**Universidad de las Ciencias Informáticas**

Facultad 3


# Componente para la gestión de notificaciones en el marco de trabajo Sauxe.

Trabajo de Diploma para optar por el título de Ingeniero Informático.

**Autor:**

Ariel Trujillo Díaz.

**Tutores:**

Lic. Orlando Arnaldo Valenzuela Aguilera.

Ing. Yuniel Cedeño Mendoza.

# DECLARACIÓN DE AUTORÍA

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

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

Autor: Ariel Trujillo Díaz

_____

Tutor: Lic. Orlando Arnaldo Valenzuela Aguilera.

_____

Tutor: Ing. Yuniel Cedeño Mendoza.

_____

# DATOS DE CONTACTO

**Autor:**

Ariel Trujillo Díaz.

Universidad de las Ciencias Informáticas, La Habana, Cuba

Email: atdiaz@estudiantes.uci.cu

**Tutores**:

Lic. Orlando Arnaldo Valenzuela Aguilera.

Universidad de las Ciencias Informáticas, La Habana, Cuba

Email: ovalenzuela@uci.cu

Ing. Yuniel Cedeño Mendoza

Universidad de las Ciencias Informáticas, La Habana, Cuba

Email: ycedenom@uci.cu

# AGRADECIMIENTOS

*Primeramente a mis padres que supieron con su esfuerzo, darme la educación y guiarme por el camino correcto hasta llegar a lo que soy hoy, mi hermano Arián que representa siempre un ejemplo a seguir para mí, mi otro hermano Ariexi que me dio mucha fuerza voluntad y ayuda también para lograr todo lo que me he propuesto hasta ahora. De toda esa maravillosa familia me he apoyado para estar aquí hoy discutiendo mi tesis para poder graduarme de ingeniero y de los cuales me siento orgulloso. Sin dejar de mencionar a todo el colectivo de profesores de mi proyecto que me ayudaron a echar a andar el dichoso componente de notificaciones del marco de trabajo Sauxe junto a mis tutores, mis compañeros de proyecto, los de mi aula con los que he compartido todos estos años. No puede faltar mi novia la que me ha aguantado mis pesadeces estos últimos años y es la que más cerca de mí ha estado en los momentos tensos de mi tesis y siempre lograba sacarme de ellos. Mis amigos y conocidos dentro y fuera de la uci que han complementado todo este esfuerzo.*

# DEDICATORIA

*Dedico este resultado a toda mi familia que sin ellos no hubiese sido posible, a mis amigos que todos me apoyaron en los momentos difíciles y con los cuales conté siempre en todos estos años y todos contribuyeron a ser lo que soy hoy, a todos ellos va dedicado este resultado.*

## RESUMEN

El presente trabajo tiene como objetivo desarrollar una solución que permita a los usuarios del marco de trabajo Sauxe poder conocer la ocurrencia de un evento determinado en el instante que suceda en el servidor de aplicaciones, sin que tengan estos que intervenir para que se actualice la información. Con esta solución se pretende mejorar el tiempo de respuesta que pueda tener un usuario determinado para realizar una tarea que tenga que ver con que se desencadene dicho evento. Para ayudar al desarrollo de la solución se realizó un estudio de algunas técnicas de notificación en tiempo real empleadas en aplicaciones web, con el objetivo de determinar los nuevos escenarios que se deberían cubrir. Además se efectuó un estudio de las herramientas y tecnologías a utilizar a lo largo del desarrollo de la solución y se identificaron y validaron los requisitos funcionales con que contaría la misma, mediante técnicas de captura y validación de requisitos. También se describen los patrones arquitectónicos y de diseños a utilizar para implementar la propuesta de solución para proveer reutilización al componente, ahorrar tiempo en la implementación y obtener un producto con mejor calidad y mediante pruebas de software se valida que el componente pueda cumplir con el objetivo que persigue esta investigación.

**Palabras claves:**

Tiempo de respuesta, información, tiempo real.

## Índice

## INTRODUCCIÓN

El hecho de vivir en la era de la información establece al ser humano como meta esencial, la necesidad de aumentar el volumen de información que se necesita para subsistir e incluso poder ser competitivo, en este ámbito, la rapidez con que se es capaz de obtener la información juega un papel muy importante y realmente llega a marcar la diferencia en un ámbito en el cual, el más informado tiene ventaja sobre los demás. Las Tecnologías de la Información y las Comunicaciones (TIC), son un ejemplo muy elocuente e ilustrativo de esto. Las cuales han tomado un rol protagónico y determinante en el desarrollo social y económico de las sociedades. Es a través de las TIC fundamentalmente, que hoy en día, se comercia, se atan lazos de cooperación entre naciones y sociedades, se establecen contratos y se fomenta el intercambio de información.

Esta situación ha traído consigo nuevos conceptos, normas y estándares. Muchos paradigmas se han vuelto obsoletos e inútiles, otros nuevos han ocupado su lugar y han visto la luz por primera vez técnicas y herramientas que han venido a solucionar situaciones anteriormente complejas y facilitar el trabajo en la sociedad, como por ejemplo, las que agilizan el proceso de desarrollo de aplicaciones web de gestión; entre las que se encuentran los marcos de trabajo, los cuales, son plataformas que gestionan artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado. Incluyen además, un lenguaje interpretado para ayudar a desarrollar y unir los diferentes componentes de un proyecto y permite la portabilidad entre arquitecturas. (1)

Cuba no se puede quedar retrasada en cuanto a estos avances, eso sería negar la natural e inevitable evolución del pensamiento humano. Por eso es creada la Universidad de las Ciencias Informáticas (UCI), en la cual dentro de otros proyectos e ideas surge el Centro Informatización de la Gestión de Entidades (CEIGE).

En este centro se desarrolla un marco de trabajo llamado Sauxe, el cual contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo incluyendo:

El primer elemento que debe regir una selección adecuada de alcance de un marco tecnológico el cual radica en las restricciones de diseño que el mismo asume, basados en las tecnologías, la capacidad

técnica del equipo de desarrollo, los intereses jurídicos mercantiles así como la infraestructura tanto de la organización productora como cliente de los productos que el mismo facilitara desarrollar. Sauxe cuenta con una arquitectura en capas como se muestra en la ***Figura 1*** que a su vez presenta en su capa superior un MVC[1]. Contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. Siguiendo el paradigma de independencia tecnológica por el cual apuesta el país (2).

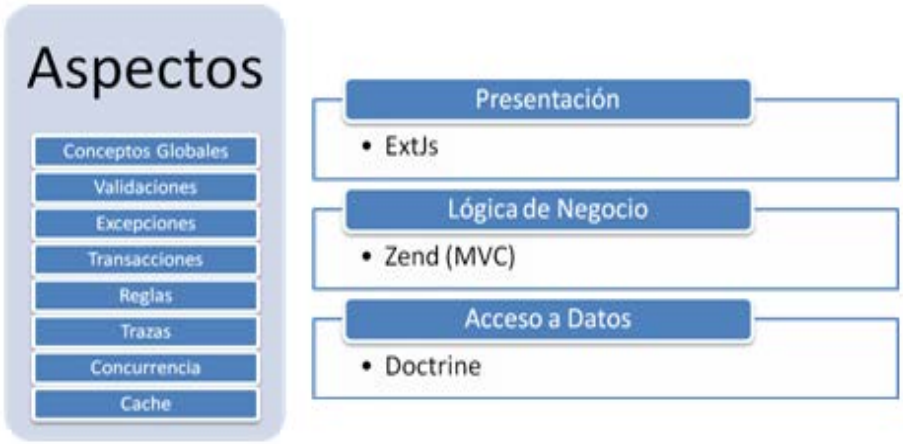



*Figura 1 Arquitectura de Sauxe.*

Actualmente por la propia naturaleza de la arquitectura cliente-servidor empleada por las aplicaciones web de Sauxe, un usuario perteneciente al mismo puede conocer una determinada información generada en el servidor de aplicaciones, cuando realice una petición utilizando el protocolo HTTP[2] desde su puesto de trabajo, posteriormente y a través de mismo protocolo obtiene la respuesta del servidor con los datos que pidió. También puede recibir dicha información fuera del ámbito del marco de trabajo, cuando a través de otro usuario y utilizando otras vías de comunicación se dé por enterado. Ambos escenarios afectan el tiempo de respuesta de un usuario a una actividad y en consecuencia aumenta el tiempo de duración de la misma, afectándose de igual forma el proceso de toma de toma de decisiones.

[1] **MVC:** Patrón arquitectónico Modelo-Vista-Controlador.
[2] **HTTP:** por su significado en español (Protocolo de transferencia de hipertexto).

En aras de erradicar estos obstáculos se define como **problema a resolver**: ¿Cómo garantizar que las notificaciones de los sistemas desarrollados en Sauxe lleguen a los usuarios correspondientes en tiempo real? Por lo cual se toma como **objeto de estudio**, las técnicas y herramientas para la gestión de notificaciones en tiempo real. Según lo planteado anteriormente se establece como **Campo de Acción**, la gestión de notificaciones en aplicaciones web de gestión. Quedando definido de esta manera, que el **objetivo general** que tiene esta investigación es:

Desarrollar un componente para garantizar que las aplicaciones del marco de trabajo Sauxe puedan enviar notificaciones en tiempo real.

A partir del objetivo general se derivan 4 **Objetivos Específicos** quedando de la siguiente manera:

1. Realizar el Marco Teórico de la investigación.
2. Realizar el análisis y el diseño de la solución propuesta.
3. Implementar un componente que permita la gestión de las notificaciones en el marco de trabajo Sauxe.
4. Validar la solución propuesta.

Para dar cumplimiento a los objetivos específicos se plantean las siguientes **tareas de investigación** a realizar:

- Recopilar información sobre el estado del arte de sistemas similares existentes en la actualidad, para obtener datos que puedan ayudar en la elaboración del sistema.
- Identificar de los Requisitos funcionales que debe cumplir el componente.
- Realizar el modelo conceptual para poder comprender el alcance del problema.
- Recopilar información sobre los patrones de diseño y estilos arquitectónicos para garantizar un diseño robusto y flexible.
- Definir los prototipos de interfaz de usuario para la validación de los requisitos.
- Implementar los componentes diseñados.
- Elaborar diagrama de componentes para dar cumplimiento a los objetivos propuestos.
- Realizar el diagrama de clases correspondiente al componente.
- Realizar los diseños de casos de prueba.

- Realizar pruebas de caja blanca.
- Realizar pruebas de caja negra.

La **idea a defender** con la realización del componente de notificaciones para el marco de trabajo Sauxe, se podrá dar conocer a los usuarios del mismo una determinada información, en el instante en que se genere o cambie del lado del servidor, sin la necesidad de que el usuario actualice manualmente la fuente de información, impregnando al proceso instantaneidad, lo cual contribuirá a economizar el tiempo y así beneficiar el proceso de toma de decisiones.

Para el desarrollo de las tareas científicas se utilizan Métodos de Investigación en la búsqueda y procesamiento de la información. Los mismos se dividen en teóricos y empíricos. Los Métodos Teóricos son factibles en el estudio de las características poco observables del objeto de investigación. Dentro de este grupo se utilizan:

- El método **Análisis Histórico-Lógico**, posibilitó estudiar de forma analítica la trayectoria histórica real de los fenómenos, su evolución y desarrollo. El método permitió realizar la primera parte de la investigación, al hacer un análisis bibliográfico de otros módulos o componentes de notificación de alertas o avisos, el Razonamiento Basado en caso y técnicas de programación más utilizadas en el desarrollo de este tipo de componentes. Dio paso a la exploración de trabajos realizados en el ámbito de notificar eventos en un sistema informático y de soluciones previas existentes a problemas similares al actual. Se utilizó para determinar a través de la evaluación de la bibliografía conceptos de esta temática, que permiten conocer el estado de la evolución actual del fenómeno e identificar posibles mejoras y alternativas de solución.
- El **Analítico-Sintético**, se utilizó para el estudio a partir de fuentes bibliográficas seguras conceptos y técnicas con soluciones previamente desarrolladas. Permitió además descomponer el problema de investigación en elementos por separado y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución propuesta.

Los **métodos Empíricos** tienen gran importancia ya que permiten efectuar el análisis preliminar de la información, así como verificar y comprobar las concepciones teóricas. Dentro de este grupo se utiliza:

- El **método de Observación**: Planificada y dirigida con el fin de realizar la fundamentación teórica del problema.
- El **método de Entrevista**: Apoyará a la incorporación de conocimientos mediante las entrevistas planificadas efectuadas a los especialistas de áreas de CEIGE.

El documento cuenta con la siguiente estructura:

**Capítulo 1: Fundamentación Teórica**. Se realiza un estudio del estado del arte sobre las técnicas de notificación en tiempo real empleadas en aplicaciones web. Se describen tanto las tecnologías, metodologías y herramientas definidas por CEIGE para el desarrollo de sistemas en el centro, como algunas otras propuestas por el autor.

**Capítulo 2: Propuesta de solución**. Se describen los requisitos funcionales y no funcionales con que contará el componente para la gestión de notificaciones en el marco de trabajo Sauxe. Se elabora el diagrama de clases del diseño correspondiente a la solución y por último se describen los patrones de diseño y arquitectónicos definidos para el desarrollo de la solución.

**Capítulo 3: Implementación y Pruebas**. Se especifican aspectos de interés para el proceso de implementación tales como los estándares codificación para el código fuente del sistema, el diagrama de componentes que conforma la solución y el diagrama de despliegue donde se identifican los nodos necesarios para el despliegue de la solución. Se especifican además los resultados de la validación del componente mediante métricas de diseño y pruebas de software realizadas y finalmente se valora que beneficios trae el resultado obtenido en dichas pruebas.

