend, de forma que se puedan almacenar en caché diferentes datos como resultados de funciones, páginas completas, etc., y que esta información se almacene en archivos, en memoria, en base de datos, etc.
- ✓ Simplifica la gestión de archivos de configuración.
- ✓ Proporciona los componentes que forman la infraestructura del patrón MVC.
- ✓ Proporciona una capa de acceso a base de datos, construida sobre PDO pero ampliándola con diferentes características.

- ✓ Proporciona mecanismos de filtrado y validación de entradas de datos.
- ✓ Permite convertir estructuras de datos PHP a JSON y viceversa, para su utilización en aplicaciones AJAX.
- ✓ Proporciona las características necesarias para proveer y consumir servicios web vía REST.
- ✓ Proporciona capacidades de búsqueda sobre documentos y contenidos.
- ✓ Permite consumir y proveer servicios web.

Zend_Ext Marco de trabajo

Es un marco de trabajo de código abierto, que está diseñado para PHP 5 y tiene buenas capacidades de ampliación. Es elaborado a partir de Zend Marco de trabajo cumpliendo con todas sus características. Este trae de novedoso un controlador vertical para el control de las acciones realizadas por las vistas hacia el controlador, un motor de reglas para las validaciones en el servidor, se le incluyó el IoC para la comunicación entre los módulos o componentes. Se le incorporó la integración con el ORM Doctrine Marco de trabajo para trabajo en la capa de abstracción a base de datos y el ExtJs Marco de trabajo para el desarrollo de las vistas.

Doctrine Marco de trabajo

Es un potente y completo sistema ORM (object relational mapper) para PHP 5.2+ que incorpora una DBL (capa de abstracción a base de datos). Uno de sus rasgos importantes es la habilidad de escribir opcionalmente las preguntas de la base de datos orientado a objeto. Esto les proporciona una alternativa poderosa a diseñadores de SQL que mantiene un máximo de flexibilidad sin requerir la duplicación del código innecesario. También permite exportar una base de datos existente a sus clases correspondientes y también convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos.

ExtJs

Es un marco de trabajo Java Script del lado del cliente para el desarrollo de aplicaciones Web. Tiene un sistema dual de licencia: Comercial y Código Abierto. Este marco de trabajo puede correr en cualquier plataforma que pueda procesar POST y devolver datos estructurados (PHP, Java, .NET y algunas otras)

en tiempo de ejecución carga y crea todos los objetos HTML a través del uso intenso del DOM. Los datos son obtenidos mediante mensajes AJAX a través de XML y/o JSON.
Dispone de un conjunto de componentes para incluir dentro de una aplicación web, como:

- ✓ Cuadros y áreas de texto.
- ✓ Campos para fechas.
- ✓ Campos numéricos.
- ✓ Combobox.
- ✓ Radiobuttons y checkboxs.
- ✓ Editor HTML.
- ✓ Elementos de datos (con modos de sólo lectura, datos ordenables, columnas que se pueden bloquear y arrastrar, etc.).
- ✓ Árbol de datos.
- ✓ Pestañas.
- ✓ Barra de herramientas.
- ✓ Menús al estilo de Windows.
- ✓ Paneles divisibles en secciones.
- ✓ Sliders.

También contiene numerosas funcionalidades que permiten añadir interactividad a las páginas HTML, como:

- ✓ Cuadros de diálogo.
- ✓ Quicktips para mostrar mensajes de validación e información sobre campos individuales.

1.7. HERRAMIENTAS UTILIZADAS

1.7.1. Herramientas CASE

Las Herramientas de Ingeniería de Software Asistida por Computadoras (Computer Aided Software Engineering, CASE por sus siglas en inglés) son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y dinero. Dentro de sus objetivos está realizar el diseño del proyecto, cálculo de costo, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación y detección de errores.

1.7.1.1. Visual Paradigm

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite crear diagramas de clases, generar código desde diagramas y documentación.

Visual Paradigm se integra al Entorno de Desarrollo Integrado (Integrated Development Environment, IDE por sus siglas en inglés) de Eclipse. Está diseñado para desarrollar software con Programación Orientada a Objetos, reduce la duración del ciclo de desarrollo brindando ayuda tanto a arquitectos, analistas, diseñadores como a desarrolladores. Busca también automatizar tareas tediosas que pueden distraer a los desarrolladores.

Dentro de sus características fundamentales están:

- ✓ Multiplataforma
- ✓ Interoperabilidad
- ✓ Modelamiento de los Requisitos
- ✓ Colaboración de Equipo
- ✓ Generación de Documentación
- ✓ Editor de Detalles de Casos de Uso
- ✓ Ingeniería de Código

- ✓ Modelado de Procesos de Negocio
- ✓ Integración con Entornos de Desarrollo
- ✓ Modelamiento de Bases de Datos

1.7.1.2. Lenguaje Unificado de Modelado (UML)

Lenguaje estándar para el modelado de software lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema con gran cantidad de software. Lenguaje usado por el Proceso Unificado. Lenguaje que permite a los desarrolladores visualizar el producto de su trabajo (Artefactos) en esquemas o diagramas estandarizados.

Ventajas:

- ✓ Permite modelar sistemas utilizando técnicas orientadas a objetos.
- ✓ Permite especificar todas las decisiones de análisis y diseño, construyéndose así modelos precisos, no ambiguos y completos.
- ✓ Puede conectarse con lenguajes de programación (Ingeniería directa e inversa).
- ✓ Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, etc.).
- ✓ Es un lenguaje muy expresivo que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.
- ✓ Existe un equilibrio entre expresividad y simplicidad, pues no es difícil de aprender ni de utilizar.

1.7.2. ZendStudio Neon para PHP

Zend Studio Neon es un completo entorno integrado de desarrollo para el lenguaje de programación PHP. Está escrito en Eclipse, y disponible para las plataformas Microsoft Windows, Mac OS X y GNU/Linux. Está diseñado para usarse con el lenguaje PHP; sin embargo ofrece soporte básico para otros lenguajes Web, como HTML, Javascript y XML.
Sus principales características son:

- ✓ No requiere la instalación previa de PHP ni del entorno de ejecución de Eclipse.
- ✓ Soporte para PHP 4 y PHP 5.

- ✓ Resaltado de sintaxis, autocompletado de código, ayuda de código y lista de parámetros de funciones y métodos de clase.
- ✓ Inserción automática de paréntesis y corchetes de cierre.
- ✓ Emparejamiento de paréntesis y corchetes.
- ✓ Detección de errores de sintaxis en tiempo real.
- ✓ Funciones de depuración.
- ✓ Manual de PHP integrado.
- ✓ Soporte para navegación en bases de datos y ejecución de consultas SQL.(4)

1.7.3. Apache

Apache, es un servidor de protocolo para la transferencia de hipertextos (Hypertext Transfer Protocol, HTTP por sus siglas en inglés) de software libre para plataformas Unix, Windows, y Macintosh, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. (15) Las características que lo definen son las siguientes:

- ✓ Multiplataforma
- ✓ Es un servidor de web conforme al protocolo HTTP/1.1
- ✓ Modular: Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos, para el desarrollo de módulos específicos.
- ✓ Basado en hebras en la versión 2.0.
- ✓ Incentiva la realimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para la solución de los mismos.
- ✓ Se desarrolla de forma abierta

- ✓ Extensible: gracias a ser modular se han desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor. (16)

1.7.4. Base de Datos PostgreSQL

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos de software libre, dentro de sus características destacan las siguientes:

Alta concurrencia

Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo cambios. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases de datos, eliminando la necesidad del uso de bloqueos explícitos.

Amplia variedad de tipos nativos

PostgreSQL suministra nativamente soporte para:

- ✓ Números de precisión arbitraria.
- ✓ Texto de largo ilimitado.
- ✓ Figuras geométricas (con una variedad de funciones asociadas)
- ✓ Direcciones IP (IPv4 e IPv6).
- ✓ Arrays.

Como características adicionales tiene:

Claves ajenas también denominadas Llaves ajenas o **Claves Foráneas**

Disparadores: Un disparador se define en una acción específica basada en algo ocurrente dentro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica.

Todos los disparadores se definen por seis características:

- ✓ El nombre del disparador.
- ✓ El momento en que el disparador debe arrancar.

- ✓ La tabla donde el disparador se activara.
- ✓ La frecuencia de la ejecución.
- ✓ La función que podría ser llamada.

PostgreSQL posee integración con un gran número de lenguajes tales como PHP, Java, C, C++, etc.

1.7.5. TortoiseSVN

TortoiseSVN es un cliente gratuito de código abierto para el sistema de control de versiones Subversion. TortoiseSVN maneja ficheros y directorios a lo largo del tiempo. Los ficheros se almacenan en un repositorio central. Esto permite que pueda recuperar versiones antiguas de sus ficheros y examinar la historia de cuándo y cómo cambiaron sus datos, y quién hizo el cambio. (17)

Las principales características de TortoiseSVN son:

- ✓ **Integración con el shell de Windows:** TortoiseSVN se integra perfectamente en el shell de Windows (por ejemplo, el explorador). Esto significa que puede seguir trabajando con las herramientas que ya conoce.
- ✓ **Iconos sobreimpresionados:** El estado de cada carpeta y fichero versionado se indica por pequeños iconos sobreimpresionados. De esta forma, puede ver fácilmente el estado en el que se encuentra su copia de trabajo.
- ✓ **Fácil acceso a los comandos de Subversion:** Todos los comandos de Subversion están disponibles desde el menú contextual del explorador. TortoiseSVN añade su propio submenú allí.

1.7.6. Mozilla Firefox

Mozilla Firefox es un navegador de Internet libre y de código abierto. Es usado para visualizar páginas web. Incluye corrector ortográfico, búsqueda progresiva, marcadores dinámicos y un sistema de búsqueda integrado que utiliza el motor de búsqueda que desee el usuario. Además se pueden añadir funciones a través de complementos desarrollados por terceros.

Las características de Mozilla Firefox son las siguientes:

- ✓ Multiplataforma.
- ✓ Cuenta con una protección antiphishing, antimalware e integración con el antivirus.

- ✓ La navegación por pestañas.
- ✓ Bloqueador de ventanas emergentes.
- ✓ Múltiples Extensiones.
- ✓ Incluye de serie un buscador integrado en la interfaz que hace búsquedas en Google.
- ✓ Posee gestor de descargas.
- ✓ Utiliza el sistema SSL para proteger la comunicación con los servidores web, utilizando fuerte criptografía cuando se utiliza el protocolo HTTPS.

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 validaciones, arrojó las siguientes conclusiones:

- Symfony y Zend marco de trabajo proveen los medios para el tratamiento de las validaciones en las aplicaciones desarrolladas sobre estas plataformas, pero carece de una herramienta que permita gestionar la información sobre las validaciones.
- Las herramientas que permiten la gestión de la información sobre las validaciones no se encuentran disponibles para su utilización por ser software propietario, no ajustándose así a las necesidades del CEIGE.

Se utilizarán el Modelo de desarrollo orientado a componentes como guía en el proceso de desarrollo de la aplicación, así como herramientas y tecnologías acorde al resultado que se quiere alcanzar,

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

Capítulo 2: Propuesta de solución

2.1. INTRODUCCIÓN

El siguiente capítulo parte de los conocimientos adquiridos en la fundamentación teórica de la solución, así como algunos de los artefactos generados por el modelo de desarrollo utilizado. Primeramente son descritos los procesos de negocio relativos al campo de acción, luego los escenarios arquitectónicos con el fin de describir los escenarios con sus requisitos asociados, le continua el modelo conceptual con el objetivo de esclarecer los conceptos fundamentales con los que se trabajarán en el desarrollo de la aplicación y seguido los requisitos funcionales y no funcionales de la solución para lograr un entendimiento de lo que se quiere lograr. Posteriormente se realiza una descripción del diseño elaborado por los autores para alcanzar las metas trazadas, además de la estrategia a seguir durante el proceso de implementación.

2.2. DESCRIPCIÓN DE LOS PROCESOS DE NEGOCIO

Un proceso de negocio es un conjunto de procedimientos o actividades relacionadas que logran obtener un objetivo de negocio; en general, dentro del contexto de una estructura organizacional que define roles funcionales. (18)

Un proceso de negocio es un conjunto de actividades estructuradas para lograr un objetivo de la organización, que convierte un conjunto de elementos de entrada en salidas, según las reglas de negocio. (19)

La descripción de los procesos de negocio facilita las actividades del análisis ya que hace posible el entendimiento de los procesos en cuestión para así definir con mayor facilidad los requisitos que cumplirán las necesidades del usuario.