# Capítulo I: FUNDAMENTACIÓN TEÓRICA

## Introducción

En este capítulo se explica en qué consiste una técnica de notificación en tiempo real y se especifica el concepto de tiempo real, el cual será necesario conocer ya que se tratará a lo largo de todo el desarrollo del trabajo. Además se realiza un estudio de las distintas técnicas de notificación en tiempo real que se utilizan fuera del ámbito de CEIGE para tratar de resolver problemas similares al que da razón a este trabajo. Se detallan algunas características fundamentales del modelo de desarrollo a utilizar y se exponen los respectivos artefactos generados por dicho modelo durante las primeras fases de desarrollo del componente. Finalmente se caracterizan las técnicas y herramientas que define CEIGE a utilizar para realizar soluciones para el Departamento de Tecnologías.

A continuación se introduce el concepto de tiempo real para tener una referencia y un punto de partida común que ayude a entender mejor esta situación, se exponen distintos conceptos de diferentes bibliografías

**Sistemas de Tiempo Real (STR)**

Un Sistema de Tiempo Real, se define como: “Un sistema en el que el tiempo en que se produce su salida es significante. Esto es debido a que generalmente la entrada corresponde a algún instante del mundo físico y la salida tiene relación con ese mismo instante" (3).

**Tiempo real**

Funcionamiento de un sistema en el que los procesos se realizan simultáneamente a los hechos que representan (4).

Se dice que un proceso ocurre en tiempo real cuando realiza una transacción que ha sido enviada desde un terminal en ese mismo momento, sin espera alguna (5).

## 1.1 Técnicas de notificación en tiempo real

Las técnicas de notificación utilizadas en aplicaciones web permiten a los usuarios recibir notificaciones tan pronto como la información es publicada, sin necesidad de chequear la fuente original manualmente para obtener actualizaciones. (6) A partir de esta definición se exponen a continuación las técnicas de notificación estudiadas.

### 1.1.1 Web Sockets

Según W3C (World Wide Web Consortium) Es una nueva tecnología de HTML5 que permite a las aplicaciones web mantener una comunicación bidireccional con procesos en el lado del servidor. La misma sucede a través del capa 3 del modelo de referencia de Interconexión de Sistemas Abiertos (OSI, Open System Inter connection), y se realiza entre cliente – servidor utilizando un mecanismo para ponerlos en contacto, API[3]. Tanto el servidor y el cliente o clientes utilizan una aplicación, la cual es responsable de interactuar con los sockets[4], dándole responsabilidad el sistema operativo de utilizar un proceso para crear, utilizar y luego cerrar el socket cuando ya se ha transmitido el mensaje. El usar procesos del sistema operativo puede representar un gran consumo de recursos de ambas partes, además de representar una relativa demora, la cual se puede incrementar considerablemente si se usa Web Socket Secure connections (wss), porque el mismo utiliza Transport Layer Security (TLS) y representa un costo adicional en cuanto al tiempo que puede tardar para encriptar.

Aunque el protocolo Websockets es indiferente a la conexión sobre servidores proxy o cortafuegos, implementa una negociación compatible con HTTP[5] para que los servidores HTTP puedan compartir sus puertos HTTP y HTTPS[6] por defecto (80 y 443) con una pasarela o servidor WebSocket y los mensajes enviados corren el riesgo de ser bloqueados, provocando que la conexión falle (7).

También es importante destacar que esta tecnología usada en navegadores web por ser precisamente nueva, presenta algunas desventajas ya que existen algunas versiones navegadores que no lo implementan o lo hacen parcialmente. Para complementar esta información en la ***Figura 2*** se muestra una tabla la cual relaciona algunos componentes de HTML5 dentro de ellos websockets que son o no

[3] **API:** Application Programming Interface.
[4] **Socket:** (enchufe), es un método para la comunicación entre un programa del cliente y un programa del servidor en una red.
[5] **HTTP:** por su significado en español (Protocolo de transferencia de hipertexto).
[6] **HTTPS:** por su significado en español (Protocolo seguro de transferencia de hipertexto).

soportados por algunas versiones de los navegadores existentes en la actualidad para los sistemas operativos Windows y MAC. Esta información se encuentra disponible en el sitio *http://www.findmebyip.com/litmus/.*


*Figura 2 Versiones de navegadores que soportan web sockets.*

### 1.1.2 COMET

El término COMET describe a las aplicaciones donde el servidor se mantiene enviando los datos, o manteniendo un cúmulo continuo de datos a la aplicación cliente, en vez de tener al navegador realizando peticiones al servidor para actualizar el contenido, es decir COMET cambia fundamentalmente la naturaleza de la comunicación realizada entre la arquitectura conocida Cliente-Servidor.

COMET se basa en que el servidor envía datos aunque el cliente no los pida (HTTP Push), si se hace una llamada, el canal se queda abierto y el servidor va enviando información, o lo que es lo mismo, que el servidor va a estar devolviendo el resultado en partes. En otras palabras todas las aplicaciones de COMET utilizan conexiones HTTP de larga duración para reducir a latencia [7]con la cual los mensajes son enviados al servidor. En esencia no realizan pedidos ocasionalmente al servidor. En vez de eso el servidor mantiene una línea abierta de comunicación mediante la cual puede "empujar" datos hacia el cliente. COMET utiliza XMLHttpRequest para la entrega de datos entre el cliente y el servidor a través del protocolo HTTP. También es conocido como Server Push o HTTP Push. He aquí algunas desventajas relacionadas con el uso de esta técnica:

- Este método podría significar un desperdicio en términos de sockets e hilos, y también representaría una sobrecarga para los cortafuegos, y los balanceadores de carga ya que los mismos deben estar chequeando frecuentemente los datos enviados por el canal de información que efectúa COMET.
- Esta técnica al mantener abierta la conexión por un largo período, puede presentar problemas de escalabilidad[8] en una aplicación. (8)

### 1.1.3 Jabber

*Jabber es un sistema basado en el lenguaje XML[9] para el intercambio en cuasi tiempo real de mensajes y presencia.*

**Presencia**: *Indica si una entidad Jabber está disponible para la comunicación. Incluye disponibilidad básica (o sea, conectado o desconectado) y también indicadores de estado que determinan tipos de disponibilidad*. (9)

### 1.1.4 Protocolo XMPP/Jabber

Es el protocolo usado por Jabber, también el usado por otras aplicaciones como Google Talk y LiveJournal Talk por lo que son interoperables[10]. El protocolo está estandarizado por el IETF[11], si bien las

[7] **Latencia**: la suma de retardos temporales dentro de una red.
[8] **Escalabilidad:** Es la capacidad de mejorar recursos para ofrecer una mejora (idealmente) lineal en la capacidad de servicio.
[9] **XML**: Extensible Markup Language.
[10] **Interoperable:** Característica de los ordenadores que les permite su interconexión y funcionamiento conjunto de manera compatible.

extensiones (XEP, XMPP Estandard Propásale, antes conocidas como JEP extensiones a este estándar son definidas por la XMPP Software Foundation. Está basado en un conjunto de tecnologías XML para aplicaciones en tiempo real. Fue desarrollado originalmente como un marco de trabajo de aplicaciones de mensajería instantánea y presencia para escenarios de empresas. El mismo utiliza ficheros en formato XML para hacer las transferencias de mensajes entre sus entidades cliente-servidor o servidor-servidor (9).

**Características**

**Protocolo abierto:** Con todas las ventajas del software libre, se puede programar un servidor o un cliente o ver el código.

**Descentralizado:** Se puede crear un servidor para Jabber, y se puede interoperar o unirse al resto de la red Jabber.

**Extensible:** Se puede ampliar con mejoras sobre el protocolo original. Las extensiones comunes son manejadas por la XMPP Standards Foundation**.**

**Seguro:** Cualquier servidor Jabber está aislado del exterior. El servidor de referencia permite SSL para comunicaciones cliente-servidor y algunos clientes aceptan GPG como cifrado de las comunicaciones usando cifrado asimétrico. En desarrollo uso de claves de sesión y SASL (10).

### 1.1.5 Servidores que utilizan el protocolo XMPP/Jabber

#### 1.1.5.1 EJabberd

Es un servidor multiplataforma desarrollado en lenguaje Erlang y de código abierto. La instalación es sencilla, ya que la distribución de Erlang [12]provee de todos los componentes que necesitará eJabberd. Soporta Ipv6. Implementa casi de forma completa el XMPP y varias de las extensiones J.E.P (Jabber Extension Protocol). El número de sitios que lo utilizan como solución no es muy elevado, no tiene una comunidad de usuarios muy amplia y eso se traduce en un soporte menos eficiente. En cuanto a la parte de desarrollo, el estar implementado en Erlang es lo que lo pone en desventaja con respecto a servidores como es el caso de Jive y Jabberd (hechos en java y C/C++ respectivamente) ya que está limitado el

[11] **IETF:** Internet Engineering Task Force.
[12] **Erlang:** Es un lenguaje de programación concurrente y un sistema de ejecución que incluye una máquina virtual y bibliotecas.

desarrollo a una comunidad más reducida de personas que conozcan el lenguaje. Por otro lado, la configuración de eJabberd no resulta ser tan sencilla. Pero posee una interfaz gráfica que permite configurarlo, la cual es accesible vía web a la dirección **http://[Url donde se ubica el servidor]/admin**, para acceso remoto. Esta interfaz permitirá administrar las ACL[13]. (11)

#### 1.1.5.2 Openfire

Antes llamado Servidor Wildfire es un servidor desarrollado en Java provee licencias comerciales y GNU. La administración del servidor se hace a través de una interfaz web, que corre por defecto en el puerto 9090 (HTTP) y 9091 (HTTPS). Los administradores pueden conectarse desde cualquier lugar y editar la configuración del servidor, agregar y borrar usuarios, crear cuartos de conferencia permanentes. Openfire implementa las siguientes características. Presenta un panel de administración web a la cual se puede acceder por la dirección http://**[Url donde se ubica el servidor]**:9090, y una interfaz ajustable para agregar plugins[14] para particularizar su funcionamiento, agregar o modificar funcionalidades. Utiliza SSL/TLS para la encriptación durante la transmisión de datos a través de las comunicaciones cliente-servidor o servidor-servidor. Puede interactuar con otros servicios de mensajería instantánea mundialmente conocidos como MSN, Google Talk, Yahoo messenger, AIM, ICQ y es capaz de ofrecer estadísticas del servidor, mensajes y paquetes. Tiene soporte para la autenticación vía Certificados, Kerbeos, LDAP, MS SQL, MySQL, Oracle y Postgres (10).

Al utilizar un servidor que implementa el protocolo XMPP\Jabber se obtiene de forma nativa la posibilidad de gestionar usuarios y grupos de usuarios lo cual es necesario para la solución propuesta y representa una ventaja que debe ser aprovechada, ya que además de representar una manera de organizar los emisores y destinatarios de las notificaciones, es importante destacar que Sauxe también organiza sus usuarios de la misma forma, lo cual representa un punto en común a tener en cuenta.

La investigación que se realiza demuestra que las técnicas de notificación en tiempo real Web sockets y COMET aunque cubren escenarios comunes, no satisfacen las necesidades que existen para asegurar la comunicación en tiempo real del componente de gestión de notificaciones, ya que estas presentan algunas desventajas que pueden atentar contra el buen funcionamiento del componente en todos los

[13] **ACL:** término inglés Acces control list.
[14] **Plugin:** Programa que puede anexarse a otro para aumentar sus funcionalidades.

escenarios como es el caso de web sockets su funcionamiento en las versiones antiguas de los navegadores que no tienen soporte para esta técnica. En el caso de la técnica COMET la cual puede sobrecargar el canal de comunicación. Por estas razones antes expuestas se determina que el protocolo XMPP sea la técnica que garantice el mecanismo requerido para lograr enviar las notificaciones en tiempo real en el componente de notificaciones. El hecho que este protocolo utilice el lenguaje XML para el proceso de comunicación entre sus extremos representa una ventaja ya que el mismo; *es la tecnología que permite compartir la información de una manera segura, fiable, fácil. Además, XML permite al programador y los soportes dedicar sus esfuerzos a las tareas importantes cuando trabaja con los datos, ya que algunas tareas tediosas como la validación de estos o el recorrido de las estructuras corre a cargo del lenguaje y está especificado por el estándar, de modo que el programador no tiene que preocuparse por ello* (12). *El metalenguaje aparece como un estándar que estructura el intercambio de información entre las diferentes plataformas* (13).

Por estas razones, no importa que lenguaje o plataforma se encuentre en los extremos, los datos enviados o recibidos podrán ser receptados, comprendidos y manejados, lo cual asegura que el flujo de datos sea seguro y sobre todo rápido sin muchas complicaciones. Como servidor que implemente el protocolo XMPP/Jabber se utilizará el servidor Openfire, ya que el mismo está escrito en java, el cual es un lenguaje potente y con una gran comunidad que brinda soporte al mismo. Además brinda una interfaz para extensiones o plugins y comparte con Sauxe la característica de gestionar usuarios, lo cual ayuda a un mejor control del componente de notificaciones.

### 1.2 Modelo de desarrollo

Para el desarrollo de cualquier producto de software se realizan una serie de tareas entre la idea inicial y el producto final, un modelo de desarrollo establece el orden en el que se harán las cosas en el proyecto, provee de requisitos de entrada y de salida para cada una de las actividades (14). Sommerville (15) define modelo de proceso de software como "*Una representación simplificada de un proceso de software, representada desde una perspectiva específica. Por su naturaleza los modelos son simplificados, por lo tanto un modelo de procesos del software es una abstracción de un proceso real.*"

### 1.3.1 Modelo de desarrollo propuesto

El modelo de desarrollo propuesto a utilizar para desarrollar el componente de notificaciones está definido por CEIGE para los proyectos o sistemas que se desarrollen en el mismo, y fue creado teniendo en cuenta las principales características con las que cuenta el centro. Este modelo está orientado a la reutilización de componentes parametrizables, donde se simplifican las actividades y se eleva el nivel de especialización de los involucrados. Proporciona una guía para regir el proceso de desarrollo de software tecnológico, centrado en la arquitectura. Dicho modelo está basado en principios, prácticas propuestas y algunos elementos de las metodologías SCRUM y RUP. El mismo fue elaborado teniendo en cuenta las características especiales que presenta la UCI.

En general propone una solución sencilla y novedosa, que se centra en el desarrollo de componentes como base tecnológica, con una mayor calidad y en menor tiempo, para su posterior uso en la construcción de productos concretos; además propone dividir el trabajo y el equipo de desarrollo para lograr mayor especialización. Su buena aplicación proporcionará en gran medida la independencia tecnológica de los sistemas finales (16).

**Características**

- **Centrado en la arquitectura:** La arquitectura determina la línea base, los elementos de software estructurales a partir de los elementos de la arquitectura de negocio. Interviene en la gestión de cambios y diseña la evolución e integración del producto. La arquitectura orienta las prioridades del desarrollo y resuelve las necesidades tecnológicas y de soporte para el desarrollo.

- **Orientado a componentes:** Las iteraciones son orientadas por el nivel de significancia arquitectónicas de los componentes, los mismos son abstracciones arquitectónicas de los procesos de negocio y requisitos asociados que modelan, el componente es la unidad de medición y ordenamiento de las iteraciones.

- **Iterativo e incremental:** Las iteraciones son planificadas y coordinadas con el equipo de arquitectura, los clientes y la alta gerencia. Cada iteración constituye el desarrollo de componentes, los cuales son integrados al término de la iteración, permitiendo de esta manera la evolución incremental del producto.

- **Ágil y adaptable al cambio:** El desarrollo de las partes formaliza solamente las características principales de la solución, priorizando los talleres y las comunicaciones entre las personas. Los clientes y funcionales están involucrados en el proyecto y poseen parte de la responsabilidad del éxito del mismo. Los cambios son conciliados semanalmente, discutidos y aprobados.

## 1.3 Tecnologías propuestas

Se estudiaron un conjunto de tecnologías a utilizar durante todo el proceso de desarrollo del componente, debido a que el desarrollo del mismo forma parte de un proceso iniciado por CEIGE, las tecnologías propuestas son definidas por dicho centro, las cuales están referenciadas y brevemente explicadas a continuación:

### 1.4.1 Lenguajes de programación.

Un lenguaje de programación es aquel elemento dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que se ponen a disposición del programador para que éste pueda comunicarse con los dispositivos hardware y software existentes (17). A continuación se describen los lenguajes de programación a utilizados en el desarrollo del componente

#### 1.4.1.1 PHP v5.3.3

PHP (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje de código abierto interpretado, de alto nivel, embebido en páginas HTML y el lenguaje está orientado al desarrollo de aplicaciones web, que son interpretadas del lado del servidor. Sus sintaxis son muy similares a lenguajes como C y PERL. Puede ser utilizado en casi todos los sistemas operativos existentes, permitiendo migrar las aplicaciones de un sistema a otro sin necesidad de realizar cambios en el código. Su rapidez en la ejecución y los bajos requerimientos de consumo en los sistemas donde es desplegado lo hacen uno de los preferidos por los desarrolladores.

Se integra perfectamente a la mayoría de los Sistemas Gestores de Bases de Datos. Su mayor ventaja radica en ser un lenguaje libre, por lo que se convierte en una alternativa de muy fácil acceso, además de poseer una comunidad de desarrolladores que intercambian experiencias, de esta forma cuando se presenta un problema, es muy fácil obtener documentación para darle solución de forma rápida y sin costo

alguno. Quizás una de sus mayores desventajas radica en que promueve la creación de código desordenado, por lo que lo hace muy complejo de mantener. (18)

#### 1.4.1.2 JavaScript v1.10.3

JavaScript es un lenguaje basado en objetos, que se utiliza principalmente para crear páginas web dinámicas y permite el desarrollo de interfaces de usuario mejoradas. Una página web dinámica es aquella que permite la interacción entre el contenido de la misma y el usuario. JavaScript permite incorporar a dichas páginas efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. (19)

### 1.4.2 Lenguaje de modelado UML

Es una herramienta de diseño UML[15] libre y profesional, diseñado para contribuir al desarrollo de software. Soporta los principales estándares como UML, SysML, BPMN[16] y XML. Ofrece un completo conjunto de herramientas a los equipos de desarrollo de software, necesarios para la captura de requisitos, la planificación de software, la planificación de pruebas, el modelado de clases y el modelado de datos (20).

### 1.4.3 Librerías y Marcos de Trabajo:

El proceso de implementación de la solución se efectuará utilizando el marco de trabajo Sauxe, desarrollado por CEIGE, el cual contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo (21).

#### 1.4.3.1 ExtJS v2.2

Es una librería Java Script de código abierto de alto rendimiento para la creación y desarrollo de aplicaciones web dinámicas. Provee interfaces gráficas de usuario que brindan experiencias parecidas o iguales a las que se tienen con aplicaciones de escritorio. Permite la creación de aplicaciones complejas utilizando componentes predefinidos. Es extensible para la gran mayoría de los navegadores, evitando el

[15] **UML**: Lenguaje de Modelado Unificado.
[16] **BPMN:** Business Process Modeling Notation.

tedioso problema de validar el código para cada uno de estos. Entre sus principales ventajas se encuentra el balance entre Cliente-Servidor, distribuyendo la carga de procesamiento en el último, y este al tener menor carga, maneja los clientes de manera más eficiente. La comunicación asíncrona permite el intercambio de información con el servidor sin necesidad de pedirle una acción al usuario, dando la libertad de cargar la información sin que este lo note. (22)

#### 1.4.3.2 Zend Framework 1.9.5

No es más que un marco de trabajo de código abierto para el desarrollo de aplicaciones Web con PHP. Brinda soluciones para construir sitios web modernos, robustos y seguros. Posee un bajo acoplamiento entre sus componentes, lo que posibilita la utilización de los mismos a conveniencia. (…) Brinda una alta abstracción de bases de datos, haciendo extremadamente simple la interacción con estas, sin necesidad de escribir ninguna consulta SQL[17]. Cuenta con módulos para el manejo de ficheros PDF[18], canales RSS, entre otros. Cuenta con clientes para el acceso a WS[19] y robustas clases para la autenticación y el filtrado de entrada; completa documentación y pruebas de alta calidad. (23)

#### 1.4.3.3 Doctrine 0.11

Doctrine es un potente y completo sistema ORM[20] para PHP con un DBAL[21] incorporado que permite exportar una base de datos a sus clases correspondientes y viceversa, o sea, a partir de las clases creadas y siguiendo las especificaciones de ORM, generar las tablas de la base de datos. Se encuentra en la parte superior de una poderosa DBAL. Una de sus principales características es la opción de escribir las consultas de base de datos en un objeto con una propiedad orientada al dialecto SQL llamado Doctrine Query Language (DQL), inspirada en Hibernate (HQL). Esto proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad, sin necesidad de la duplicación de código innecesaria (24).

[17] **SQL:** structured query language.
[18] **PDF:** portable document format
[19] **WS:** Web Services
[20] **ORM:** mapeador relacional de objetos
[21] **DBAL:** capa de abstracción de bases de datos

#### 1.4.3.4 Strophejs

Es una colección de librerías para comunicarse con el protocolo XMPP. Mientras otras librerías e implementaciones se centran en aplicaciones basadas en chat, Strophe tiene una visión más grande es usado para implementar juegos en tiempo real, sistemas de notificaciones, motores de búsquedas, así como para la mensajería instantánea tradicional (25). Es una librería JavaScript dado que este último no facilita conexiones TCP persistentes, esta librería se basa en flujos bidireccionales sobre HTTP sincrónico (BOSH siglas en inglés) para emular la persistencia con estado de conexión en ambos sentidos a un servidor XMPP (26).

#### 1.4.3.5 XMPPHP

XMPPHP es el sucesor de Class.Jabber.PHP que se ha estado elaborando durante años. Esta librería toma ventaja de PHP5, y proporciona una solución elegante, con un enfoque directo.

Algunas de las características que incluye son: (27)

- Permite conectarse a cualquier servidor XMPP 1.0 (Google Talk, Talk LJ, jabber.org, entre otros).
- Soporta el cifrado TLS.
- Varios enfoques de procesamiento XML y soporte de estilos.

### 1.4.4 Herramientas de desarrollo

#### 1.4.4.1 Servidor Web Apache v2.0

Es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows[22], Macintosh[23] y otras, 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 (28).Es una tecnología gratuita, de código abierto y altamente configurable de diseño modular por lo que resulta muy sencillo ampliar sus capacidades. Actualmente existen muchos módulos para Apache que son adaptables a este, y están disponibles para su instalación cuando sean necesarios. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor y es posible configurarlo para que ejecute un determinado script cuando esto suceda (29).

[22] **Windows:** Sistema operativo desarrollado por Microsoft
[23] **Macintosh:** Sistema operativo desarrollado y distribuido por la compañía Apple Inc.

#### 1.4.4.2 PostgreSQL 8.3

Es un sistema de gestión de bases de datos libre basado en el proyecto Postgres, perteneciente a la Universidad de Berkeley. Es un sistema objeto-relacional, que incluye características como la herencia, valores no atómicos (atributos basados en vectores y conjuntos), funciones, disparadores, entre otras. Es altamente extensible, permitiendo el uso de operadores, funciones y tipos de datos definidos por el usuario. Soporta la integridad referencial garantizando la integridad de los datos en la base de datos. PostgreSQL permite realizar múltiples conexiones desde procesos clientes, existiendo un proceso maestro en el servidor que siempre se ejecuta y que está a la espera de nuevas conexiones clientes, de forma tal que cuando alguien se conecta, se inicia un nuevo proceso, asegurando que el cliente y la nueva conexión no necesitan del proceso Postgres original. Una de sus principales características es la alta concurrencia. Esto permite que mientras se realizan cambios en una tabla, otros procesos accedan a la misma sin la necesidad de bloqueos, además de que cada usuario tiene visión de la última modificación. Presenta el inconveniente de que para bases de datos pequeñas su velocidad de respuesta no es muy eficiente en comparación con otras relativamente grandes (30).

#### 1.4.4.3 Visual Paradigm 6.4

Visual Paradigm para UML[24] es una herramienta CASE[25] profesional que soporta el ciclo de vida completo del desarrollo de software. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Esta herramienta también proporciona abundantes tutoriales de UML, demostraciones interactivas y proyectos UML. Soporta UML versión 2.1, permite modelado colaborativo con CVS y Subversion, generación de código, ingeniería inversa, generación de bases de datos (transformación de diagramas entidad-relación en tablas de la base de datos), importación y exportación a ficheros XML, distribución automática de diagramas, entre otras características (31).

#### 1.4.4.4 Mozilla Firefox 8.0

Mozilla Firefox es un navegador de Internet, con interfaz gráfica de usuario desarrollado por la Corporación Mozilla y un gran número de voluntarios externos. Es un navegador Web multiplataforma, que

[24] **UML:** Lenguaje de Modelado Unificado.
[25] **CASE:** Computer Aided Software Engineering.

está disponible en versiones para Microsoft Windows y Linux. Entre sus principales características se encuentran:

Barra de direcciones inteligente, Identificación instantánea del sitio web, Anti-malware, Anti-phishing, Control de contenido, Programas antivirus, Gestor de contraseñas, Actualización automática, Bloqueador de ventanas emergentes, Gestor de descargas, Corrector ortográfico, Restauración de sesiones, Sugerencias de búsqueda y búsqueda en la web integrada (29).

#### 1.4.4.5 NetBeans 7.0.

El IDE NetBeans es un entorno de desarrollo integrado disponible para Windows, Mac, Linux y Solaris. Consiste en un IDE de código abierto y una plataforma de aplicaciones que permiten a los desarrolladores crear rápidamente aplicaciones web, empresariales, de escritorio y aplicaciones móviles utilizando la plataforma Java, así como PHP, JavaScript y Ajax, entre otros (32).

#### 1.4.4.6 Control de Versiones RapidSVN 0.12.0

RepidSVN es un cliente gratuito de código abierto para el sistema de control de versiones Subversion. Maneja ficheros que se almacenan en un repositorio central que registra todos los cambios hechos a los ficheros y directorios. Permite recuperar versiones antiguas de los mismos y examinar las trazas de los datos, así como quién accedió a ellos.

## Conclusiones parciales

Partiendo de los resultados obtenidos al haber realizado el estudio de las técnicas de notificación en tiempo real referenciadas anteriormente, se puede concluir que el protocolo XMPP puede proveer el mecanismo que necesita el componente de notificaciones para enviar notificaciones en tiempo real través del servidor Openfire. Se utilizará el modelo de desarrollo orientado a componentes que define CEIGE, el cual provee durante el proceso de desarrollo, una guía que incluye las tareas, actividades, fundamentos y herramientas necesarias que ayuden a garantizar un software como resultado final de mejor calidad. Dentro de las tecnologías a utilizar se encuentran algunas definidas por el centro como, el servidor Apache y el gestor de bases de datos PostgreSQL, la herramienta de diseño Visual Paradigm, los lenguajes de programación PHP y JavaScript, los cuales son utilizados en los marcos de trabajo ZendFramework y EXT JS respectivamente, además del marco de trabajo Doctrine. El autor define además otras librerías que son necesarias para lograr comunicación entre el servidor Openfire y el lenguaje PHP, es el caso de XMPPHP y Strophe para que el servidor de mensajería instantánea se pueda comunicarse con el servidor apache y el lenguaje JavaScript respectivamente.