El autor, teniendo en cuenta la opinión de especialistas y de los usuarios finales, identificó el siguiente proceso de negocio que debe ser asistido por la solución. Este proceso es descrito a continuación.

2.2.1. Descripción del proceso: Gestionar validación.

En este proceso de negocio lo que se hace es, según la acción del desarrollador, si la acción es adicionar una validación el desarrollador busca el archivo donde adicionará la validación, luego el desarrollador adiciona la validación o validaciones que desee, escribiendo el código el mismo y termina la acción de adicionar validación. Por otro lado si la acción es modificar una validación, lo primero que hace el desarrollador es buscar el archivo al cual le modificará la validación, luego de abrir el archivo el

desarrollador busca la validación que modificará y después de encontrarla la modifica, escribiendo el código el mismo y termina la acción de modificar validación. De esta manera concluye el proceso de gestionar validación.

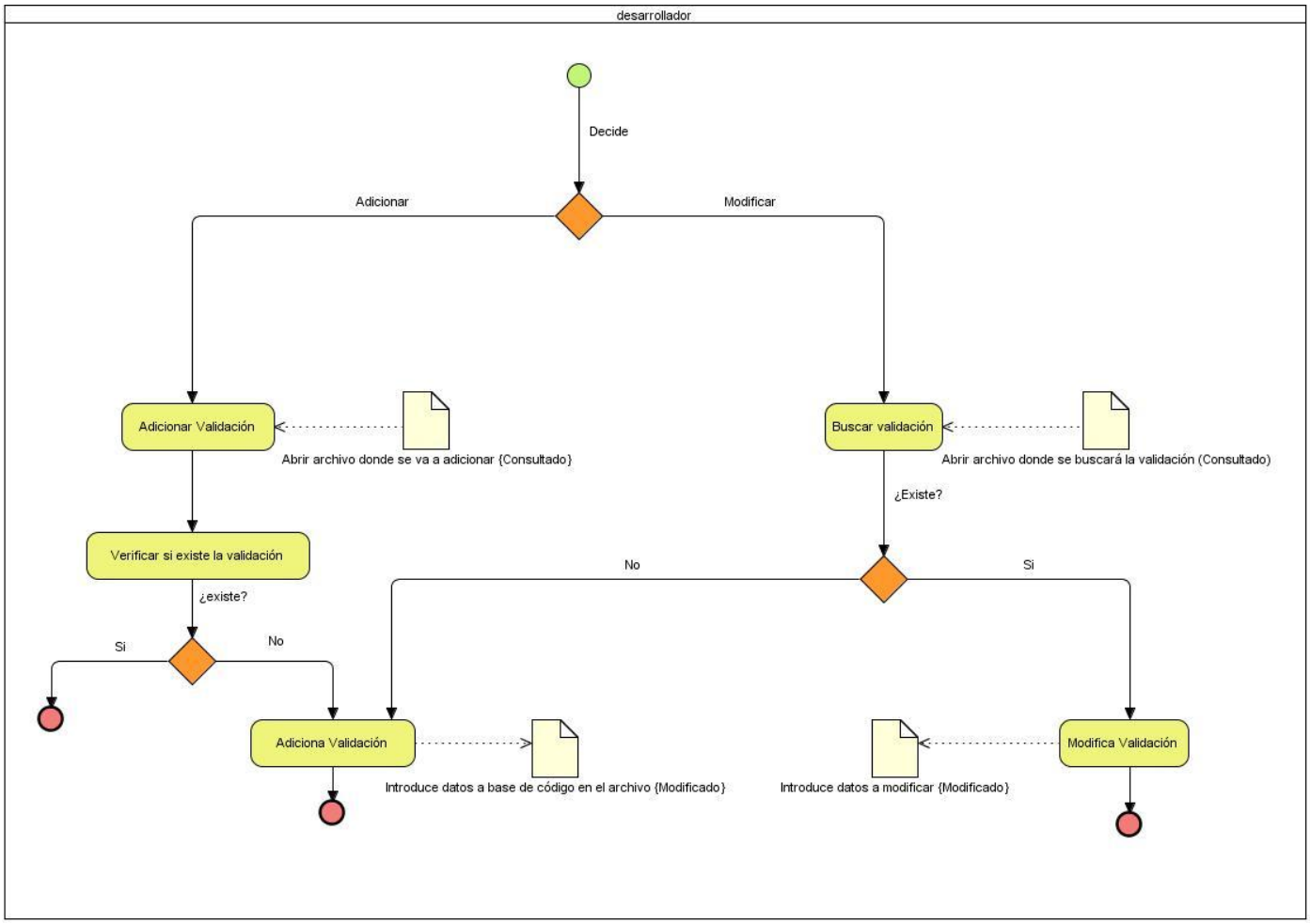

Ilustración 2 Proceso de negocio gestionar validación.

2.3. REQUISITOS DE LA SOLUCIÓN PROPUESTA

Una vez que se realiza el modelado del negocio se puede comenzar a ejecutar el proceso de captura de requisitos del sistema, esta es una de las actividades fundamentales que se desarrolla en el proceso de desarrollo de software. Los requisitos son la condición que tiene que ser alcanzada por un sistema o

componente software para satisfacer un contrato, estándar, u otro documento impuesto formalmente. Los mismos pueden dividirse en requisitos funcionales y requisitos no funcionales.

Los requisitos funcionales son condiciones que el sistema debe cumplir. Los requisitos no funcionales son propiedades que el producto debe tener. A continuación se listan los requisitos que debe cumplir la aplicación a desarrollar.

2.3.1. Requisitos Funcionales

R1 Gestionar validaciones de negocio.

R 1.1 Adicionar validaciones.

R1.2 Modificar validaciones.

R1.3 Eliminar validaciones.

R1.4 Listar validaciones

R1.5 Buscar validaciones.

R2 Gestionar validaciones de datos.

R 2.1 Adicionar validaciones.

R2.2 Modificar validaciones.

R2.3 Eliminar validaciones.

R2.4 Listar validaciones

R2.5 Buscar validaciones.

R3 Gestionar ayuda.

- ✓ **Requisito funcional: Gestionar validaciones de datos.**

Tabla 1 Especificación del requisito Adicionar validaciones de datos.

Conceptos tratados	**Conceptos**	**Atributos**
	Validaciones	• name. • type. • not_null. • null_error. • type_error.
Precondiciones	**Precondiciones**	**Pre-requisitos**
	El usuario se ha autenticado en	Cargar solución.

	el sistema y tiene permisos para ejecutar esta acción.	
	Debe existir al menos un método declarado.	N/A
Descripción	Para adicionar una validación de datos el usuario debe seleccionar el subsistema al cual quiere añadirle la validación e introducir sus datos; debe elegir la clase y dentro de esta el método al cual se le hará la validación, además el usuario tendrá que introducir de manera obligatoria el nombre, el tipo de datos, decir si puede o no ser nulo, en caso de ser nulo deberá introducir entonces el error para ese dato y el error para el tipo de datos. El sistema deberá detectar posibles errores en los datos especificados por el usuario y notificar al mismo para su corrección. En caso de que no existan errores en los datos, el sistema adicionará la validación y mostrará un mensaje de notificación. El sistema debe permitir la cancelación de esta acción.	
Validaciones	No se permite la creación de una validación con el mismo nombre de otro ya existente asignada a un método. No se permiten valores nulos en los datos de carácter obligatorio.	
Post-condiciones	Se ha adicionado una nueva validación al XML.	
Post-requisito	El sistema lista las validaciones.	

Luego de haber realizado una descripción textual del requisito funcional Adicionar validaciones de datos se muestra un prototipo elemental de interfaz gráfica de usuario para dicho requisito, ver Ilustración 4.

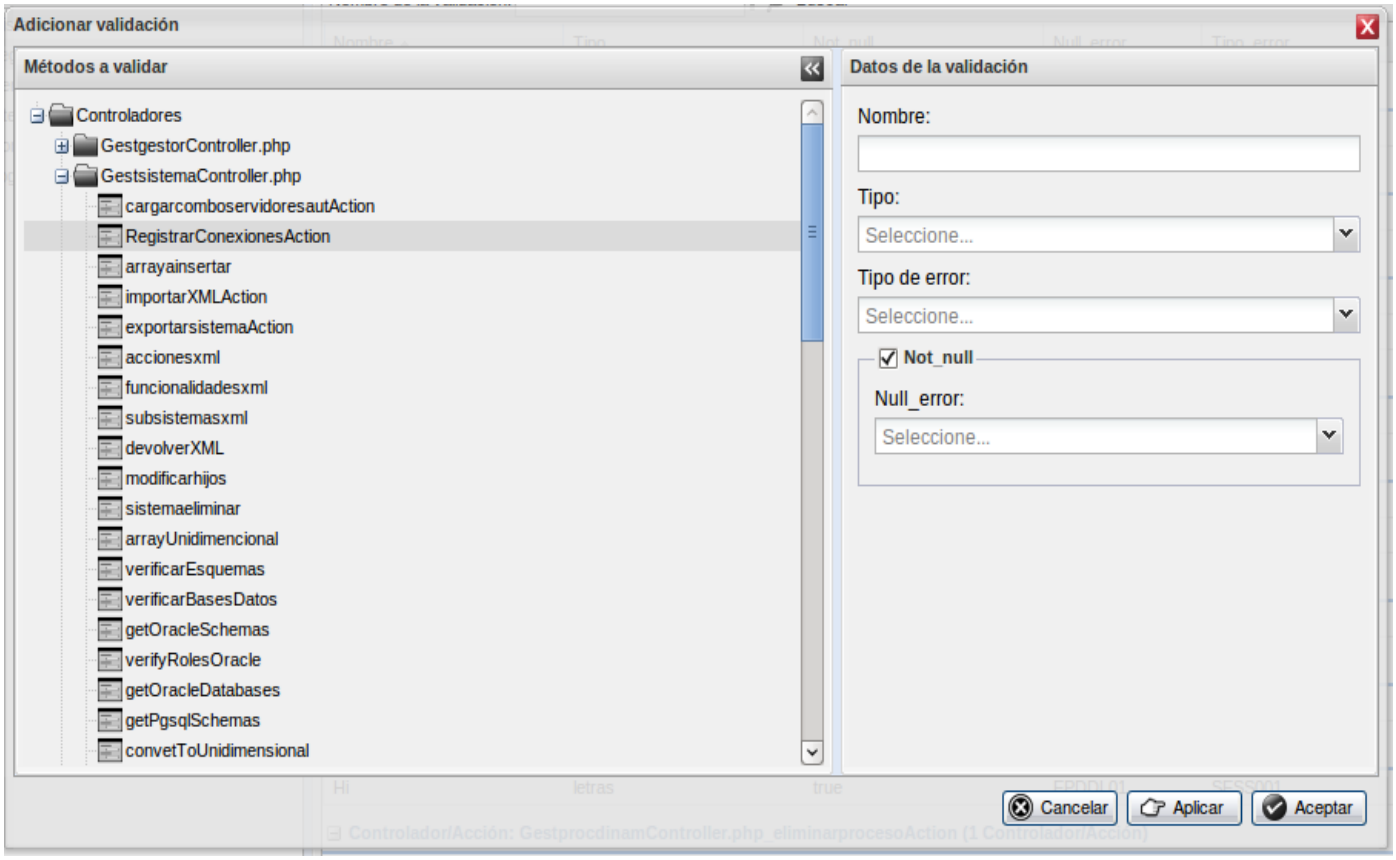

Ilustración 3 Interfaz Adicionar validación de datos

Tabla 2 Especificación del requisito Modificar validaciones de datos.

Conceptos tratados	**Conceptos**	**Atributos**
	Validaciones	• name. • type. • not_null. • null_error. • type_error.
Precondiciones	**Precondiciones**	**Pre-requisitos**
	El usuario se ha autenticado en el sistema y tiene permisos para ejecutar la acción modificar.	Cargar solución.
	Debe existir al menos una	N/A

	validación adicionada	
Descripción	Para modificar una validación de datos el usuario debe seleccionar el subsistema y después de que el sistema liste las validaciones que posee dicho subsistema el usuario deberá seleccionar la validación que desea modificar. Luego debe introducir los datos que desea modificar, excepto el nombre de la validación que será único y una vez que se crea no puede ser modificado. El sistema deberá detectar posibles errores en los datos especificados por el usuario y notificar al mismo para su corrección. En caso de que no existan errores en los datos, el sistema adicionará la validación y mostrará un mensaje de notificación. El sistema debe permitir la cancelación de esta acción.	
Validaciones	No se permiten valores nulos en los datos de carácter obligatorio.	
Post-condiciones	Se ha modificado la validación.	
Post-requisito	El sistema lista las validaciones.	