# Capítulo II: PROPUESTA DE SOLUCIÓN

## Introducción

En este capítulo se desarrolla la propuesta de solución para el componente de notificaciones la cual comienza con la identificación y descripción de los requisitos funcionales y no funcionales además muestra algunos artefactos y resultados generados por el modelo de desarrollo y que fueron obtenidos durante la fase de análisis y diseño del componente. Además se de una breve descripción de los patrones de diseño y arquitectónicos utilizados en la solución.

### 2.1 Modelo de dominio o conceptual.

Puede utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema. El modelo de dominio es utilizado por el analista como un medio para comprender el sector de negocios al cual el sistema va a servir (14).

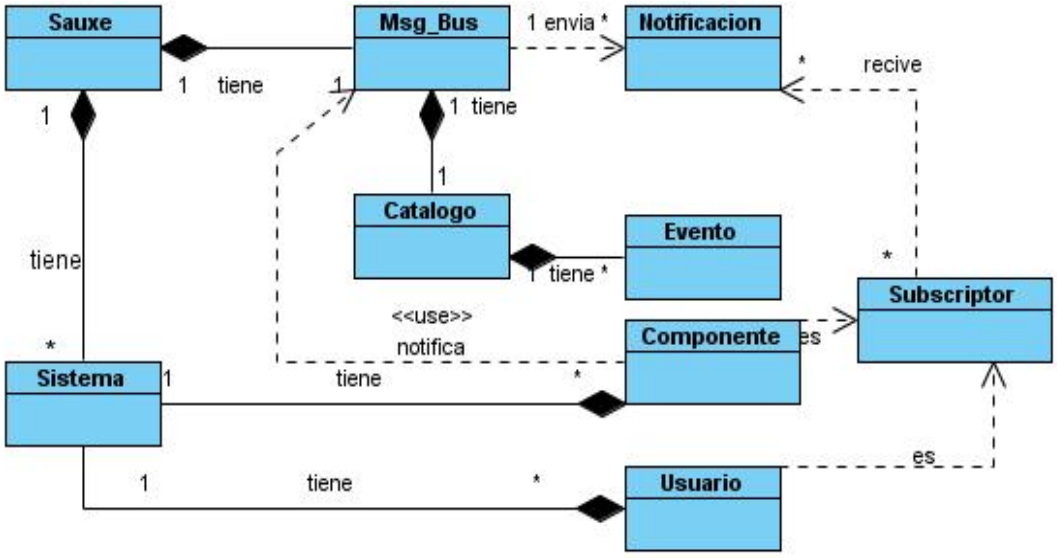



*Figura 3 Modelo conceptual.*

**Sauxe:** Marco de trabajo que contiene sistemas y a Msg_Bus.

**Sistema:** Forma parte de Sauxe.

**Msg_Bus:** Nombre del componente que gestiona y envía las notificaciones a un subscriptor.

**Subscriptor:** Puede ser un componente o un usuario y recibe las notificaciones.

**Usuario:** Pertenece a un sistema, juega una función determinada en el mismo y es un subscriptor.

**Componente:** Forma parte de un sistema y puede notificar a Msg_Bus.

**Catálogo:** Forma parte de Msg_Bus y contiene la relación de todos los eventos y los subscriptores relacionados a ellos.

**Evento:** Forma parte del catálogo.

**Notificación:** Es enviada por Msg_Bus y recibida por uno o varios subscriptores.

## 2.2 Requisitos de software

Los requisitos son un punto clave en el desarrollo de las aplicaciones informáticas, permiten una comunicación efectiva entre los usuarios y el equipo de desarrollo, con el objetivo de llegar a un entendimiento de lo que hay que realizar, y describen el comportamiento esperado en el software una vez desarrollado. Gran parte del éxito de un proyecto de software radicará en la identificación de las necesidades del negocio.

Los requisitos de software se clasifican en funcionales y no funcionales. Los requisitos funcionales describen qué es lo que el sistema debe hacer para dar soporte a las funciones y objetivos del usuario. Los requisitos no funcionales imponen restricciones de cómo los requisitos funcionales deben ser implementados (33).

### 2.2.1 Requisitos funcionales

El proceso de definición de los requisitos funcionales es dirigido por el equipo de desarrollo el cual teniendo en cuenta la información brindada por el cliente realiza el dicho proceso. Posteriormente a partir de la información que brinda el cliente se elaboran los documentos de especificación de requisitos y

finalmente se valora y valida la información en busca de errores, inconsistencias o faltas para evitar omitir algún requerimiento del cliente (34).

### 2.2.2 Técnicas de captura de requisitos

A continuación se relacionan y se explica brevemente en qué consisten algunas técnicas de captura de requisitos (35).

- **Entrevistas y cuestionarios:** permite al equipo de desarrollo interpretar las necesidades del cliente. Pueden ser lo mismo provechosos o no, ya que dependen de las habilidades del entrevistador y los entrevistados para obtener información con la mayor calidad posible.
- **Desarrollo conjunto de aplicaciones:** proviene del inglés *JointApplicationDevelopment* (JAD) y es una técnica exploratoria que, aunque puede ser muy costosa por la cantidad de personal que involucra, es muy popular, ya que incluye a los usuarios como participantes activos en el proceso de desarrollo de sistemas. De ésta técnica puede surgir una declaración bastante fiel de los requisitos.
- **Tormenta de ideas:** consiste en la acumulación de ideas sin prejuicios y valoraciones que puedan descartarlas y aunque no evidencia los detalles concretos que se pueden necesitar del sistema, es muy común en los comienzos del proceso de ingeniería de requisitos.
- **Mapas conceptuales:** Es otra técnica bastante común, usada para la representación gráfica de las ideas y sus relaciones.
- **Escenarios:** valiosos medios para proporcionar contexto a las exigencias del consumidor. Proporcionan un marco para preguntas sobre tareas del usuario permitiendo preguntas "*y si*" y "*cómo se hace esto*".
- **Comparación de terminología:** se utiliza para complementar otras técnicas, pero consiste en identificar todos los términos con los que se trabajará durante el desarrollo del sistema. Para su correcta utilización, es necesario identificar el uso de términos diferentes para los mismos conceptos (correspondencia), misma terminología para diferentes conceptos (conflictos) o cuando no hay concordancia ni en el vocabulario ni en los conceptos (contraste).
- **Reuniones:** el propósito de éstas es intentar alcanzar un efecto aditivo por el que un grupo de gente puede obtener más penetración en los requisitos del software que trabajando individualmente. Ellos pueden inspirarse y refinar las ideas que pueden ser difíciles de traer a la superficie usando entrevistas. Otra ventaja es que dejan a los stakeholders9 reconocer donde hay requisitos en conflicto.

Cuando se aplica bien, esta técnica puede resultar en un rico y constante sistema de requisitos que difícilmente sería realizable de otro modo.

- **Observación:** los ingenieros de software aprenden sobre las tareas del usuario sumergiéndose en la observación de cómo los usuarios obran recíprocamente con su software. Estas técnicas son relativamente costosas, pero son instructivas porque ilustran que muchas tareas del usuario y procesos del negocio son demasiado sutiles y complejos para que sus agentes los describan fácilmente.

### 2.2.3 Validación de requisitos

Los requisitos una vez definidos necesitan ser validados. La validación de requisitos tiene como misión demostrar que la definición de los requisitos define realmente el sistema que el usuario necesita o el cliente desea. Es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de requisitos son correctos. Pocas son las propuestas existentes que ofrecen técnicas para la realización de la validación y muchas de ellas consisten en revisar los modelos obtenidos en la definición de requisitos con el usuario para detectar errores o inconsistencias. Aun así, existen algunas técnicas que pueden aplicarse para ello: (35)

- **Walk-throughs o Revisiones:** Está técnica consiste en la lectura y corrección de la documentación completa o del modelado de la definición de requisitos. Con ello solamente se puede validar la correcta interpretación de la información transmitida. Más difícil es verificar la consistencia de la documentación o información faltante.
- **Auditorías:** La revisión de la documentación con esta técnica consiste en un chequeo de los resultados contra una lista de lista de chequeos predefinida o definida a comienzos del proceso, es decir sólo una muestra es revisada.
- **Matrices de trazabilidad:** Esta técnica consiste en marcar los objetivos del sistema y chequearlos contra los requisitos del mismo (Durán, Bernáldez, Ruíz &Toro, 1999). Es necesario ir viendo que objetivos cubre cada requisito, de esta forma se podrán detectar inconsistencias u objetivos no cubiertos.
- **Prototipos:** Algunas propuestas se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura

de la interfaz del sistema con el usuario (Olsina, 1999). Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final (36).

### 2.2.4 Técnicas de captura y validación de requisitos seleccionadas

Durante el proceso de ingeniería de requisitos se propone, como Técnicas de Captura de requisitos: realizar varias **reuniones**, en las que mediante **mapas conceptuales**, **tormenta de ideas** y la idealización de varios posibles **escenarios** se pueda llegar a la conclusión de cuáles serían los requisitos de software del componente de gestión de notificaciones.

Con la participación del equipo de desarrolladores del componente de notificaciones y el cliente interesado en la solución, a través de los métodos de captura de requisitos mencionados anteriormente se obtuvieron 10 requisitos no funcionales los cuales se listan a continuación:

- RF1 Gestionar el proceso de conexión con el servidor XMPP.
- RF2 Crear un catálogo con las aplicaciones que se encuentras suscritas a cada evento y con los roles que escuchan los mismos.
- RF3 Autenticar usuario al servidor XMPP a través del marco de trabajo.
- RF4 Gestionar a través de los roles, los códigos de mensaje recibe cada rol.
- RF5 Gestionar las notificaciones que envía cada sistema.
- RF6 Gestionar las notificaciones que recibe cada sistema.
- RF7 Mostrar historial de mensajes de un usuario.
- RF8 Eliminar mensaje del buzón de un usuario.
- RF9 Enviar notificación a subsistema.
- RF10 Enviar notificación a usuario.

Teniendo en cuenta la importancia del proceso de Validación de Requisitos se propone utilizar varias estas como las **Revisiones** en las que estén presentes la gran mayoría del personal involucrado en la solución, tanto por parte del cliente como de los desarrolladores y se propone la creación de los **Prototipos de Interfaz de Usuario** y la **Generación de casos de prueba** para probar cada requisito identificado. A continuación se muestra la descripción del requisito funcional Autenticar usuario al servidor XMPP a través del marco de trabajo, las demás descripciones de requisitos funcionales se pueden encontrar en los anexos del documento.

#### 2.2.4.1 Especificación de Requisitos.

**Requisito funcional:** Autenticar usuario al servidor XMPP a través del marco de trabajo**.**

*Tabla 1 Descripción del requisitos funcional Autenticar usuario al servidor XMPP a través del marco de trabajo.*

| | |
|---|---|
| **Precondiciones**: | El usuario no debe estar autenticado en el sistema y debe estar registrado tanto en el servidor XMPP y como en ACAXIA. |
| **Descripción** | |
| En este requisito comienza cuando el usuario se autentica en el marco de trabajo. | |
| **Flujo de eventos** | |
| **Flujo básico de eventos** | |
| 1 El usuario introduce el usuario y contraseña de ACAXIA. | |
| 2 Debe dar clic en el botón **Aceptar.** | |
| **Post-Condiciones** | |
| 1 El sistema muestra un mensaje: ¨Sistema de notificaciones Listo¨. | |
| **Flujos alternativos 1.a** | |
| 1 El usuario no introduce correctamente el usuario o contraseña. | |
| 2 El sistema señala que los datos introducidos no son correctos. | |
| 3 Volver al paso 1 del flujo básico de eventos. | |
| **Validaciones** | |
| 1 Se validan los datos según lo establecido en el Modelo conceptual | |
| **Conceptos** | |
| Visible en la interfaz: usuario y contraseña | |

### 2.2.5 Requisitos no funcionales

Los requerimientos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con toda la funcionalidad requerida, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación.

Como se ha mencionado con anterioridad, 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 desarrollado en el mismo. Por tanto los requisitos no funcionales con los que debe cumplir la aplicación a desarrollar fueron establecidos por el centro al inicio del proceso de desarrollo, a continuación se describen los más importantes. (37)

#### 2.2.5.1 Software

**Para el cliente:**

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

**Para el servidor**:

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

#### 2.2.5.2 Rendimiento

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

#### 2.2.5.3 Seguridad

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

#### 2.2.5.4 Hardware

**Para el servidor Openfire:**

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

**Para el cliente:**

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

## 2.3 Diagrama de clases del diseño web

El diagrama de clases del diseño describe la estructura del sistema, mostrando sus clases, asociaciones, atributos, métodos y sus relaciones entre ellos (37). En la **Figura 14** se muestra el diagrama de clases del diseño de los escenarios gestionar notificaciones emitidas por sistemas, donde se representan las principales clases, operaciones y relaciones que se necesitan para darle cumplimiento a los requisitos funcionales relacionados con la gestión de las notificaciones que emiten los sistemas. Las clases GestSistEmiten.js y GestSistEmiten.phtml conforman la capa arquitectónica de presentación. La clase GestSubsistemaEmitenController solo maneja la comunicación entre la vista y el modelo, la clase Model es la encargada de la lógica del negocio, implementando funcionalidades que garantizan el cumplimiento de los requisitos identificados.

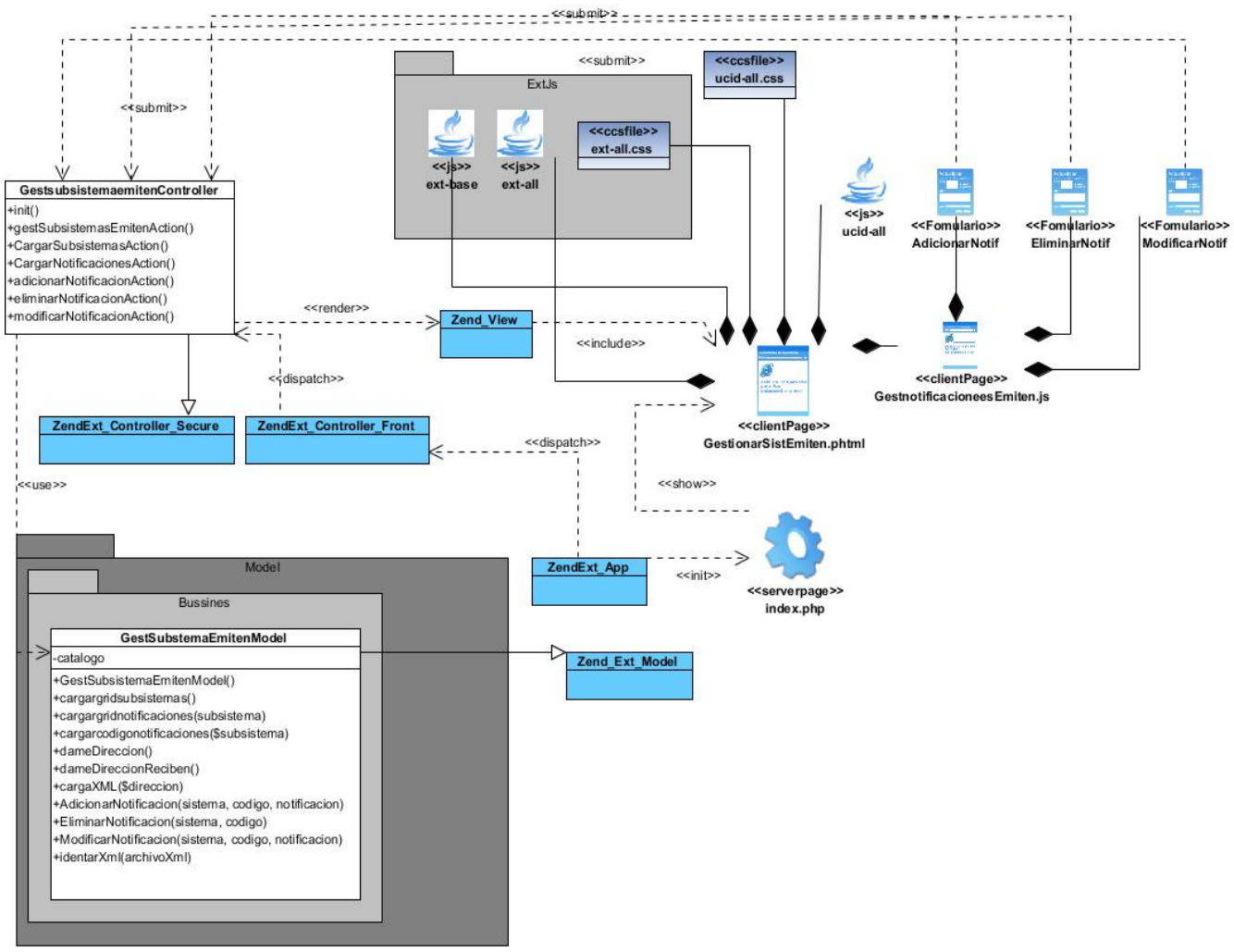



*Figura 4 Diagrama de clases del diseño gestionar notificaciones emitidas por sistemas.*

En la **Figura 15** se muestra el diagrama de clases del diseño del escenario Gestionar notificaciones recibidas por sistemas, en la cual se le da solución a los requisitos funcionales relacionados con la gestión de las notificaciones recibidas por sistemas, se representan las principales clases, operaciones y relaciones que se necesitan para darle cumplimiento a estos requerimientos, donde la capa arquitectónica de presentación está formada por las clases GestSistemasReciben.js y SistReciben.phtml. La comunicación entre la vista y el modelo es realizada por la clase GestSubsistemasrecibenController y la clase

*GestSubsistemasRecibenModel* es la encargada de implementar las funcionalidades que le dan cumplimiento a los requisitos identificados.

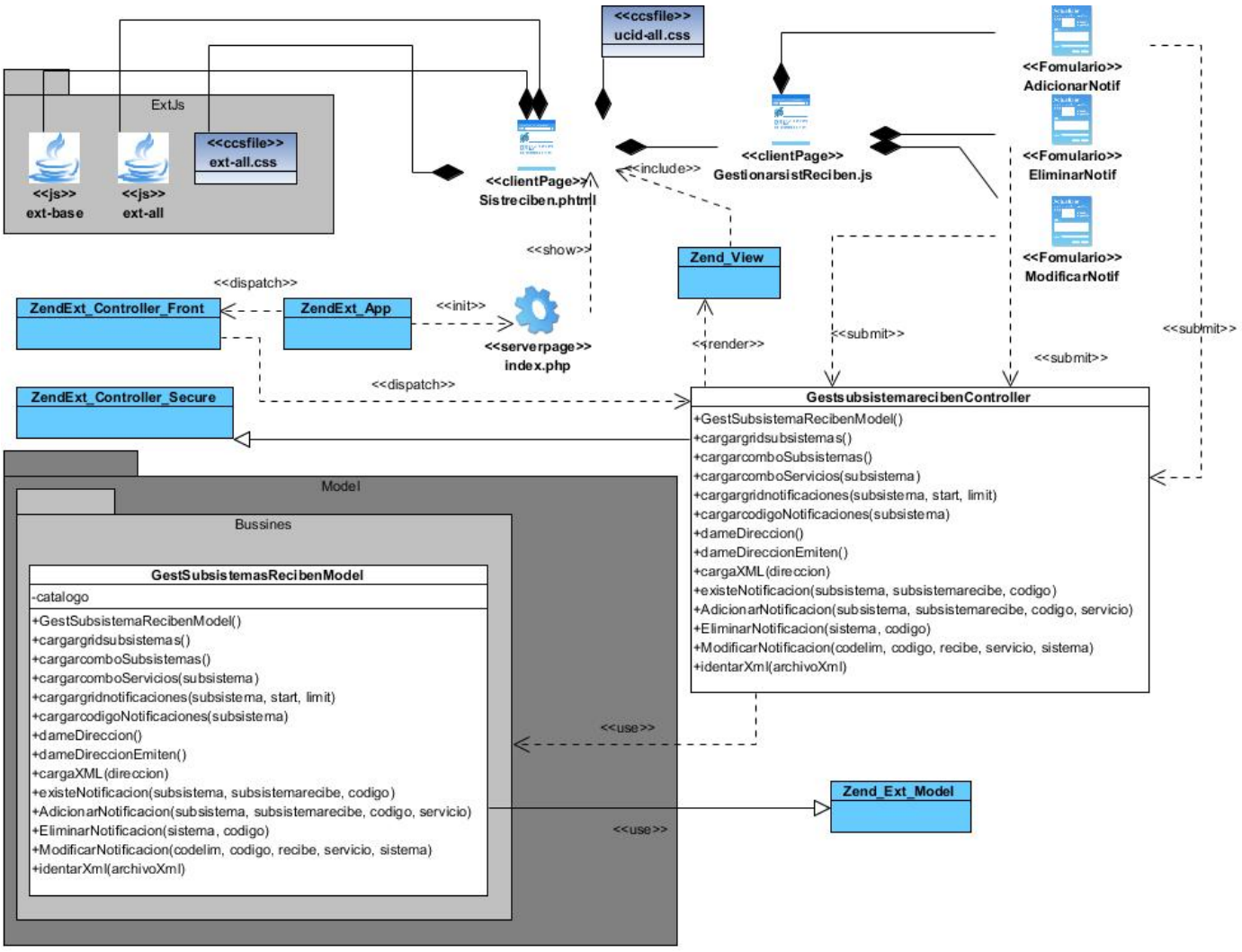



*Figura 5 Diagrama de clases del diseño gestionar notificaciones recibidas por sistemas.*

En la ***Figura 16*** se muestra el diagrama de clases del diseño del escenario Eliminar notificación del buzón. Las clases *Buzon.js* y *Buzon.phtml* conforman la capa arquitectónica de presentación. La comunicación entre la vista y el modelo es realizada por la clase *BuzonController*, la encargada de la lógica del negocio es la clase *Model* y encargada del acceso a los datos se encuentra en *DatBuzonNotif*, así como la clase Base de la cual extiende.

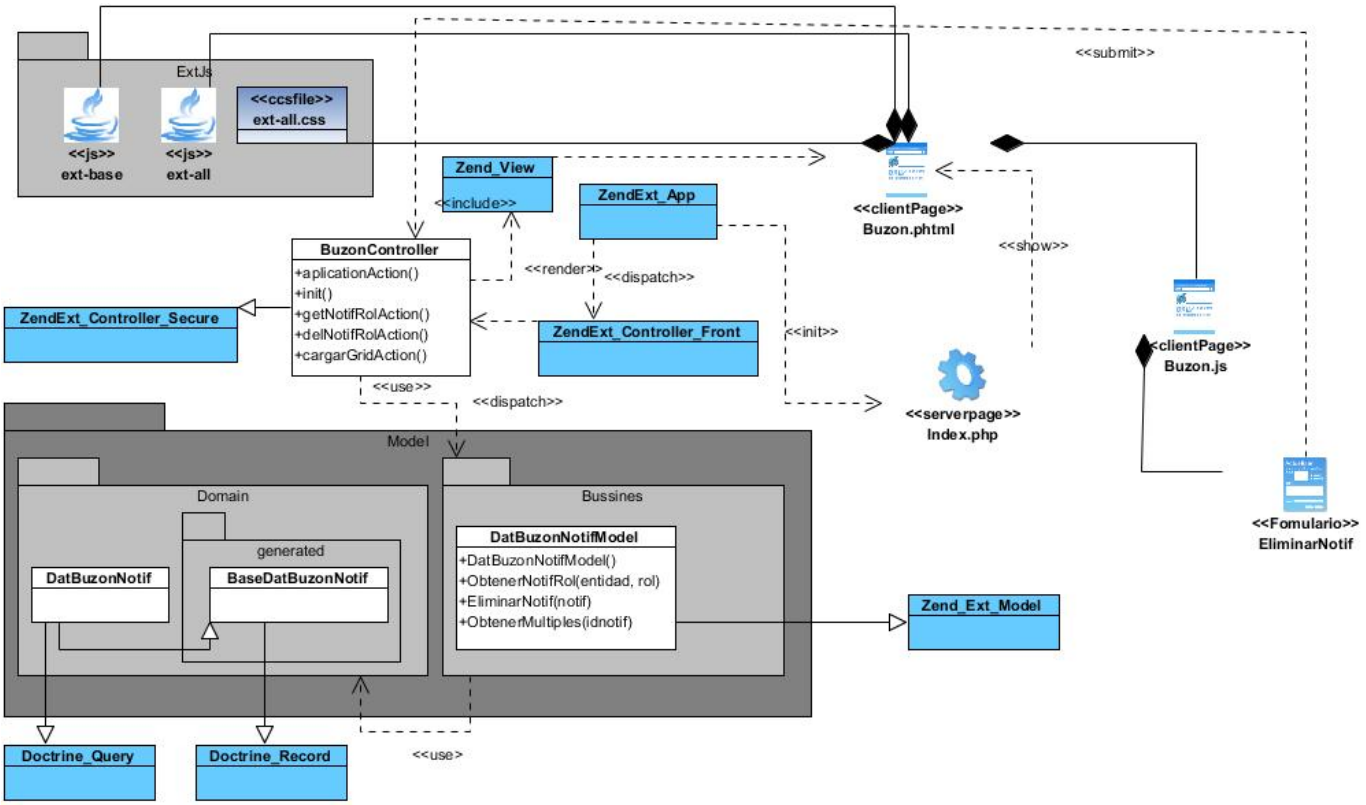



*Figura 6 Diagrama de clases del diseño gestionar buzón de mensajes.*

En la **Figura 17** se muestra el diagrama de clases del diseño del escenario Gestionar rol, en la cual se le da solución a los requisitos funcionales relacionados con gestionar los roles que tiene asignado cada notificación y se ejecuta también el requisito de enviar mensaje a usuarios. Las clases GestionarRoles.js y GestionarRoles.phtml conforman la capa arquitectónica de presentación. La comunicación entre la vista y el modelo es realizada por la clase GestionarController, la encargada de la lógica del negocio es la clase Model y encargada del acceso a los datos se encuentra DatMessagebus, así como la clase Base de la cual extiende.