Luego de haber realizado una descripción textual del requisito funcional Modificar validaciones de datos se muestra un prototipo elemental de interfaz gráfica de usuario para dicho requisito, ver Ilustración 5.

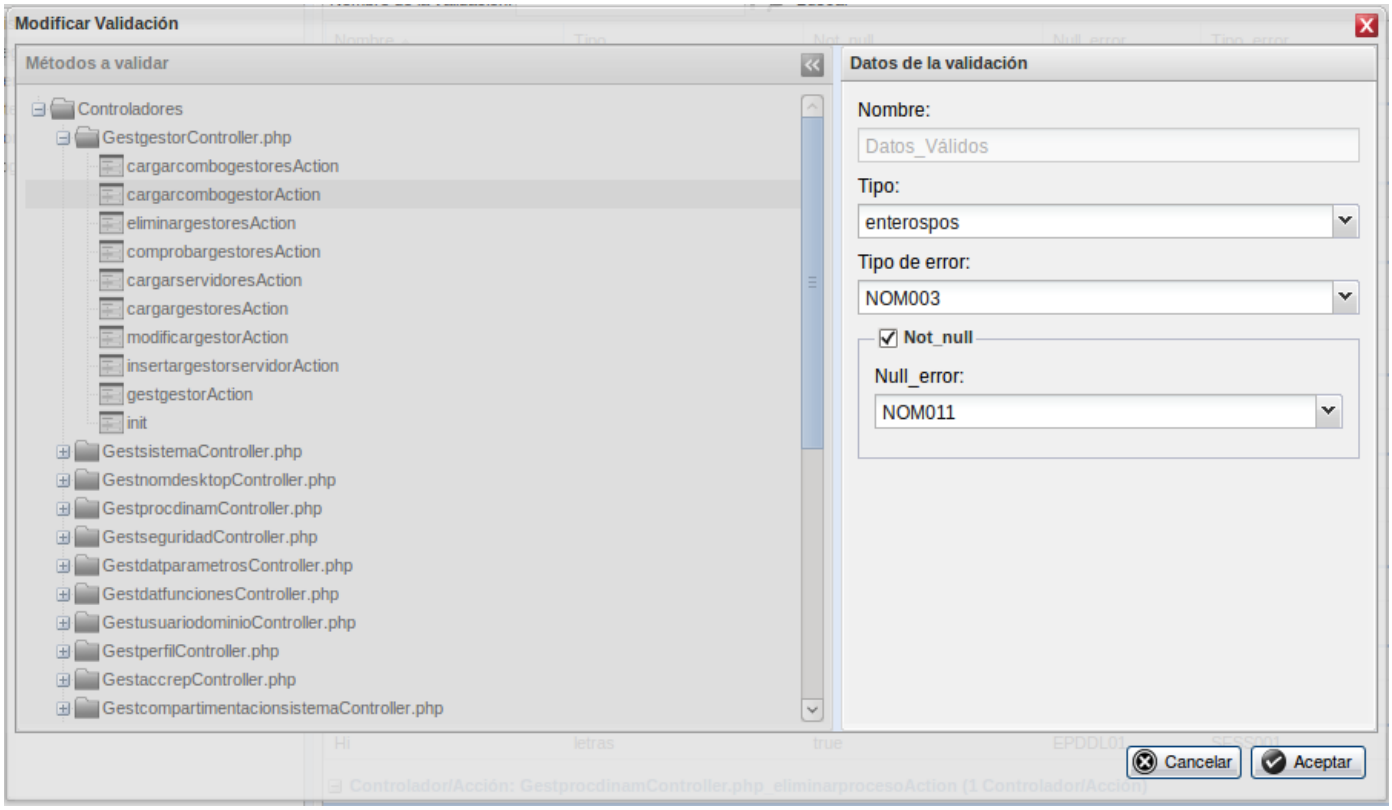

Ilustración 4 Interfaz Modificar validación de datos

Tabla 3 Especificación del requisito Eliminar validaciones de datos.

Conceptos tratados	**Conceptos**	**Atributos**
	Validaciones	
Precondiciones	**Precondiciones**	**Pre-requisitos**
	El usuario se ha autenticado en el sistema y tiene permisos para ejecutar la acción eliminar.	Cargar solución.
	Debe existir al menos una validación para poder eliminarla	N/A
Descripción	Para eliminar una validación de datos el usuario debe seleccionar el subsistema y después de que el sistema liste las validaciones que	

	posee dicho subsistema el usuario deberá seleccionar la validación que desea eliminar, el sistema debe mostrar un mensaje de confirmación. En caso de que el usuario confirme la acción, el sistema elimina la validación y muestra un mensaje de notificación. El sistema debe permitir la cancelación de esta acción.
Validaciones	N/A
Post-condiciones	Se ha eliminado la validación.
Post-requisito	El sistema lista las validaciones.

Luego de haber realizado una descripción textual del requisito funcional Eliminar validaciones de datos se muestra un prototipo elemental de interfaz gráfica de usuario para dicho requisito, ver Ilustración 6.

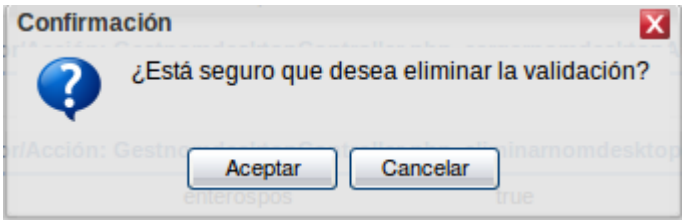

Ilustración 5 Interfaz Eliminar validación de datos

Tabla 4 Especificación del requisito Listar validaciones de datos.

Conceptos tratados	**Conceptos**	**Atributos**
	Validaciones	
Precondiciones	**Precondiciones**	**Pre-requisitos**
	El usuario se ha autenticado en el sistema y tiene permisos para ejecutar la acción listar.	Cargar solución.
	Debe existir al menos un subsistema.	N/A
Descripción	Para listar las validaciones de datos el usuario debe seleccionar el subsistema del que quiere listar sus validaciones. Luego el sistema mostrara las validaciones existentes en ese subsistema.	

Validaciones	N/A
Post-condiciones	Se muestran las validaciones.
Post-requisito	N/A

Luego de haber realizado una descripción textual del requisito funcional Listar validaciones de datos se muestra un prototipo elemental de interfaz gráfica de usuario para dicho requisito, ver Ilustración 7.

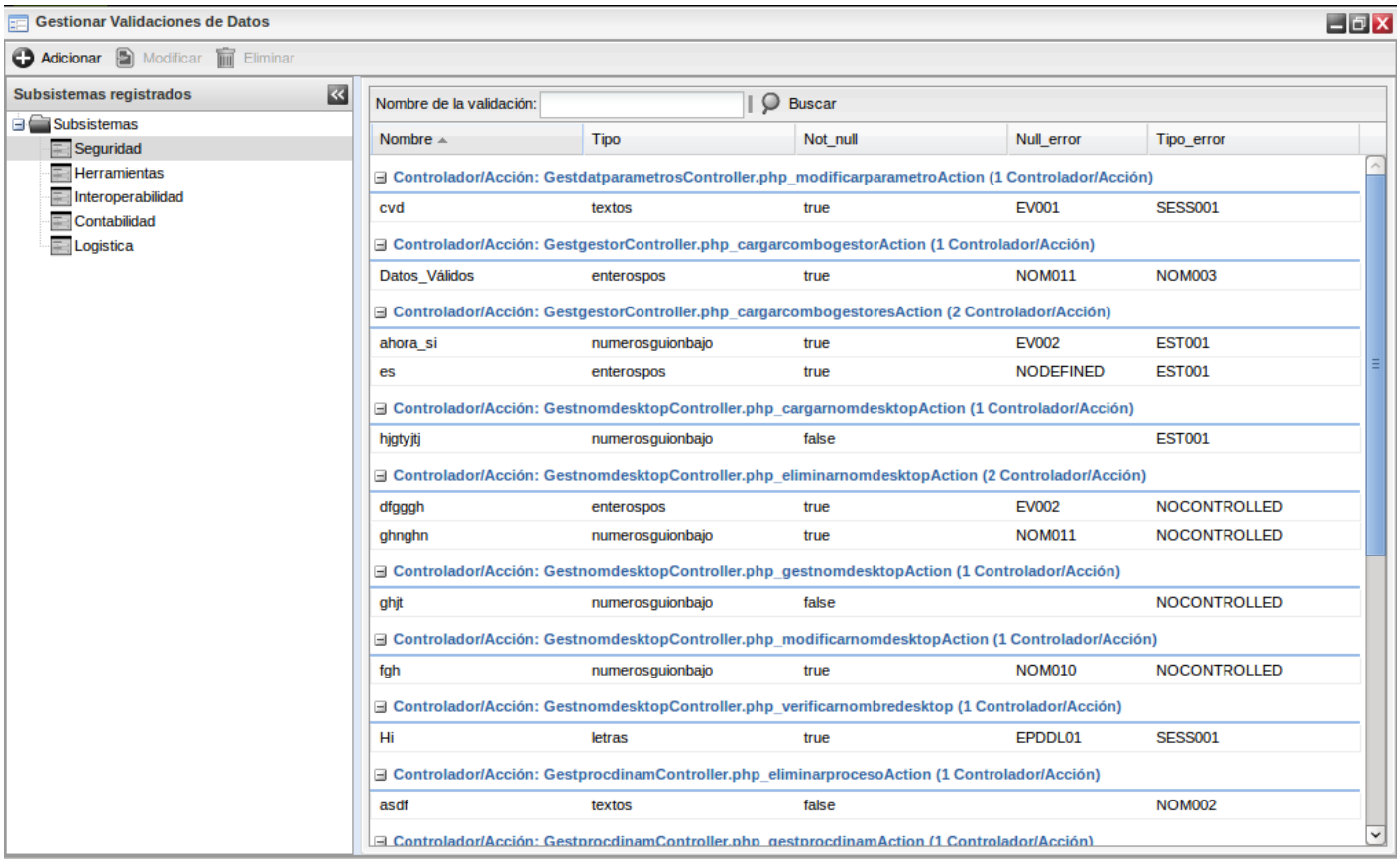

Ilustración 6 Interfaz Listar validación de datos

Tabla 5 Especificación del requisito Buscar validaciones de datos.

Conceptos tratados	**Conceptos**	**Atributos**
	Validaciones	• nombre.
Precondiciones	**Precondiciones**	**Pre-requisitos**

	El usuario se ha autenticado en el sistema y tiene permisos para ejecutar la acción buscar.	Cargar solución.
Descripción	Para buscar una validación de datos el usuario debe seleccionar el subsistema y después de que el sistema liste las validaciones que posee dicho subsistema el usuario deberá introducir el criterio de búsqueda El sistema deberá detectar posibles errores en los datos especificados por el usuario y notificar al mismo para su corrección. En caso de que no existan errores en los datos, el sistema buscara todas las validaciones que tengan ese nombre.	
Validaciones	No se permiten valores nulos en los datos del criterio de búsqueda.	
Post-condiciones	Se muestran la o las validaciones.	
Post-requisito	El sistema lista las validaciones.	

Luego de haber realizado una descripción textual del requisito funcional Buscar validaciones de datos se muestra un prototipo elemental de interfaz gráfica de usuario para dicho requisito, ver Ilustración 8.

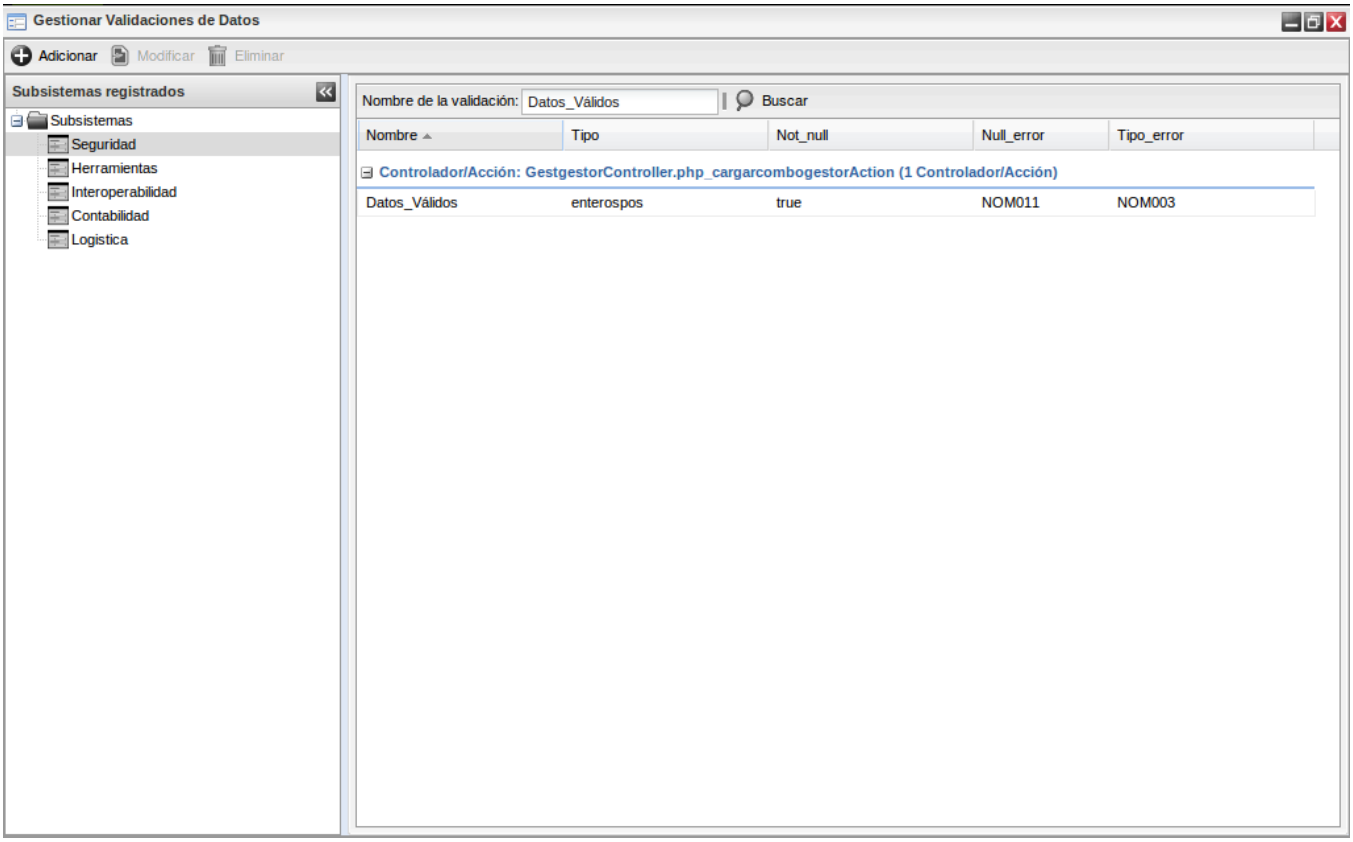

Ilustración 7 Interfaz Buscar validación de datos

- ✓ **Requisito funcional: Gestionar validaciones de negocio.**

Tabla 6 Especificación del requisito Adicionar validaciones de negocio.

Conceptos tratados	**Conceptos**	**Atributos**
	Validaciones	• class. • method. • error.
Precondiciones	**Precondiciones**	**Pre-requisitos**
	El usuario se ha autenticado en el sistema y tiene permisos para ejecutar esta acción.	Cargar solución.
	Debe existir al menos un método declarado.	N/A

Descripción	Para adicionar una validación de negocio el usuario debe seleccionar el subsistema al cual quiere añadirle la validación e introducir sus datos; debe elegir la clase y dentro de esta el método al cual se le hará la validación, luego deberá seleccionar la clase y dentro de esta clase el método que implementara la validación, además el usuario tendrá que introducir de manera obligatoria el nombre y el error, también deberá introducir la descripción de la validación la cual será de carácter opcional. El sistema deberá detectar posibles errores en los datos especificados por el usuario y notificar al mismo para su corrección. En caso de que no existan errores en los datos, el sistema adicionará la validación y mostrará un mensaje de notificación. El sistema debe permitir la cancelación de esta acción.
Validaciones	No se permite la creación de una validación con el mismo nombre de otro ya existente asignada a un método. No se permiten valores nulos en los datos de carácter obligatorio.
Post-condiciones	Se ha adicionado una nueva validación al XML.
Post-requisito	El sistema lista las validaciones.

Luego de haber realizado una descripción textual del requisito funcional Adicionar validaciones de negocio se muestra un prototipo elemental de interfaz gráfica de usuario para dicho requisito, ver Anexo 1.

Tabla 7 Especificación del requisito Modificar validaciones de negocio.

Conceptos tratados	**Conceptos**	**Atributos**
	Validaciones	• class. • method. • error.
Precondiciones	**Precondiciones**	**Pre-requisitos**
	El usuario se ha autenticado en el sistema y tiene permisos para ejecutar la acción modificar.	Cargar solución.

	Debe existir al menos una validación adicionada	N/A
Descripción	Para modificar una validación de negocio el usuario debe seleccionar el subsistema y después de que el sistema liste las validaciones que posee dicho subsistema el usuario deberá seleccionar la validación que desea modificar. Luego debe introducir los datos que desea modificar, excepto el nombre de la validación que será único y una vez que se crea no puede ser modificado. El sistema deberá detectar posibles errores en los datos especificados por el usuario y notificar al mismo para su corrección. En caso de que no existan errores en los datos, el sistema adicionará la validación y mostrará un mensaje de notificación. El sistema debe permitir la cancelación de esta acción.	
Validaciones	No se permiten valores nulos en los datos de carácter obligatorio.	
Post-condiciones	Se ha modificado la validación.	
Post-requisito	El sistema lista las validaciones.	

Luego de haber realizado una descripción textual del requisito funcional Modificar validaciones de negocio se muestra un prototipo elemental de interfaz gráfica de usuario para dicho requisito, ver Anexo 2.

Tabla 8 Especificación del requisito Eliminar validaciones de negocio.

Conceptos tratados	**Conceptos**	**Atributos**
	Validaciones	
Precondiciones	**Precondiciones**	**Pre-requisitos**
	El usuario se ha autenticado en el sistema y tiene permisos para ejecutar la acción eliminar.	Cargar solución.
	Debe existir al menos una validación para poder eliminarla	N/A
Descripción	Para eliminar una validación de negocio el usuario debe seleccionar el subsistema y después de que el sistema liste las validaciones que	

	posee dicho subsistema el usuario deberá seleccionar la validación que desea eliminar, el sistema debe mostrar un mensaje de confirmación. En caso de que el usuario confirme la acción, el sistema elimina la validación y muestra un mensaje de notificación. El sistema debe permitir la cancelación de esta acción.
Validaciones	N/A
Post-condiciones	Se ha eliminado la validación.
Post-requisito	El sistema lista las validaciones.

Luego de haber realizado una descripción textual del requisito funcional Eliminar validaciones de negocio se muestra un prototipo elemental de interfaz gráfica de usuario para dicho requisito, ver Anexo 3.

Tabla 9 Especificación del requisito Listar validaciones de negocio.

Conceptos tratados	**Conceptos**	**Atributos**
	Validaciones	
Precondiciones	**Precondiciones**	**Pre-requisitos**
	El usuario se ha autenticado en el sistema y tiene permisos para ejecutar la acción listar.	Cargar solución.
	Debe existir al menos un subsistema.	N/A
Descripción	Para listar las validaciones de negocio el usuario debe seleccionar el subsistema del que quiere listar sus validaciones Luego el sistema mostrara las validaciones existentes en ese subsistema.	
Validaciones	N/A	
Post-condiciones	Se muestran las validaciones.	
Post-requisito	N/A	

Luego de haber realizado una descripción textual del requisito funcional Listar validaciones de negocio se muestra un prototipo elemental de interfaz gráfica de usuario para dicho requisito, ver Anexo 4.

Tabla 10 Especificación del requisito Buscar validaciones de negocio.

Conceptos tratados	**Conceptos**	**Atributos**
	Validaciones	• denominación.
Precondiciones	**Precondiciones**	**Pre-requisitos**
	El usuario se ha autenticado en el sistema y tiene permisos para ejecutar la acción buscar.	Cargar solución.
Descripción	Para buscar una validación de negocio el usuario debe seleccionar el subsistema y después de que el sistema liste las validaciones que posee dicho subsistema el usuario deberá introducir el criterio de búsqueda El sistema deberá detectar posibles errores en los datos especificados por el usuario y notificar al mismo para su corrección. En caso de que no existan errores en los datos, el sistema buscara todas las validaciones que tengan esa denominación.	
Validaciones	No se permiten valores nulos en los datos del criterio de búsqueda.	
Post-condiciones	Se muestran la o las validaciones.	
Post-requisito	El sistema lista las validaciones.	

Luego de haber realizado una descripción textual del requisito funcional Buscar validaciones de negocio se muestra un prototipo elemental de interfaz gráfica de usuario para dicho requisito, ver Anexo 5.

2.3.2. Requisitos no Funcionales

Son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Existen múltiples categorías para clasificar a los requerimientos no funcionales, siendo las siguientes representativas de un conjunto de aspectos que se deben tener en cuenta, aunque no limitan a la definición de otros.

El presente trabajo forma parte de un proceso productivo iniciado por el CEIGE y los resultados que se obtengan formarán parte del marco de trabajo Sauxe. Por tanto los requisitos no funcionales con los que

debe cumplir la aplicación a desarrollar fueron establecidos por el CEIGE al inicio del proceso de desarrollo, a continuación se describen los más importantes.

Seguridad

Autenticación y Autorización (Contraseña de acceso). Protección contra maniobras no autorizadas o que puedan afectar la integridad de los datos. La vigilancia al sistema así como la salva de la información, se realizará de forma centralizada por el administrador, además el sistema debe mostrar opción de advertencia antes de borrar cualquier elemento o información que pueda existir.

Software

Para el cliente:

- ✓ Navegador Mozilla Firefox 2.2 o superior.
- ✓ Sistema operativo Windows XP 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 servidor:

- ✓ Requerimientos mínimos: Procesador Pentium IV a 2GHz de velocidad de procesamiento y 1Gb de memoria RAM.
- ✓ Al menos 50Gb de espacio libre en disco duro.
- ✓ Tarjeta de red.

Para el cliente:

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

Confiabilidad

El subsistema debe ser confiable y preciso en la información que le suministra al usuario para evitar cualquier tipo de error. Estará disponible todo el tiempo, permitiendo el trabajo a los usuarios y las acciones de mantenimiento. Este debe ser estable, fiable y la velocidad de respuesta debe ser rápida durante la utilización del mismo. La información almacenada debe ser confiable en cuanto a su veracidad e integridad desde su recopilación.

Portabilidad

El subsistema será multiplataforma (Linux-Windows) lo que permitirá ejecutarse sobre diferentes sistemas operativos sin importar sus versiones, y sin necesidad de modificar su código fuente.

2.4. MODELO CONCEPTUAL

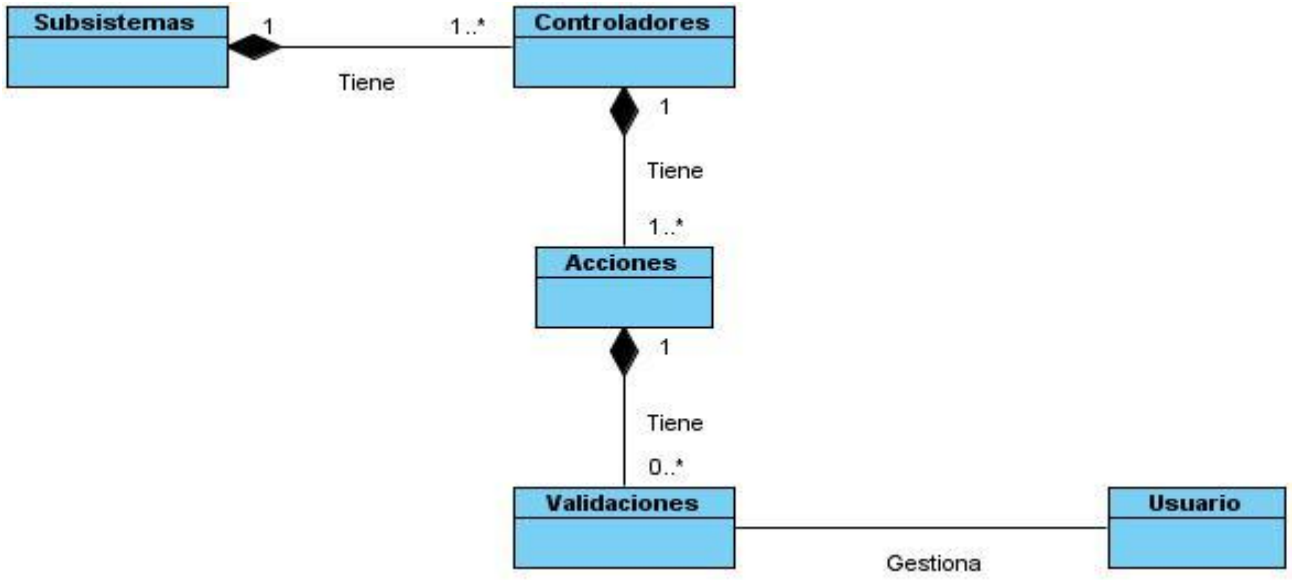

Ilustración 8 Modelo conceptual

2.5. MODELO DE DISEÑO

El Modelo de diseño es una abstracción del Modelo de Implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño. Es utilizado como entrada esencial en las actividades relacionadas a la implementación. El Modelo de Diseño puede contener: diagramas, clases, paquetes, subsistemas, cápsulas, protocolos, interfaces, relaciones, colaboraciones y atributos.