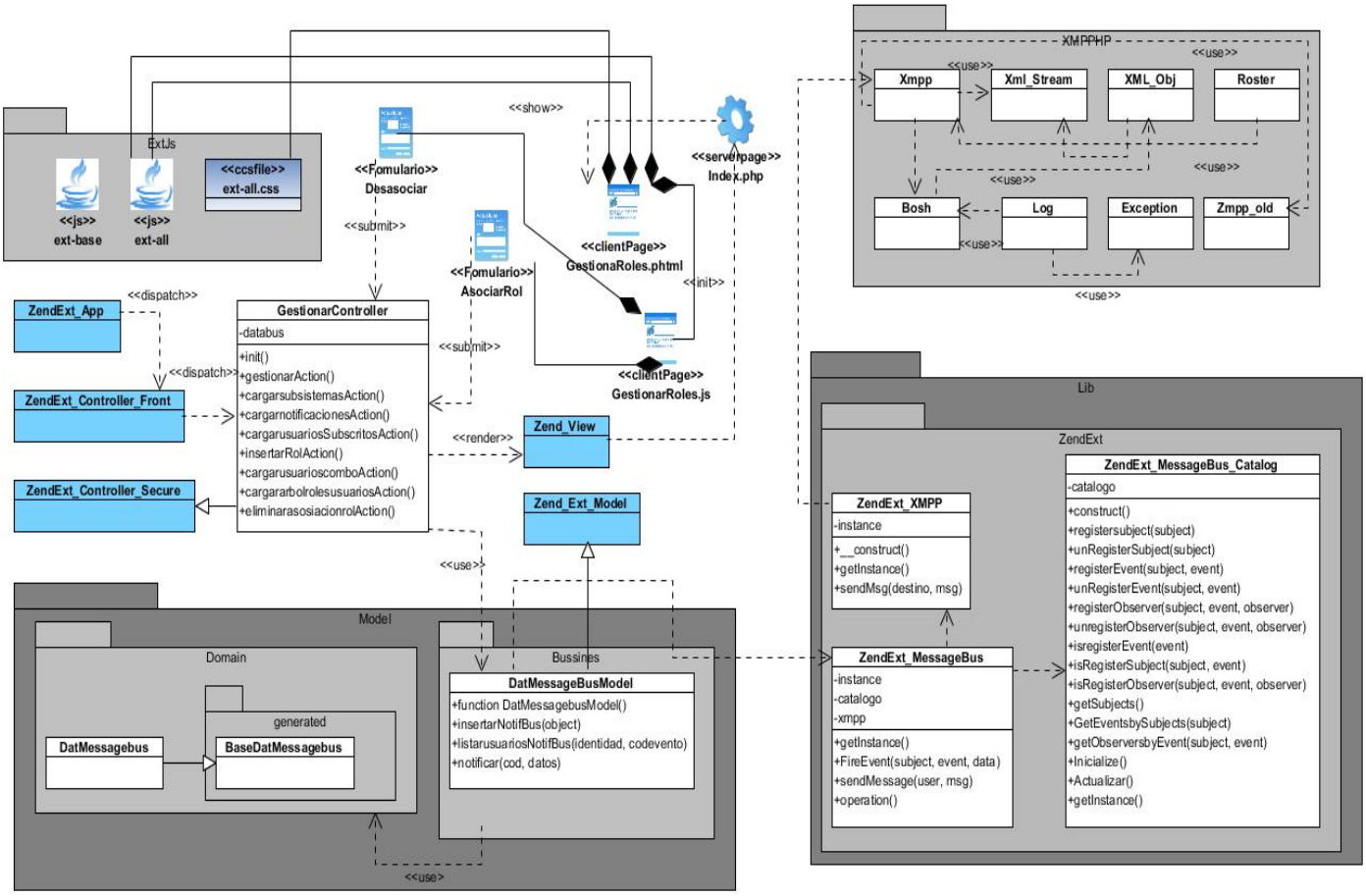


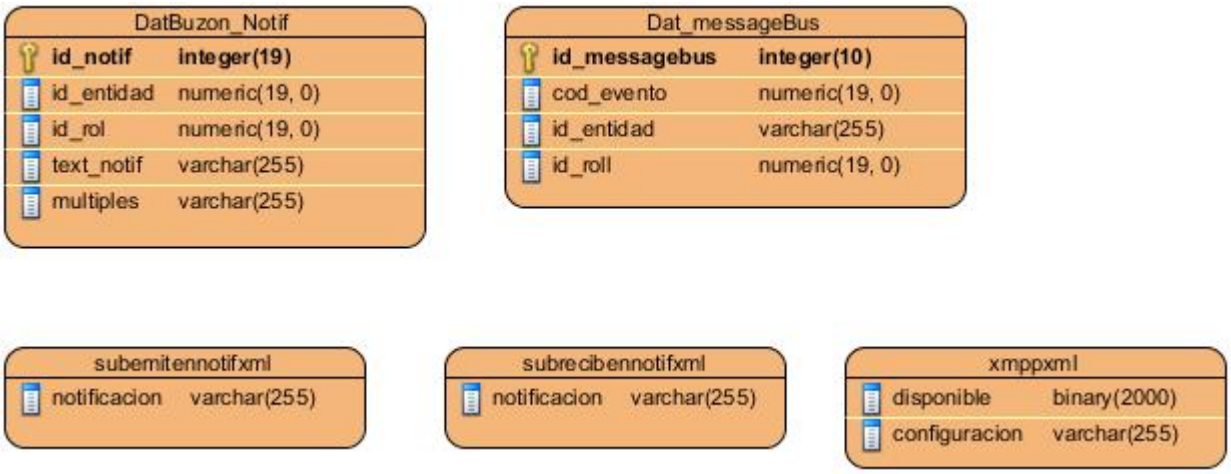

*Figura 7 Diagrama de clases del diseño gestionar rol.*

Los modelos de datos (MD), son el mecanismo formal para representar los datos de manera general y sistemática, establece una relación entre el mundo real y la información almacenada físicamente en la base de datos (BD) (38). El objetivo de construir un MD es identificar y representar las tablas (entidades) de importancia para el funcionamiento del negocio, sus propiedades (atributos), y la forma en que estas tablas se comunican entre sí (relaciones). Este modelo se desarrolla para facilitar el diseño de la BD y mostrar los datos que contendrá el sistema.

## 2.4 Modelo de Datos

El modelo de datos propuesto en la solución cuenta con un total de 2 tablas las cuales no están relacionadas entre ellas. La tabla DatBuzonNotif almacena todas las notificaciones emitidas por los

sistemas y es la fuente de datos cuando un usuario pide listar todas las notificaciones que ha recibido. La tabla DatMessagebus almacena la información que tiene que ver con los roles que tiene asociado un evento en una entidad. Los ficheros .XML subemitennotif.xml y subrecibennotif.xml se encargan de guardar la información de las notificaciones que es capaz de emitir y recibir un subsistema respectivamente. El fichero .xml xmpp.xml guarda la información del servidor XMPP, si está o no funcionando y los datos para acceder al mismo desde la librería XMPPHP.




Patrones.

Un patrón es una regla que consta de tres partes, y expresa una relación entre un contexto problema y una solución. Por lo general, sigue el siguiente esquema:

- ***Contexto*:** situación de diseño en la que aparece un problema de diseño.
- ***Problema*:** conjunto de fuerzas que aparecen repetidamente en el contexto.
- ***Solución*:** configuración que equilibra estas fuerzas. Abarca:
  - Estructura con componentes y relaciones.
  - Comportamiento a tiempo de ejecución: aspectos dinámicos de la solución, como la colaboración entre componentes, la comunicación entre ellos, etc.

Un patrón, al igual que un estilo, se enfoca en imponer transformaciones a la arquitectura de un software, mientras que soporta el desarrollo, mantenimiento y evolución de sistemas complejos (39).

## 2.5 Patrones de diseño.

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software, por lo que es de suma importancia aplicarlos en la construcción del diseño de un sistema. Con su uso, se pretende establecer un lenguaje común entre los programadores, contribuir a la reutilización, ahorrar tiempo en la implementación y obtener un producto con calidad. Son además la solución a determinado problema de diseño que se presente en el desarrollo de un sistema. Entre sus características debe incluir la habilidad de ser reutilizable y aplicable a diferentes problemas de diseño en distintas circunstancias.

Brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Se deben tener presente los siguientes elementos de un patrón: su nombre, el problema (cuándo aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios) (40).El patrón es, resumiendo, al mismo tiempo una cosa que tiene su lugar en el mundo, y la regla que dice cómo crear esa cosa y cuándo crearla (41).

### 2.4.1 Patrones GoF

Los patrones Gang Of Four (GoF), son patrones de diseño publicados en el libro *Design Patterns: Elements of Reusable Object-Oriented Software* por Gamma, Helm, Jonson y Vlissides conocidos por Banda de los cuatro. Están divididos fundamentalmente en tres grandes grupos:

- **Creacionales:** concierne al proceso de creación de objetos.
- **Estructurales:** tratan la composición de clases y/o objetos.
- **De Comportamiento:** caracterizan las formas en las que interactúan y reparten responsabilidades las distintas clases u objetos.

De los patrones GoF existentes a continuación se describen los que fueron utilizados para el desarrollo del componente y de qué forma beneficiaron al mismo:

**Fachada**

El patrón de diseño estructural Fachada tiene como propósito simplificar el acceso a un conjunto de objetos proporcionando uno que todos los clientes pueden usar para comunicarse con el conjunto. Tiene como objetivo minimizar las comunicaciones y dependencias entre componentes. Normalmente sólo hace

falta un objeto Fachada, por lo cual suele implementarse como Singleton. A continuación en la ***Figura 8*** se muestra un ejemplo de cómo funciona el patrón fachada (39).

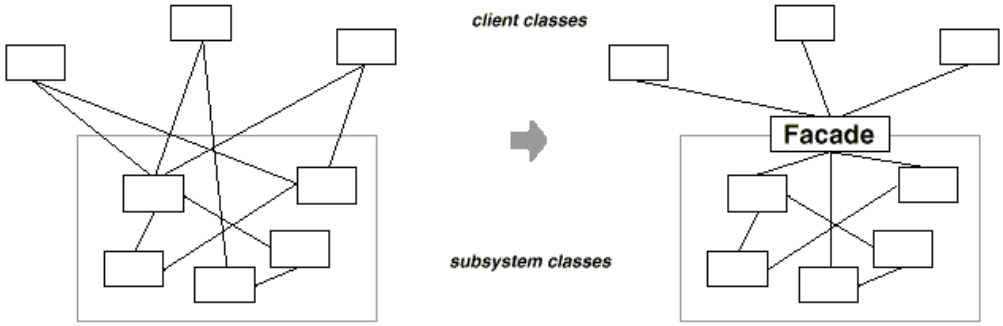



*Figura 8 Representación gráfica del patrón Fachada.*

Este patrón de diseño arquitectónico se usa en la solución del componente, específicamente en la clase **ZendExt_Xmpp**, como se puede apreciar en la figura 19. Esta clase tiene la responsabilidad de proveer un acceso global a las funcionalidades de la librería **XMPPHP**, la cual es la encargada de establecer la relación con el protocolo XMPP a través del servidor Openfire. Al utilizar el patrón fachada en el componente de notificaciones en el caso planteado anteriormente, se garantiza dependencia de las clases del componente y funcionalmente no implicaría ningún problema en caso que cambie la librería que se utiliza actualmente.

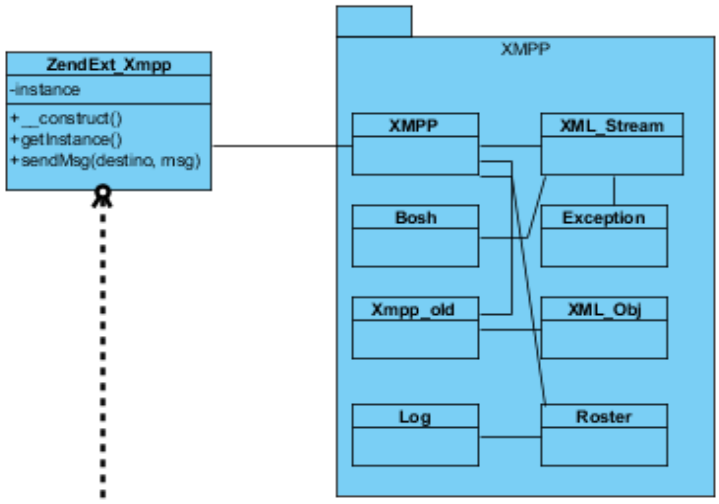



*Figura 9 Uso del patrón Singlenton en el componente.*

**Singlenton o Solitario**

Es un patrón creacional que garantiza que exista una instancia única para una clase y proporciona un punto de acceso global a ella (39). Este patrón de diseño se usa en el componente y las clases que lo implementan son **ZendExt_Xmpp, ZendExt_MessageBus** y **ZendExt_MessageBus_Catalog** como se puede observar en la ***Figura 20*****.** Estas clases manejan datos que resultan imprescindibles para el funcionamiento del componente y deben estar accesibles desde cualquier clase mismo, además dichos datos no deben ser duplicados ya que traería ambigüedad y mal un funcionamiento del componente y de que esto no suceda se encarga el patrón de diseño Singlenton.

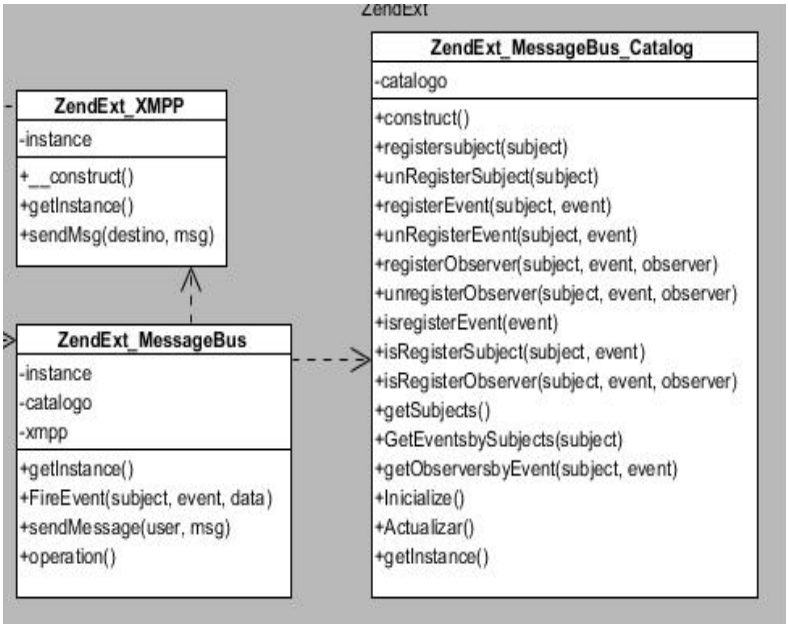



*Figura 10 Clases que implementan patrón Singlenton en el componente de notificaciones.*

**Patrones GRASP**

Otros patrones de diseño son los GRASP (General Responsibility Assignment Software Patterns, Patrones Generales de Software para Asignación de Responsabilidades), y fueron definidos por primera vez por Craig Larman en Applying UML & Patterns: Introduction to Object-Oriented Analysis & Design, & Iterative Development (42). Estos patrones describen los principios fundamentales de la asignación de responsabilidades a objetos, expresada en forma de patrones. Existen nueve patrones GRASP los cuales son: Experto, Creador, Alta cohesión, Bajo acoplamiento, Controlador, Polimorfismo, Fabricación pura, Indirección y No hables con extraños**.**

A continuación se mencionan y describen los patrones de diseño que se utilizaron durante el desarrollo del componente y de qué forma beneficiaron al mismo:

**Experto**

Asigna una responsabilidad al experto en información: la clase que posee la información necesaria para cumplir con su responsabilidad, y enfoca sus funcionalidades a manejar solo esos datos (42). Se puede decir que al usar este patrón en el componente específicamente en las clases controladoras y en las del modelo es posible que el diseño de las mismas pueda tener un bajo acoplamiento, encapsulación y reutilización, así como mayor claridad.