Dentro de este epígrafe se verán los elementos que se tuvieron en cuenta durante el diseño de la solución, dando paso así a la exposición de los resultados de este flujo de trabajo, que refleja cómo será implementado el sistema en términos de clases del diseño.

2.5.1. Diagrama de clases del diseño.

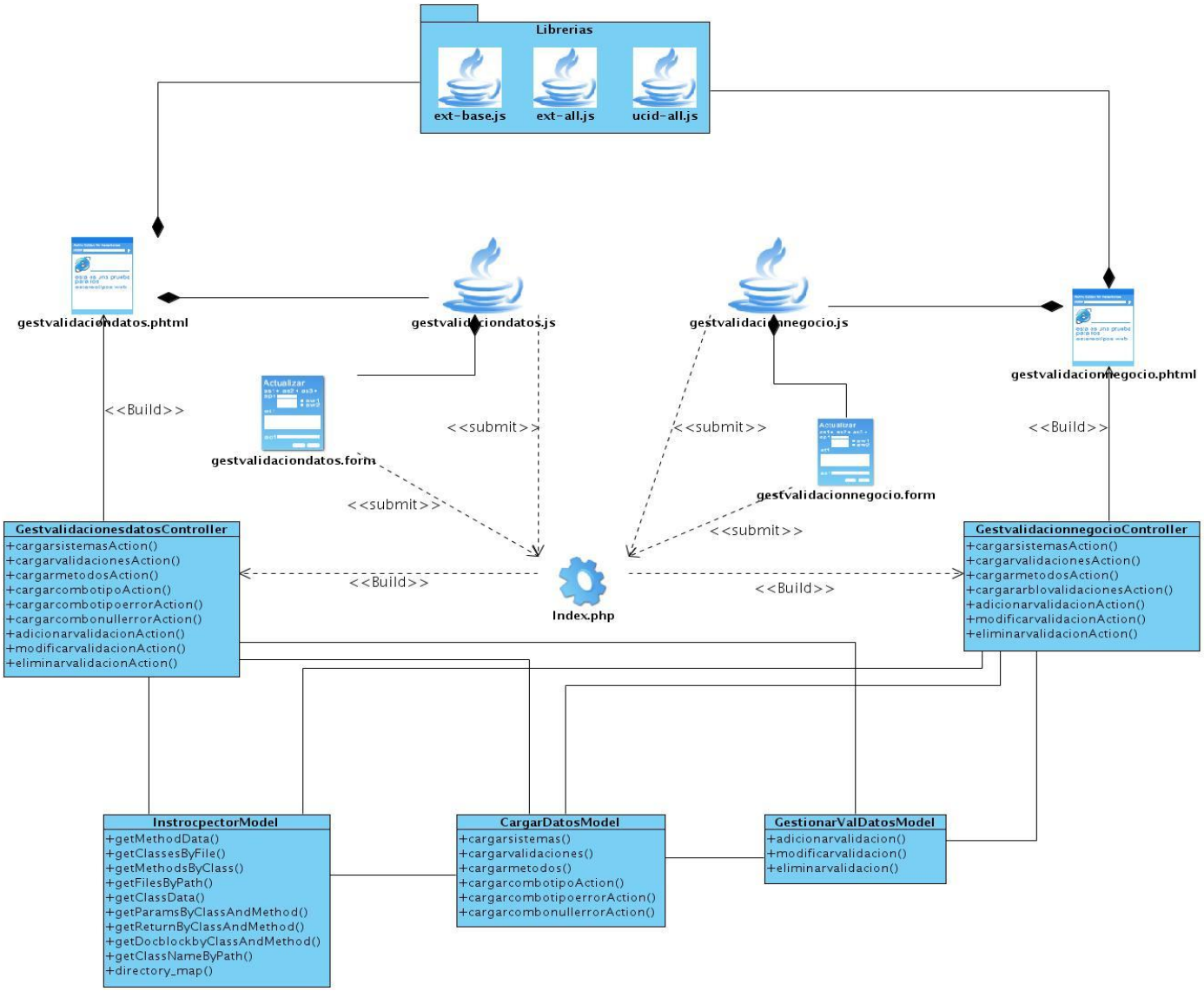

Ilustración 9 Diagrama de clases del diseño.

El diagrama muestra como las páginas clientes a través de los formularios gestvalidacionnegocio y gestvalidaciondatos envían los datos a las clases controladoras, las clases controladoras

GestvalidacionnegocioController y GestvalidaciondatosController son las encargadas de capturar la información proveniente de las vistas y envían esta información a las clases del modelo InstrocpectorModel, CargarDatosModel y GestionarValidacionModel que son las que realizan toda la lógica del componente.

2.5.2. Modelo de datos.

Como modelo de datos, aunque no se utilice base de datos, se tiene un xml para persistir los datos de las validaciones, para que cada vez que algún desarrollador necesite configurar alguna validación, a través de la herramienta a implementar pueda acceder con mayor facilidad a este XML.

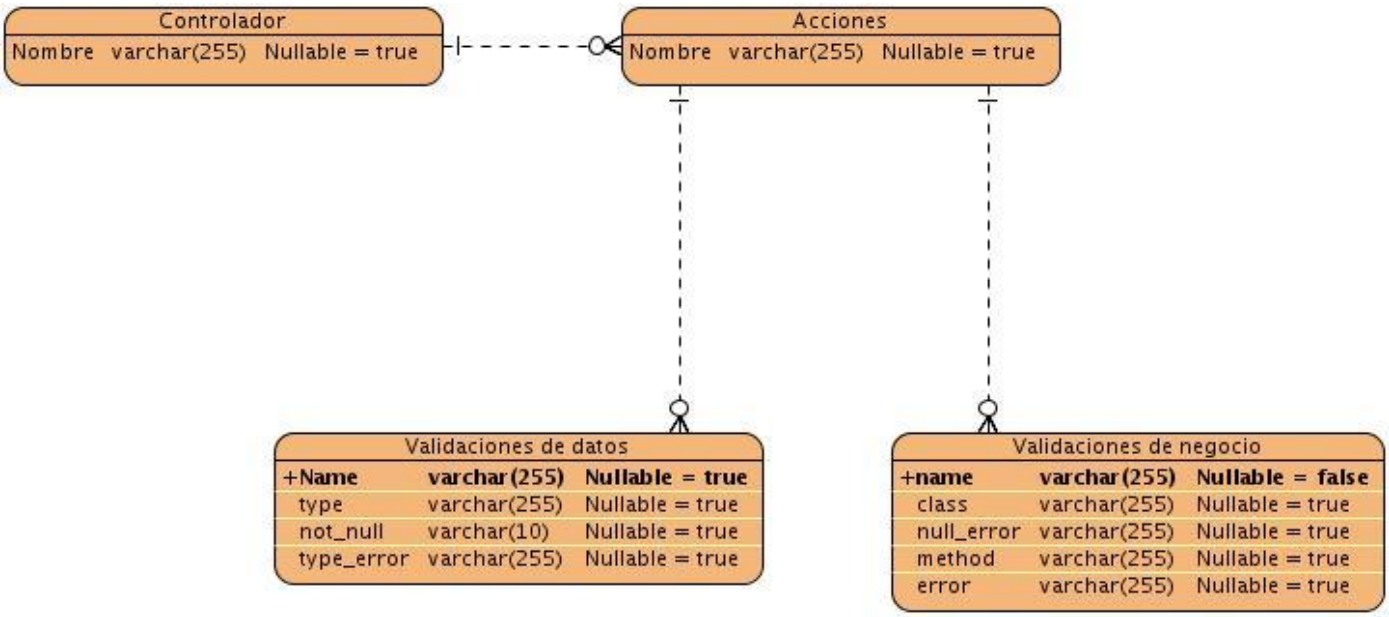

Ilustración 10 Modelo de datos.

2.5.3. Patrones utilizados.

2.5.3.1. Patrón Arquitectónico modelo- vista- controlador

El Modelo Vista Controlador (Model View Controller, MVC por sus siglas en inglés) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la

vista es la interface de usuario y el código es el que provee de datos dinámicos a la página; el modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio; y el controlador es el responsable de recibir los eventos de entrada desde la vista.

- ✓ **Modelo**: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no permitiendo comprar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o importes en un carrito de la compra.

- ✓ **Vista**: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

- ✓ **Controlador**: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. (20)

Este patrón se ve reflejado en el diagrama de clases del diseño de la aplicación, el cual tiene tres clases model, IntrospectorModel, CargarDatosModel y GestionarValModel que son las que implementan la lógica de negocio y el acceso a datos , en el caso de esta aplicación a desarrollar, el acceso al XML donde se persisten los datos de las validaciones, también tiene 2 clases controllers, GestValidaciondatosController y GestValidacionnegocioController que son las que controlan como lo dice su nombre, todas las peticiones hechas por el usuario a las clases model a través de las clases interfaces que se encuentran referenciadas en las clases gestvalidaciondatos.phtml y gestvalidacionnegocio.phtml, estas clases son gestvalidaciondatos.js y gestvalidacionnegocio.js que son las que implementan el diseño de las interfaces de usuario.

2.5.3.2. Patrones de diseño.

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software (21).

Patrones de diseño Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan (22).

Christopher Alexander a plasmado que ”*cada patrón describe un problema que ocurre una y otra vez en nuestro ambiente, y luego describe el núcleo de la solución a ese problema, de forma tal que esa solución puede ser usada un millón de veces, sin hacerlo de la misma manera dos veces.* ” (23)

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.

En el desarrollo del componente se implementan varios patrones de diseño. A continuación se especifican los que fueron utilizados dentro de la solución:

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.

Observer (Observador): clasifica como patrón de comportamiento a nivel de objeto, el problema que lo motiva es la existencia de diferentes objetos desacoplados que deben mantenerse actualizados del estado de otro objeto, o de los determinados sucesos que ocurren en él, su propósito es garantizar que los interesados en el estado del objeto observado sean notificados convenientemente. (23)

Es ampliamente utilizado en el diseño de componentes de interface de usuario de la capa de presentación.

2.6. CONCLUSIONES PARCIALES

Durante el desarrollo del capítulo se argumentaron los aspectos fundamentales que se llevaron a cabo durante el proceso de análisis y diseño de la herramienta para la configuración visual de las validaciones en el marco de trabajo Sauxe. Fueron expuestos los principales artefactos que propone el modelo de desarrollo orientado a componentes. En este capítulo se han generado los diagramas de procesos, el modelo conceptual con los conceptos más importantes de la herramienta que se desea desarrollar. Se describió la solución propuesta mediante los requisitos funcionales para tener una idea más detallada de lo que se quiere implementar, los requisitos no funcionales para garantizar una ejecución eficaz de la aplicación y el diagrama de clases compuesto por las clases del diseño y sus relaciones. Los objetivos

planteados para la realización de este capítulo han sido cumplidos, ya que resulta en el análisis y diseño de un componente para la configuración visual de las validaciones.

Una vez concluido el análisis y diseño de la solución puede darse paso a los flujos de implementación y prueba de la misma.

Capítulo 3: Implementación y Prueba

3.1. INTRODUCCION

El trabajo en este capítulo parte de los resultados obtenidos en el análisis y diseño de la solución propuesta, se mostrará el modelo de implementación que pone en práctica el diseño de la solución realizado en el capítulo anterior con el objetivo de definir la estructura y organización de la aplicación y se especifica el conjunto de validaciones y pruebas que evalúan la calidad de la herramienta.

3.2. 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 en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros, además de los recursos necesarios para poder ejecutar la herramienta desarrollada.

3.2.1. Diagrama de componentes

La solución propuesta consta de un solo componente llamado configuración visual de validaciones el cual interactúa con otros componentes del marco de trabajo como ZendExt, Zend marco de trabajo, Doctrine, Extjs. A continuación se muestra el diagrama de componentes elaborado.