**Alta cohesión**

Este patrón determina, que la información almacenada en una clase debe ser coherente y estar relacionada con esta, en mayor medida y enfocada en sus responsabilidades (42). Al realizar un diseño donde las clases del componente mantengan una alta cohesión como por ejemplo las clases **controladoras**, es posible ganar en claridad y facilidad a la hora de entender el diseño, además de simplificar el mantenimiento y soportar mayor capacidad de reutilización.

**Bajo acoplamiento**

Consiste en tener las clases lo menos relacionadas entre sí, para que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión en las demás, potenciando la reutilización y disminuyendo la dependencia entre las clases (42).

**Controlador**

Asigna la responsabilidad de administrar un mensaje de eventos del sistema a una clase (42).

## 2.6 Patrones arquitectónicos

La selección de un patrón arquitectónico es una decisión a tomar fundamental para el desarrollo de un sistema. Ya que dichos patrones expresan, según Buschmann, el esquema de organización estructural fundamental, proveen un conjunto de subsistemas predefinidos, especifican sus responsabilidades e incluyen reglas y pautas para la organización de las relaciones entre ellos. Buschmann propone además que son plantillas para arquitecturas de software concretas, que especifican las propiedades estructurales de una aplicación y tiene un impacto en la arquitectura de subsistemas (39).

### 2.4.1 Patrón MVC (Modelo-Vista-Controlador)

Divide una aplicación interactiva en 3 componentes. El modelo contiene la información central y los datos. Las vistas, despliegan información al usuario. Los controladores capturan la entrada del usuario. Las vistas y los controladores constituyen la interfaz de usuario (39).

- **Modelo:** Está compuesto por los datos, reglas de negocio y las funcionalidades correspondientes para la comunicación con el persistidor de datos Doctrine.
- **Controlador:** Gestiona las entradas del usuario, utilizando el marco de trabajo ZendFramework.
- **Vista:** Muestra la información al usuario del modelo al usuario. Esto es posible mediante el marco de trabajo de la capa de presentación EXTJS.

El hecho de utilizar este patrón arquitectónico el cual se encuentra en la arquitectura base del marco de trabajo Sauxe provee al componente de flexibilidad y facilidad a la hora de hacer futuros cambios.

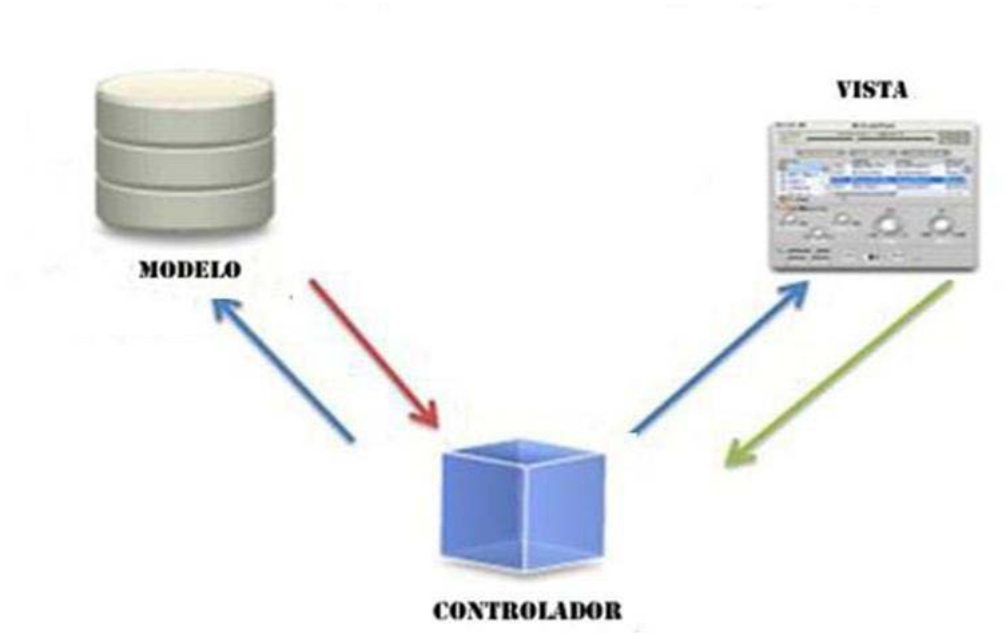



*Figura 11 Representación gráfica del patrón M-V-C.*

## Conclusiones parciales

Se concluye este capítulo dejando claro los requerimientos funcionales que debe cumplir el componente, los cuales a través de las técnicas de captura y validación de los mismos y con la ayuda del modelo conceptual permiten tener una idea y alcance del componente. Con este fin también se elaboraron los diagramas de clases del diseño en los que se plantean las clases a implementar y la relación entre ellas incluyendo los patrones arquitectónicos y de diseño propuestos. Con todos estos elementos y en conjunto con el modelo de datos físicos, se sientan las bases para la próxima fase, donde se implementará y se le realizarán las pruebas de software al componente propuesto.

# Capítulo III: IMPLEMENTACIÓN Y PRUEBA

## Introducción

En el presente capítulo se aborda los estándares de codificación empleados durante la implementación del componente de notificaciones para garantizar un buen entendimiento y legibilidad del código. Además se muestran los artefactos generados en la fase de implementación y prueba por el modelo de desarrollo orientado a componentes utilizado, es el caso del diagrama de componentes y de despliegue. Posteriormente se realiza la validación del diseño a través de métricas, en aras de verificar el cumplimiento de los atributos de calidad definidos por las propias métricas seleccionadas. Se describen las pruebas efectuadas al software que tienen como objetivo detectar y corregir el máximo de errores en el sistema, antes de su entrega al cliente.

### 3.1 Estándares de codificación.

Definido por la Real Academia de la Lengua Española como, “sirve como tipo, modelo, norma, patrón o referencia”, un estándar constituye una serie de lineamientos técnicos detallados, destinados a establecer uniformidad. En el caso específico de los estándares de codificación para los lenguajes de programación, las convenciones de codificación son pautas de 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 completo comprende todos los aspectos de la generación de código, asegurando que todos los programadores del proyecto trabajen de forma coordinada. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento (43). Para el desarrollo de este trabajo se utilizó el estándar de codificación definido por el departamento de Tecnologías perteneciente a CEIGE, el cual incluye normas y buenas prácticas de implementación para los lenguajes PHP y JavaScript, el mismo se describe a continuación y está referenciado del documento 0000073547-Sauxe-Estilo de código para PHP y JavaScript.pdf.

### 3.1.1 Estándares de codificación para lenguaje PHP

#### 3.1.1.1 Declaración de clases.

Las Clases deben ser nombradas de acuerdo a la convención de nombres de ZendFramework. Cada clase debe contener un bloque de documentación acorde con el estándar de PHPDocumentor. La llave "{" deberá escribirse siempre en la línea debajo del nombre de la clase. Todo el código contenido en una clase debe ser separado con cuatro espacios. Únicamente una clase está permitida por archivo PHP.

#### 3.1.1.2 Declaración de Funciones y Métodos.

Las Funciones deben ser nombradas de acuerdo a las convenciones de nombrado de ZendFramework. Los métodos dentro de clases deben declarar siempre su visibilidad usando un modificador “prívate”, “protected”, o “public”. Como en las clases, la llave "{" debe ser escrita en la línea siguiente al nombre de la función, no está permitido un espacio entre el nombre de la función y el paréntesis de apertura para los argumentos. Las funciones de alcance global no están permitidas.

#### 3.1.1.3 Sentencias de Control.

**If/Else/Elseif**

Las sentencias de control basadas en las construcciones “if” y “elseif” deben tener un solo espacio en blanco antes del paréntesis de apertura del condicional y un solo espacio en blanco después del paréntesis de cierre. Dentro de las sentencias condicionales entre paréntesis, los operadores deben separarse con espacios, por legibilidad. Se aconseja el uso de paréntesis internos para mejorar la agrupación lógica en expresiones condicionales más largas. La llave de apertura "{" se escribe en la misma línea que la sentencia condicional, teniendo en cuenta que hay un espacio en blanco después del paréntesis de cierre. La llave de cierre "}" se escribe siempre en su propia línea y con un espacio en blanco después de ella. Cualquier contenido dentro de las llaves debe separarse con cuatro espacios en blanco.

**For**

Las sentencias con ciclo “for” deben tener un solo espacio en blanco antes del paréntesis de apertura del ciclo y un solo espacio en blanco después del paréntesis de cierre. Lleva espacio en blanco después de la coma y del punto y coma.

### Foreach

Las sentencias con ciclo “foreach” deben tener un solo espacio en blanco antes del paréntesis de apertura del ciclo y un solo espacio en blanco después del paréntesis de cierre. Para mejor legibilidad, antes y después de los operadores de asignación "=>" se deben dejar espacios en blanco de forma tal que queden hasta que se alineen. Si hay un solo operador "=>" lleva un espacio en blanco antes y después de este.

### Switch

Las declaraciones de control escritas con la declaración "switch" deben tener un único espacio en blanco antes del paréntesis de apertura del condicional y después del paréntesis de cierre. Todo contenido dentro de una declaración "switch" debe separarse usando cuatro espacios. El contenido dentro de cada declaración "case" debe separarse usando cuatro espacios adicionales.

### While& Do while

Las sentencias con ciclo “while” y “do while” deben tener un solo espacio en blanco antes del paréntesis de apertura del ciclo y un solo espacio en blanco después del paréntesis de cierre.

### 3.1.2 Estilo de Código para Java Script

**Sangría**

Utilizar un guión, de 2 espacios, sin pestañas. No hay espacios en blanco al final.

**CamelCasing**

La diferencia de las variables y funciones definidas en PHP Drupal, multi-variables de las palabras y funciones en JavaScript, es su tipo de estilo de escritura: lowerCamelCased. La primera letra década variable o función debe estar en minúscula, mientras que la primera letra de palabras posteriores debe ser mayúscula.

**Estructuras de control**

Estas incluyen if, for, while, switch. Las instrucciones de control deben tener un espacio entre la palabra clave de control y paréntesis de apertura, para distinguirlas de las llamadas a funciones. Se le recomienda utilizar siempre llaves, incluso en situaciones en las que son técnicamente opcionales. Estas aumentan la legibilidad y disminuye la probabilidad de errores de la lógica que se presentan cuando se agregan nuevas líneas.

**Funciones y métodos**

Las funciones y los métodos deben nombrarse con lowerCamelCase. Las funciones se deben llamar sin espacios entre el nombre de la función, el paréntesis de apertura, y el primer parámetro, comas y espacios entre cada parámetro, y ningún espacio entre el último parámetro, el paréntesis de cierre, y el punto y coma.

**Variables y arreglos**

Todas las variables deben ser declaradas con “var”. Esto hace al programa más fácil de leer y más fácil de detectar las variables no declaradas que pueden convertirse en implícita globales. Las variables no deben definirse en el ámbito global, se deben definir dentro de la función. Todas las variables deben ser declaradas al principio de una función.

## 3.2 Diagrama de Componentes

El diagrama de componentes muestra la organización y dependencias entre los componentes (parte modular, desplegable y reemplazable de un sistema que encapsula una implementación y expone una serie de interfaces) (44). Para el desarrollo del componente de notificaciones es necesaria la interacción con otros componentes del marco de trabajo como son, ZendExt, Doctrine y ExtJS. El mismo hace uso de servicios internos mediante el IoC para garantizar la comunicación entre otros subsistemas del marco de trabajo. Así como el servidor Openfire que a través del protocolo XMPP garantiza la comunicación en tiempo real del componente.