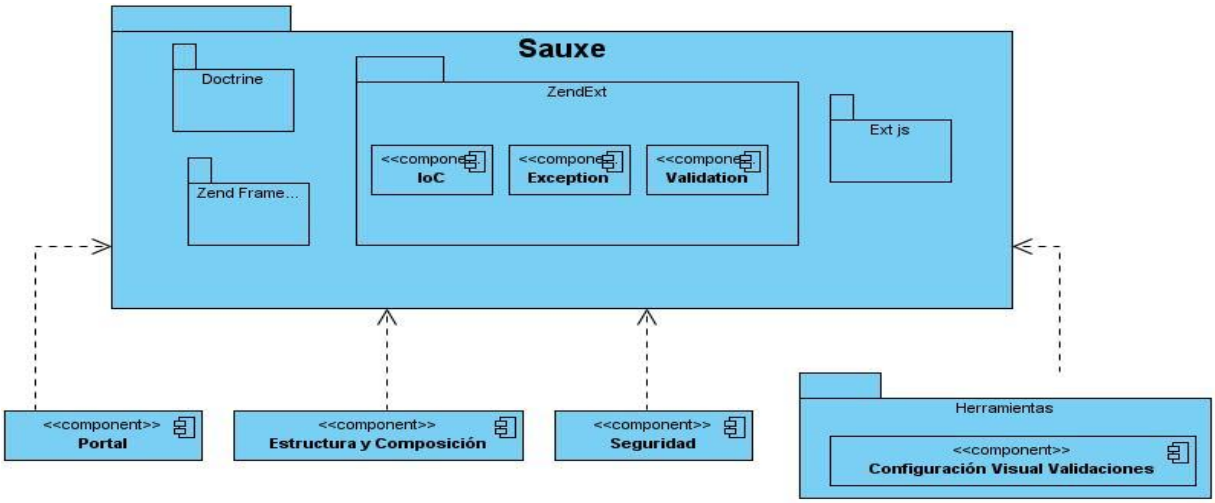

Ilustración 11 Diagrama de componentes

3.2.2. Diagrama de despliegue.

El usuario, desde una estalación de trabajo, podrá acceder al sistema el cual estará desplegado en el mismo servidor web donde se encuentre ubicada la aplicación a la cual se le desee generar servicios web. Dicho servidor estará conectado a un servidor de bases de datos en el cual se almacenará la información de interés para la solución. A continuación se muestra el diagrama de despliegue elaborado.

Ilustración 12 Diagrama de despliegue.

3.2.3. Estándares de codificación.

Los estándares de codificación son reglamentos de la programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para 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.

A continuación se describen algunos de estos estándares empleados durante la implementación de la herramienta de configuración visual de las validaciones en el marco de trabajo Sauxe.

3.2.3.1. CamelCasing

El estándar CamelCasing es parecido al PascalCasing con la particularidad de que la letra inicial del identificador no comienza con mayúscula. Esta notación se utilizó para el nombre de funciones y atributos.

- ✓ **Nomenclatura de las funciones**

El identificativo a emplear para las funciones o métodos se escribe con la primera palabra en minúscula utilizando la notación ***CamelCasing*** y nombres que deduzcan su propósito. Ejemplo: cargarMetodos. Los denominadores de las acciones de las clases controladoras tienen la peculiaridad de ir seguidos por la palabra “Action”. Ejemplo: cargarMetodosAction.

- ✓ **Nomenclatura de los atributos**

Las variables se nombran convenientemente de acuerdo con el estándar CamelCasing. Ejemplo: idsubsistema y nombreclase.

3.2.3.2. Notación húngara

Esta convención, también conocida como notación: **REDDICK** por el nombre de su creador, se basa en definir prefijos para cada tipo de datos según el ámbito de las variables con el fin de brindar mayor información al nombre de la variable, método o función. La notación húngara fue utilizada para la definición de variables de acuerdo con los siguientes prefijos:

Tabla 11 Prefijos para la creación de variables.

Tipo de Datos	Prefijos
Arreglos	**arr**
Objetos	**obj**

Enteros	**int**
Cadena	**str**
Float	**fit**
Boolean	**bool**

- ✓ **Nomenclatura de las variables**

El nombre a emplear para las variables se escribe siguiendo la notación CamelCasing comenzando con el prefijo según su tipo de datos. **Ejemplo:** arrvalidaciones define un arreglo de validaciones.

- ✓ **Nomenclatura de las constantes**

La definición de las constantes se realiza utilizando todas las letras en mayúscula. **Ejemplo:** SUBSISTEMA.

3.2.4. Métricas de diseño.

Las métricas se centran en cuantificar tanto la complejidad, como la funcionalidad y eficiencia inmersa en el desarrollo de software, inclinando sus objetivos a mejorar la comprensión de la calidad del producto, estimar efectividad del proceso y mejorar la calidad del trabajo. Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad:

- ✓ **Responsabilidad:** Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- ✓ **Complejidad de implementación:** Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- ✓ **Reutilización:** Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- ✓ **Acoplamiento:** Consiste en 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:** Consiste en 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:** Consiste en 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 escogidas como instrumento para evaluar la calidad del diseño descrito en el capítulo anterior 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 12 Atributos de calidad evaluados por la métrica TOC.

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

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

Tabla 13 Criterios de evaluación para la métrica TOC.

Atributo	Categoría	Criterio
Responsabilidad	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Complejidad implementación	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
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 14 Atributos de calidad evaluados por la métrica RC.

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

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

Tabla 15 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

Resultados obtenidos de la aplicación de la métrica TOC

Una vez obtenidos los resultados de la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 80% de las clases empleadas en el sistema posee 6 operaciones o mas lo que conlleva a evaluaciones positivas de los atributos de calidad involucrados (responsabilidad, complejidad de implementación y reutilización).
A continuación se muestran los resultados obtenidos.

Tabla 16 Instrumento de evaluación de la métrica TOC.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
GestvalidacionnegocioController	7	Baja	Baja	Alta
GestvalidaciondatosController	9	Media	Media	Media
InstrospectorModel	10	Media	Media	Media
CargarDatosModel	6	Baja	Baja	Alta
GestionarValDatosModel	3	Baja	Baja	Alta

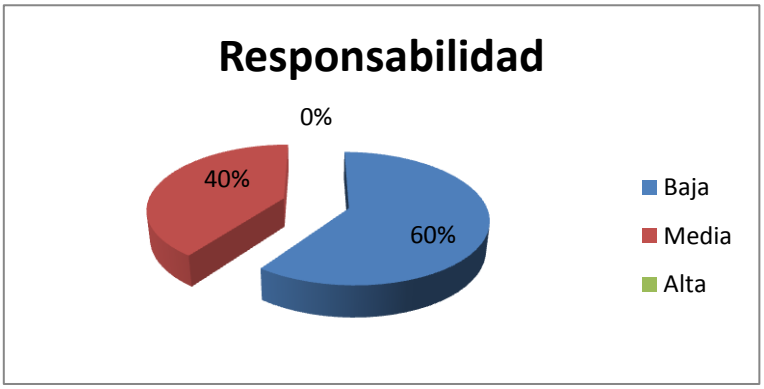

Ilustración 13 Resultados de la evaluación de la métrica TOC para el atributo responsabilidad.

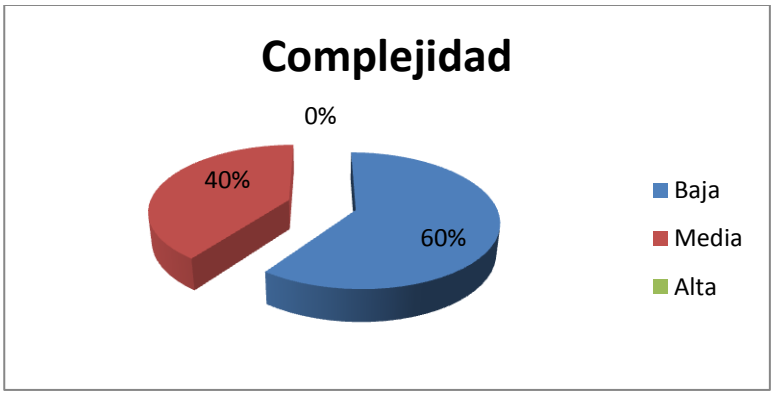

Ilustración 14 Resultados de la evaluación de la métrica TOC para el atributo complejidad.

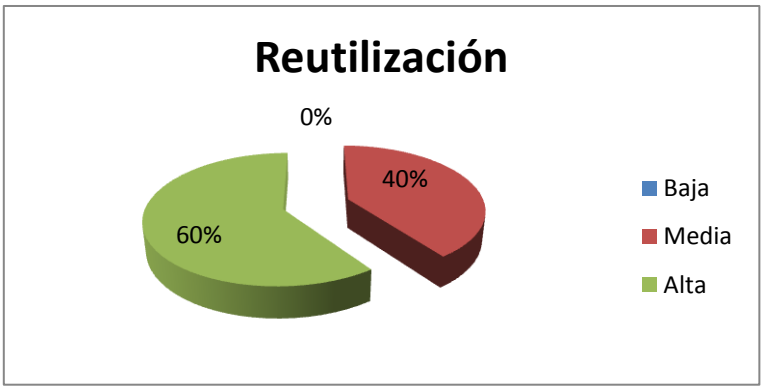

Ilustración 15 Resultados de la evaluación de la métrica TOC para el atributo reutilización.

Resultados obtenidos de la aplicación de la métrica RC

Una vez obtenidos los resultados de la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 60% de las clases empleadas posee menos de 3 dependencias de otras clases lo que conlleva a evaluaciones positivas de los atributos de calidad involucrados (acoplamiento, complejidad de mantenimiento, cantidad de pruebas y reutilización).

A continuación se muestran los resultados obtenidos.

Tabla 17 Instrumento de evaluación de la métrica RC.

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
GestvalidacionnegocioController	3	Alta	Media	Media	Media
GestvalidaciondatosController	3	Alta	Media	Media	Media
InstrospectorModel	1	Baja	Baja	Alta	Baja
CargarDatosModel	1	Baja	Baja	Alta	Baja
GestionarValDatosModel	1	Baja	Baja	Alta	Baja

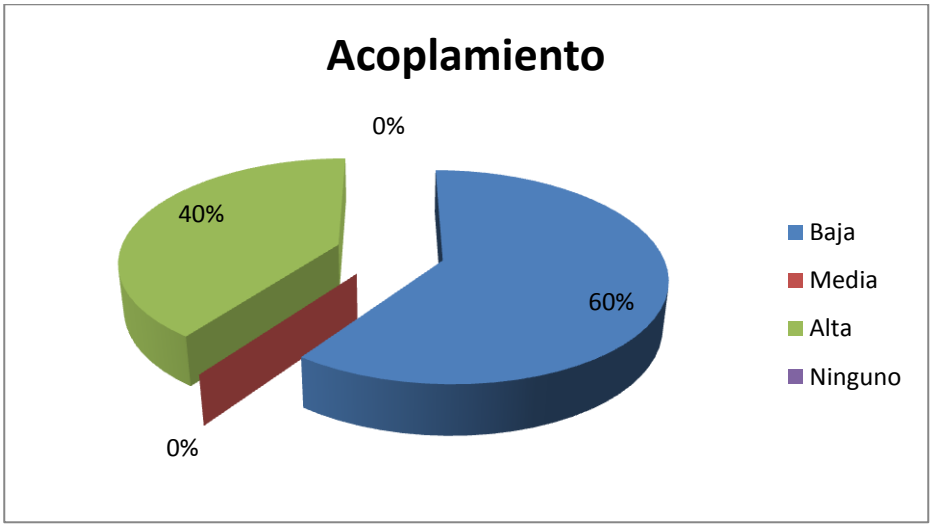

Ilustración 16 Resultados de la evaluación de la métrica RC para el atributo acoplamiento.

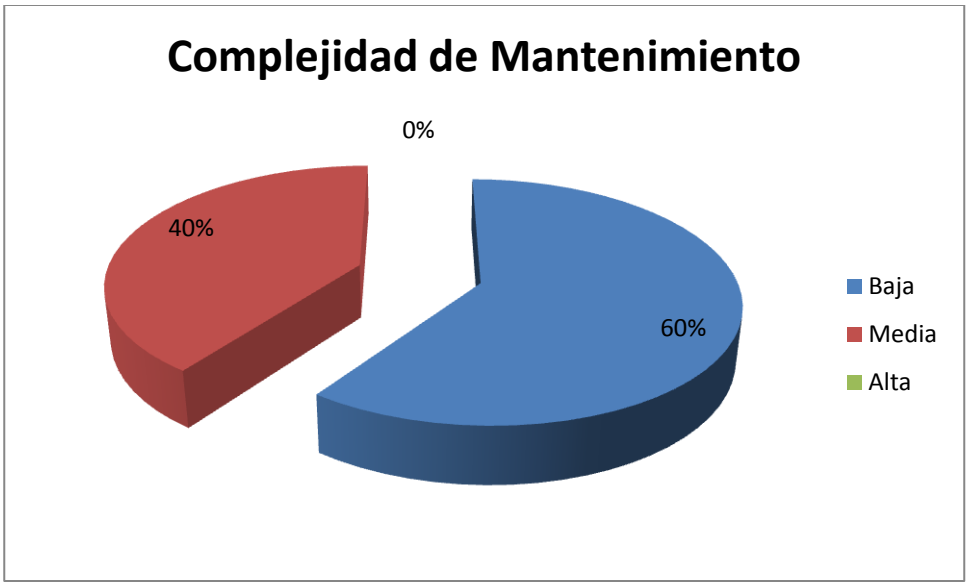

Ilustración 17 Resultados de la evaluación de la métrica RC para el atributo complejidad de mantenimiento.

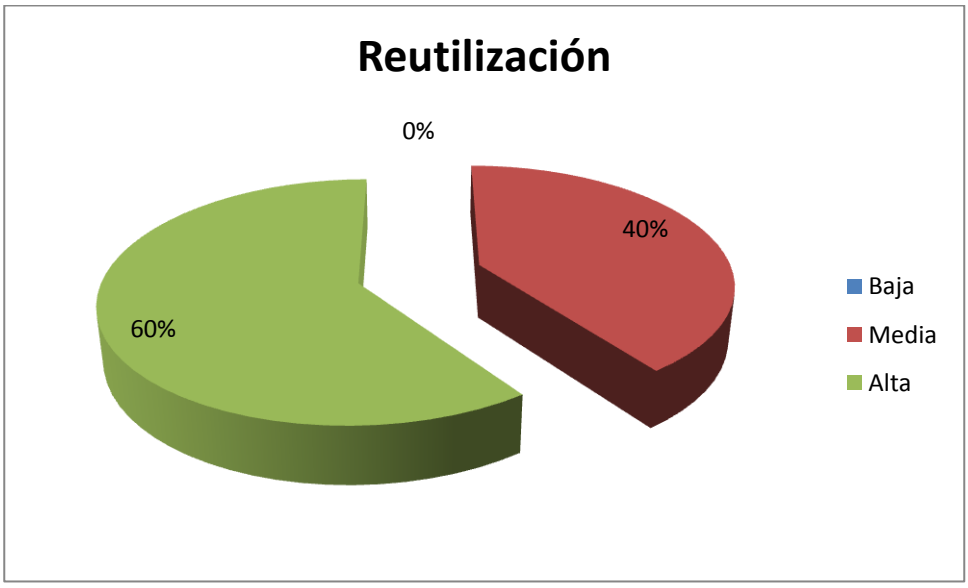

Ilustración 18 Resultados de la evaluación de la métrica RC para el atributo reutilización.

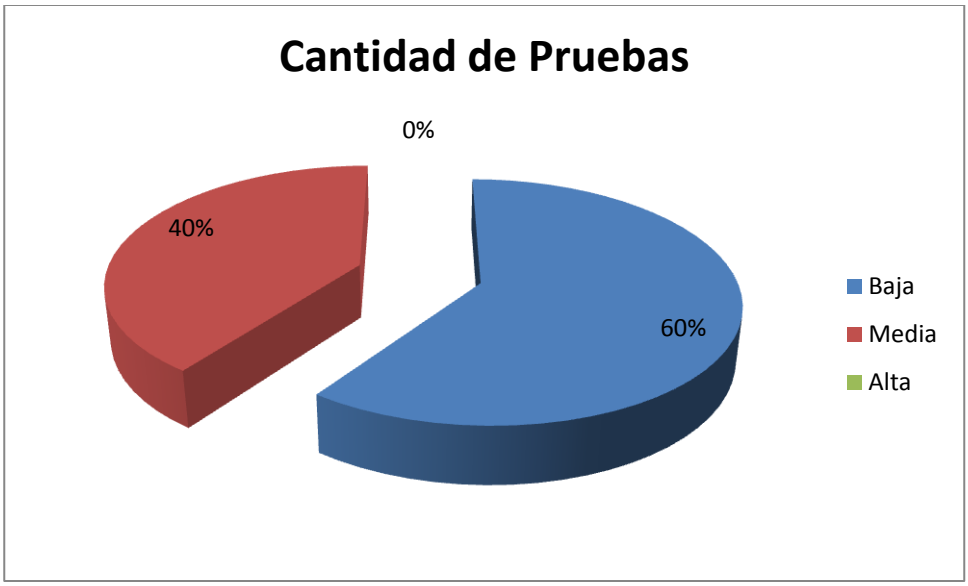

Ilustración 19 Resultados de la evaluación de la métrica RC para el atributo cantidad de pruebas.

Matriz de inferencia de indicadores de calidad

La matriz inferencia de indicadores de calidad, también llamada matriz de cubrimiento, es una representación estructurada de los atributos de calidad y métricas utilizadas para evaluar la calidad del diseño propuesto. Dicha matriz permite conocer si los resultados obtenidos de las relaciones atributo/métrica es positivo o no, llevando estos resultados a una escalabilidad numérica donde, 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.

A continuación se muestran los resultados obtenidos.

Tabla 18 Resultados de la evaluación de la relación atributo/métrica.

Atributos\Métricas	TOC	RC	Promedio
Responsabilidad	1	(-)	1
Complejidad de Implementación	1	(-)	1
Reutilización	1	1	1
Acoplamiento	(-)	1	1
Complejidad de Mantenimiento	(-)	1	1

Cantidad de pruebas	(-)	1	1

Tabla 19 Rango de valores para la evaluación de la relación atributo/métrica

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

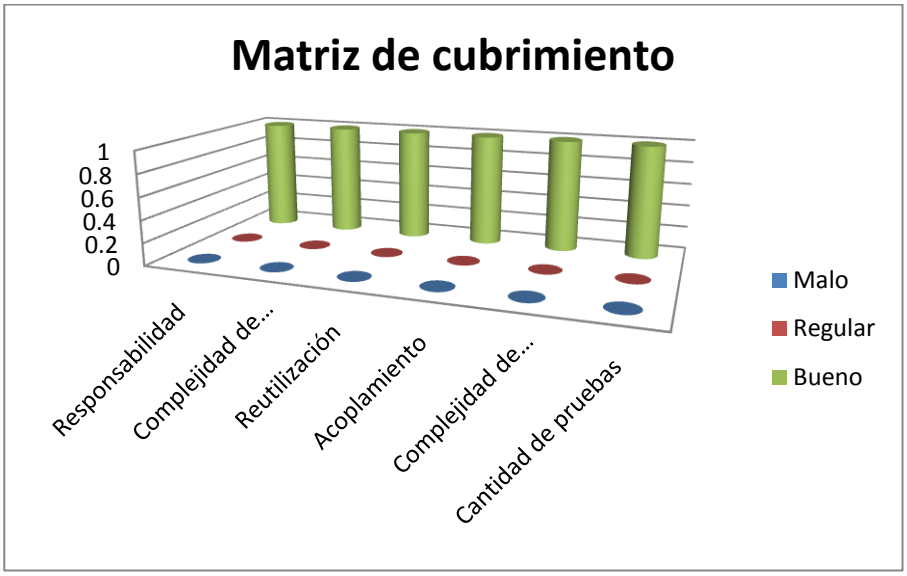

Ilustración 20 Resultados obtenidos de la evaluación de los atributos de calidad.

3.3. PRUEBAS DE SOFTWARE

Las pruebas de software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las del diseño y de la codificación. Dichas pruebas son realizadas con el objetivo de detectar errores en el sistema, por lo que se llevan a cabo durante todo el ciclo de vida del producto.

3.3.1. Casos de prueba

Los casos de prueba especifican una forma de probar el sistema, incluyendo las entradas con las que se ha de probar, las condiciones bajo las que ha de probarse, así como los resultados esperados.

3.3.2. Diseño de casos de prueba

Tabla 20 Escenarios de prueba

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Listar validación de dato.	Se realizará el listar de todas las validaciones de dato.	EP 1.1: Listar validación.	– En este escenario se deberá listar todas las validaciones de dato. – En el árbol de los subsistemas deberá seleccionar un subsistema para listar todas las validaciones de datos pertenecientes a este subsistema
2: Eliminar validación de dato	Se realizará la eliminación de la validación de dato.	EP 1.1: Eliminar validación	– En este escenario se deberá eliminar la validación de dato. – En el grid donde se muestran las validaciones de cada subsistema deberá seleccionar una validación – El sistema muestra un mensaje de confirmación, 'Está seguro que desea eliminar la validación.'. – Debe presionar el botón Aceptar. – El sistema muestra un mensaje, 'La validación fue eliminada correctamente.'.
		EP 1.2: Cancelar	– En este escenario se deberá eliminar la validación de dato. – En el grid donde se muestran las

			validaciones de cada subsistema deberá seleccionar una validación – El sistema muestra un mensaje de confirmación, 'Está seguro que desea eliminar la validación.'. – Debe presionar el botón Cancelar. – Se cierra la ventana de información sin realizar ninguna operación.

Los demás diseños de casos de pruebas ver en Anexos.

A continuación se muestra la cantidad de no conformidades por iteraciones durante las pruebas funcionales realizadas por el departamento de calidad del CEIGE, ver Ilustración 22.

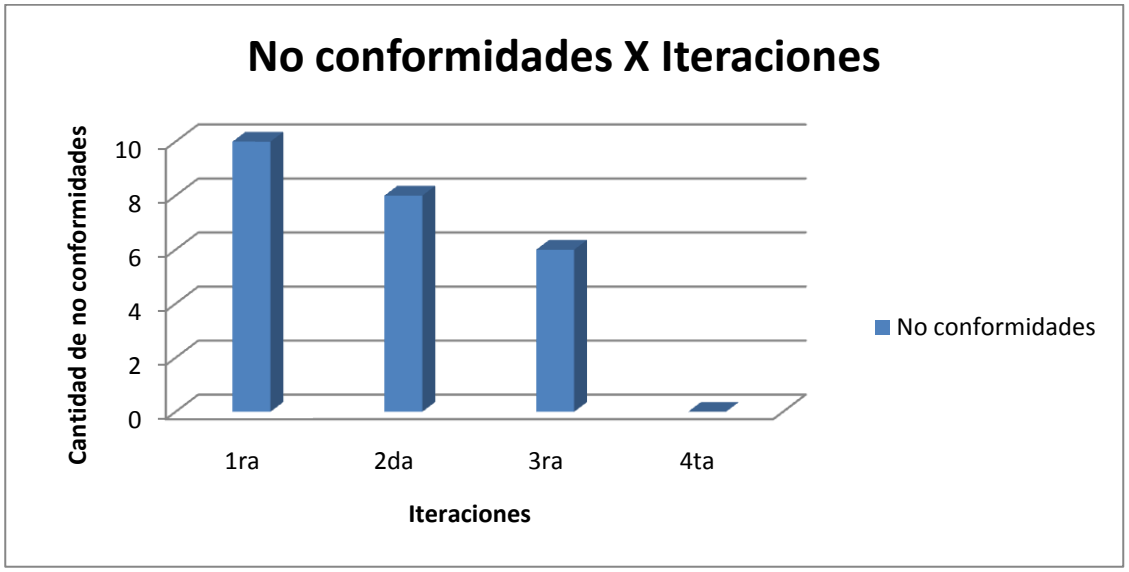

Ilustración 21 Cantidad de no conformidades por iteraciones.

Como se puede apreciar en la Ilustración 23, el 5% de los escenarios de pruebas presentaban no conformidades, las cuales fueron corregidas para dejar en un 100% de funcionamiento la herramienta.

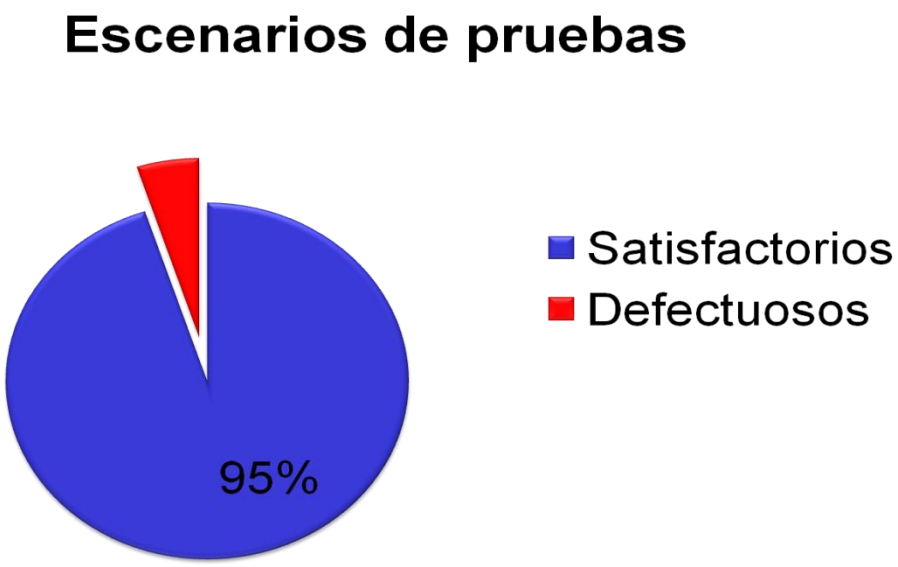

Ilustración 22 Porciento de escenarios de pruebas satisfactorios y defectuosos.

La aplicación fue probada por el Departamento de Calidad del CEIGE, donde se comprobó el correcto funcionamiento de la misma, lo cual queda avalado por el acta de liberación de la herramienta expuesta en los anexos del 6-9.

3.3.3. Pruebas de caja blanca

Las pruebas de caja blanca se basan en el minucioso examen de los detalles procedimentales, se comprueban los caminos lógicos del software proponiendo casos de pruebas que ejerciten conjuntos específicos de condiciones y/o bucles. Todo lo que tenemos que hacer es definir todos los caminos lógicos, desarrollar casos de pruebas que los ejerciten y evaluar los resultados.

Para esto primero se procede a enumerar las sentencias del código y a partir del mismo se construye el grafo de flujo asociado.

```
public function cargarsistemaAction()
    {
        $certificado = $this->global->Perfil->certificado;1
        $entidad = $this->global->Estructura->idestructura;1
        $idnodo = $this->_request->getPost('node');1

        $sistemas = $this->integrator->seguridad->getSistems($certificado, $entidad, $idnodo);1

        if(count($sistemas))2
        {
            foreach($sistemas as $valores=>$valor)3
            {
                $sistemaArr[$valores]['id'] = $valor['id'];4
                $sistemaArr[$valores]['text'] = $valor['text'];4
                $sistemaArr[$valores]['abreviatura'] = $valor['abreviatura'];4
                $sistemaArr[$valores]['descripcion'] = $valor['descripcion'];4
                $sistemaArr[$valores]['idpadre'] = $valor['idpadre'];4
                $sistemaArr[$valores]['icono'] = $valor['icono'];4
                $sistemaArr[$valores]['leaf'] = true;4
            }5
            echo json_encode ($sistemaArr);return;6
        }
        else 7
        {
            $sist = $sistemas->toArray();8
            echo json_encode($sist);return;8
        }
    }9
```

Ilustración 23 Código fuente de la funcionalidad Cargar sistemas.

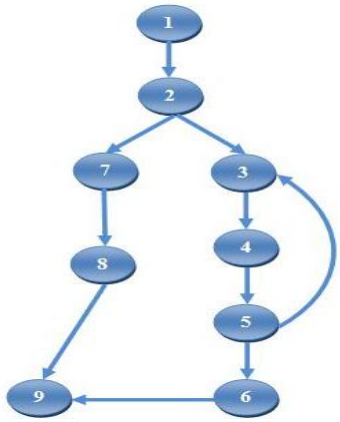

Ilustración 24 Grafo de flujo asociado a la funcionalidad Cargar Sistemas.

Luego de haber construido el grafo se realiza el cálculo de la complejidad ciclomática mediante las tres fórmulas descritas a continuación, las cuales deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad es correcto.

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

V (G) = (A – N) + 2
V (G) = (10 - 9) + 2
V (G) = 3

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

V (G) = P + 1
V (G) = 2 + 1
V (G) = 3

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

V (G) = R
V (G) = 3

El cálculo realizado mediante las tres fórmulas ha dado el mismo valor, dando como resultado 3, lo que indica que existen 3 posibles caminos por donde el flujo puede circular, y determina el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo:

- ✓ **Camino básico #1:** 1-2-3-4-5-6-9
- ✓ **Camino básico #2:** 1-2-3-4-5-3-4-5-6-9
- ✓ **Camino básico #3:** 1-2-7-8-9

Para cada camino se realiza un caso de prueba.

Caso de prueba para el camino básico #1:

Descripción: Los datos se obtienen a través del método Post y las variables globales Perfil y Estructura, donde cumplirán con los siguientes requisitos: el certificado, la entidad y el nodo será cadenas de números.

Condición de ejecución: Los valores obtenidos no deben ser nulos.

Obtiene: $certificado= 1900000, $entidad = 1900000, $nodo = 2

Resultados esperados: Un sistema.

Caso de prueba para el camino básico #2:

Descripción: Los datos se obtienen a través del método Post y las variables globales Perfil y Estructura, donde cumplirán con los siguientes requisitos: el certificado, la entidad y el nodo será cadenas de números.

Condición de ejecución: Los valores obtenidos no deben ser nulos.

Obtiene: $certificado= 1900000, $entidad = 1900000, $nodo = 2

Resultados esperados: Un arreglo de todos los sistemas existentes.

Caso de prueba para el camino básico #3:

Descripción: Los datos se obtienen a través del método Post y las variables globales Perfil y Estructura, donde cumplirán con los siguientes requisitos: el certificado, la entidad y el nodo será cadenas de números.

Condición de ejecución: Los valores obtenidos no deben ser nulos.

Obtiene: $certificado= 1900000, $entidad = 1900000, $nodo = 2

Resultados esperados: Un arreglo vacio.

Luego de aplicar los distintos casos de pruebas, se pudo comprobar que el flujo de trabajo de la función es correcto ya que cumple con las condiciones necesarias que se habían planteado.

3.4. CONCLUSIONES PARCIALES

En este capítulo fueron expuestos los artefactos generados como parte del modelo de implementación de la solución propuesta, tales como el diagrama de componente, además del diagrama de despliegue y los estándares de codificación. Se pudo apreciar como el modelo de implementación constituye una entrada fundamental para las pruebas de software. Se describió la prueba de caja blanca realizada que permitió comprobar el correcto funcionamiento de la herramienta. Se validó el diseño mediante la aplicación de métricas para la evaluación de atributos de calidad, las cuales arrojaron resultados satisfactorios, además de las pruebas realizadas a la aplicación por el departamento de Calidad del CEIGE, la cual fue avalada por un acta de liberación.

Conclusiones

Se desarrollo una investigación de los distintos conceptos y aspectos necesarios para entender el objetivo fundamental del trabajo, así como un estudio de los marcos de trabajos existentes que de una forma u otra gestionan las validaciones. Durante la etapa de análisis y diseño se identificaron procesos de negocios a los cuales asistió la aplicación desarrollada, además se realizaron varios diagramas con el objetivo de organizar el proceso de desarrollo de la aplicación, Se validó el diseño de la aplicación a través de las métricas de diseño aplicadas obteniendo resultados satisfactorios, además se realizaron pruebas de calidad a la herramienta arrojando varias no conformidades las cuales fueron corregidas en el transcurso de las distintas iteraciones realizadas a la herramienta. Se realizaron pruebas de caja blanca con las cuales se pudo comprobar que el flujo de trabajo de la función es correcto. Por todo lo antes planteado se desarrollo de una herramienta que permita la configuración visual de las validaciones en el marco de trabajo Sauxe, con la que se disminuyó el tamaño del archivo XML, se redujo el tiempo de búsqueda ya que implementa un buscador disminuyendo la complejidad del proceso, además de que permite al desarrollador realizar de forma visual la configuración de las validaciones por lo que el proceso se hace más fácil y rápido, por lo antes planteado se puede concluir que con el desarrollo de esta herramienta se optimizó el proceso de configuración de las validaciones en el marco de trabajo Sauxe.

Recomendaciones

Ningún sistema es estático en el tiempo debido a que la evolución del negocio y el desarrollo de las tecnologías 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.
- Que se utilice el componente como punto de partida para la implementación de futuros componentes relacionados con la configuración de validaciones.

Referencia bibliográfica

1. **Joomla.** Syswoody. *Syswoody.* [En línea] [Citado el: 12 de Febrero de 2011.] http://www.syswoody.com/utilidades/analisis-de-seguridad-en-aplicaciones-web.

2. **Vega, Jesus.** Introducción a las aplicaciones web. *Introducción a las aplicaciones web.* [En línea] 21 de Marzo de 2002. [Citado el: 12 de Febrero de 2011.] http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node11.html.

3. **Technology Solutions.** Ventajas de las aplicaciones web. [En línea] [Citado el: 20 de Mayo de 2011.] http://www.solcre.com/files/ventajas_de_las_aplicaciones_web.pdf.

4. Visitask. *Visitask.* [En línea] [Citado el: 12 de Febrero de 2011.] http://www.visitask.com/validation-g.asp.

5. **Marcos Guglielmetti, Analia Lanzillotta, Guillem Alsina, David Yanover.** MasterMagazine. *MasterMagazine.* [En línea] 2004. [Citado el: 5 de Marzo de 2011.] http://www.mastermagazine.info/termino/7068.php.

6. **Microsoft.** msdn. *msdn.* [En línea] [Citado el: 22 de Marzo de 2011.] http://msdn.microsoft.com/es-es/library/aa291820%28VS.71%29.aspx.

7. —. msdn. *msdn.* [En línea] [Citado el: 22 de Marzo de 2011.] http://msdn.microsoft.com/en-us/library/aa659223%28AX.10%29.aspx.

8. **Codebox.** Web coders in a box. *Web coders in a box.* [En línea] [Citado el: 22 de Marzo de 2011.] http://www.codebox.es/glosario .

9. **Eguiluz, Javier.** Symfony.es. *Symfony.es.* [En línea] [Citado el: 24 de Marzo de 2011.] http://www.codebox.es/glosario .

10. **Zend Technologies Ltd.** Zend framework. *Zend framework.* [En línea] Zend Technologies Ltd, 2010. [Citado el: 24 de Marzo de 2011.] http://www.framework.zend.com/about/overview.

11. **Sommerville, Ian.** *Ingeniería del software(Séptima edición).* Madrid : PEARSON ADDISON WESLEY, 2005. 84-7829-074-5.

12. **usr.code.** *Ciclo de vida del software.*

13. **Producción, Equipo de.** Modelo de Desarrollo orientado a componentes del proyecto ERP-CUBA.

14. Lenguajes de programación. [En línea] 2009. [Citado el: 9 de Junio de 2011.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml.

15. **Quer System Informática, S.L.** Quersystem. *Quersystem.* [En línea] 2009. [Citado el: 20 de Abril de 2011.] http://www.quersystem.com/index.php?option=com_content&view=article&id=20&Itemid=3.

16. Servidor de web Apache. *Servidor de web Apache.* [En línea] [Citado el: 28 de Marzo de 2011.] http://acsblog.es/articulos/trunk/LinuxActual/Apache/html/x31.html.

17. **Küng, S. Onken, L. Large.** rediris.es. [En línea] 2007. [Citado el: 10 de Abril de 2011.] https://forja.rediris.es/docman/view.php/123/117/TortoiseSVN-1.4.1-es.pdf.

18. **Atlassian Confluence 3.4.6.** ideasoft.biz. [En línea] [Citado el: 15 de Abril de 2011.] https://www.ideasoft.biz/wiki/display/O3PS/Glosario+O3+Process .

19. Modelado de procesos negocio. *Presentacion.* 2010.

20. **Sergio Martín, Eugenio López, José Antonio Cámara, Germán Carro, Guillermo F. Lafuente, Maria M. García, Gloria Murillo,Serafín Doblado.** DIEEC - Departamento de Ingenieria Eléctrica, Electrónica y de Control (ieee). *DIEEC - Departamento de Ingenieria Eléctrica, Electrónica y de Control (ieee).* [En línea] 2007. [Citado el: 15 de Abril de 2011.] http://www.ieec.uned.es/ieee/investigacion/ieee_dieec/sb/boletin/boletin_8_Octubre_2007.pdf.

21. **Tedeschi, Nicolás.** MSDN. [En línea] [Citado el: 24 de 3 de 2011.] http://msdn.microsoft.com/es-es/library/bb972240.aspx.

22. **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.

23. **Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides.** *Design Patterns Elements of Reusable Object-Oriented Software.*

24. **Allsoft.** slideshare. [En línea] 2008. [Citado el: 8 de Junio de 2011.] http://www.slideshare.net/inventa2/modelos-de-desarrollo.

25. Definición.de. *Definición de metodología.* [En línea] [Citado el: 8 de Junio de 2011.] http://definicion.de/metodologia/.