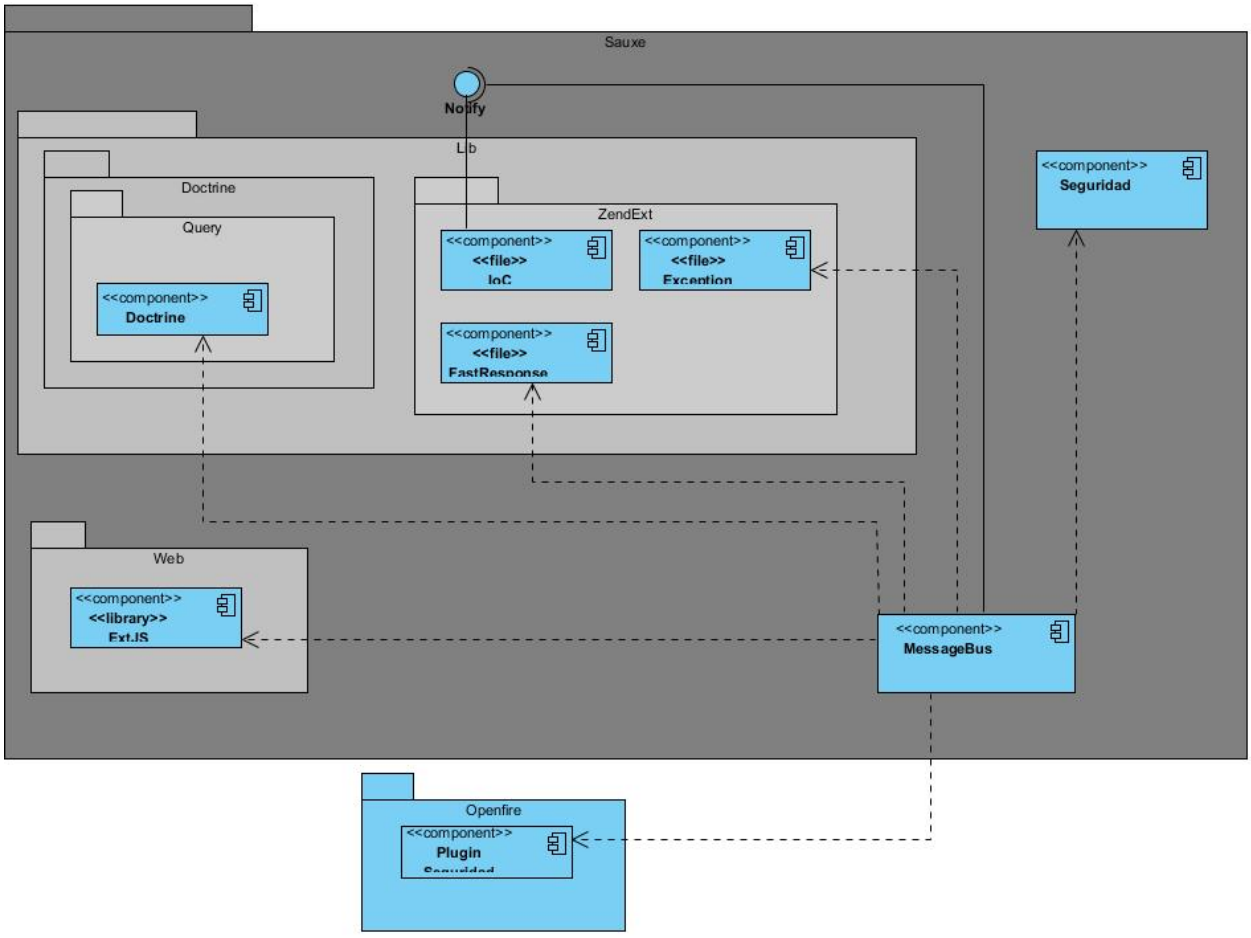



*Figura 12 Diagrama de componentes.*

### 3.3 Diagrama de despliegue

Diagrama que muestra la configuración de nodos de proceso de tiempo de ejecución y los componentes, procesos y objetos que viven en ellos. Los componentes representan manifestaciones de unidades de código de tiempo de ejecución (44). Se necesitan para la solución un servidor con apache donde estarán todas las funcionalidades del componente, al cual se conectará el cliente para realizar consultas a la base de datos que estará implementada sobre un servidor de base de datos PostgreSQL. Por último el servidor de mensajería instantánea Openfire garantizará la comunicación y el envío de las notificaciones desde el servidor apache al cliente.

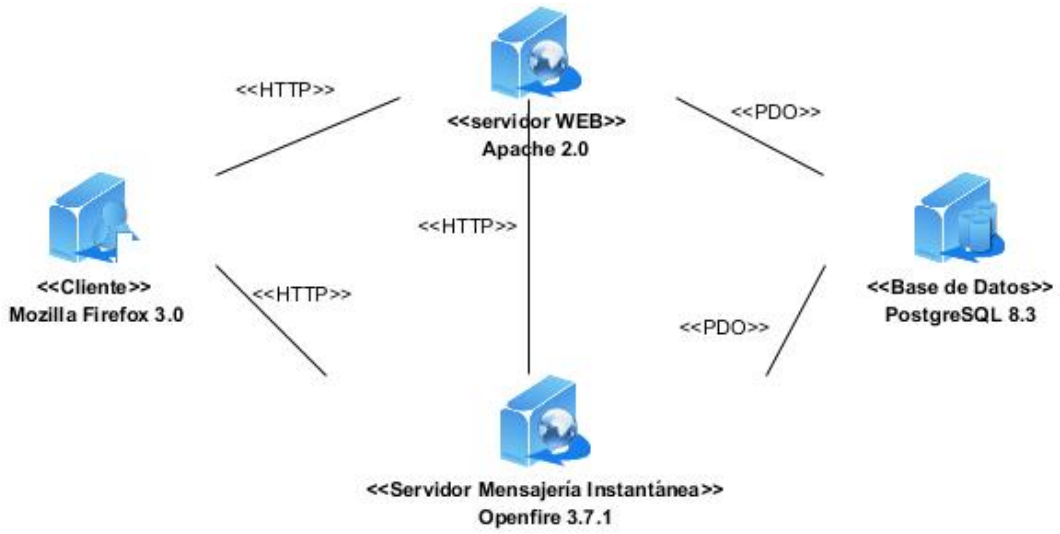



Figura 13. Diagrama de despliegue.

### 3.4 Métricas de software.

Las métricas del producto son una medida cuantitativa que permite a las personas que tienen que ver con el software tener una visión profunda de la eficacia de los procesos del software y de los proyectos que dirigen. Se reúnen los datos básicos de calidad y productividad que son analizados, comparados y evaluados para determinar las mejoras en la calidad y productividad. Las métricas pueden ser usadas para señalar áreas con problemas a través de una evaluación objetiva, de manera que puedan ser solucionados y de esta forma mejorar el proceso de desarrollo del software. Existen cuatro razones para medir los procesos del software, los productos y los recursos, **Caracterizar, Evaluar**, **Predecir** y **Mejorar**.

Se evalúa para comprender mejor los procesos, los productos, los recursos y los entornos de trabajo para establecer las líneas bases para las comparaciones con evaluaciones futuras. Se evalúa para determinar el estado con respecto al diseño, las medidas sirven como sensores para saber cuándo los proyectos y procesos no se encuentran bajo control. Al predecir se puede planificar y aumentar la comprensión de las relaciones entre los procesos y los productos y así establecer objetivos alcanzables para el coste, planificación y calidad de manera que se puedan aplicar los recursos apropiados. Se mide para mejorar al

recoger información cuantitativa que ayude a identificar los problemas e ineficiencias y oportunidades para mejorar la calidad del producto y el rendimiento del proceso (45).

Debido a que este software se realizó bajo la programación orientada a objetos (POO) y las clases constituyen la unidad básica y fundamental en este tipo de programación, resulta viable realizar la validación del mismo con la aplicación de métricas dirigidas a sus clases de forma individual, sus jerarquías y colaboraciones.

De un conjunto de métricas definidas en el documento *Métricas a utilizar en el diseño.doc* perteneciente al departamento Tecnologías de CEIGE, se seleccionaron las métricas, Tamaño operacional de clase (TOC) y Relaciones entre clases(RC), También estas métricas están contenidas en el libro realizado por Lorenz y Kidd (46). Los atributos que evalúan ambas métricas son los siguientes:

- **Responsabilidad:** Se le asigna a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- **Complejidad de implementación:** Grado de dificultad en la implementación de un diseño de clases determinado.
- **Reutilización:** Nivel de reutilización que tiene una clase o estructura de clase, dentro de un diseño de software determinado.
- **Acoplamiento:** Valor de dependencia de una clase o estructura de clase con otras. Este atributo está muy ligado al de Reutilización.
- **Complejidad del mantenimiento:** Categoría de esfuerzo para realizar un arreglo, mejora o 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:** Número de esfuerzos para realizar las pruebas de calidad (Unidad) del producto (componente, clase, conjunto de clases, etc.) diseñado.

### 3.4.1 Tamaño operacional de clase (TOC)

Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

| 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. |

*Tabla 2 Atributos que evalúa por (TOC).*

Para estos atributos de calidad están definidos los siguientes criterios y categorías de evaluación:

| 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 |

*Tabla 3 Rango de valores para los criterios de evaluación de la métrica (TOC).*

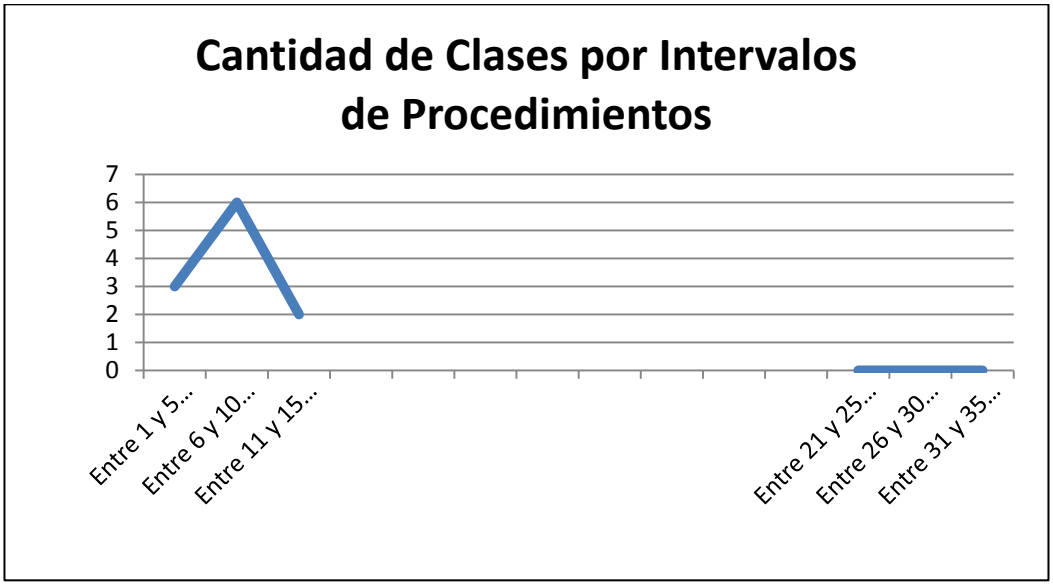
### 3.4.2 Resultados obtenidos al aplicar (TOC)

En la siguiente tabla se recoge como datos las 11 clases a las cuales se le aplicó la métrica TOC en la columna **Clase**, la **cantidad de métodos** que tiene cada una en la columna correspondiente con un total de80 y los atributos de calidad que utiliza esta métrica en las demás columnas y en ellas se representa la categorización en baja, media o alta según los criterios que aparecen en la tabla anterior para la cual se tomó el valor promedio obtenido en la siguiente tabla con un valor de 7.27.

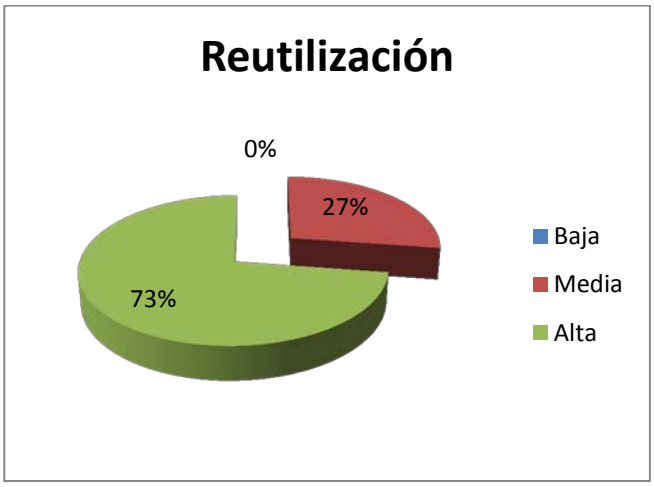
| Clase | Cantidad de Procedimientos | Responsabilidad | Complejidad | Reutilización |
|---|---|---|---|---|
| AplicacionController | 6 | Baja | Baja | Alta |
| GestionarController | 9 | Media | Media | Media |
| GestsubsistemaemitenController | 7 | Baja | Baja | Alta |
| GestsubsistemarecibenController | 6 | Baja | Baja | Alta |
| DatBuzonNotifModel | 4 | Baja | Baja | Alta |
| DatMessagebusModel | 7 | Baja | Baja | Alta |
| GestSubsistemaEmitenModel | 12 | Media | Media | Media |
| GestSubsistemaRecibenModel | 13 | Media | Media | Media |
| SubsistemaeventoModel | 6 | Baja | Baja | Alta |
| DatBuzonNotif | 5 | Baja | Baja | Alta |
| DatMessagebus | 5 | Baja | Baja | Alta |

*Tabla 4 Evaluación de la métrica (TOC).*

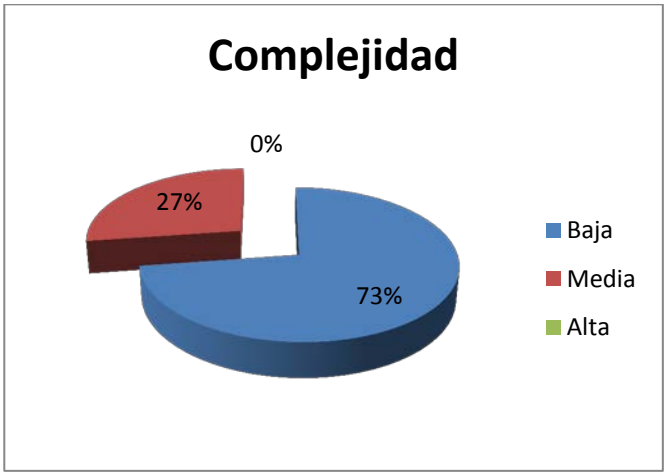
Los elementos: ***gráfica 2***, ***gráfica 3***, ***gráfica 4***, ***gráfica 5*** guardan relación con la ***tabla 3***, y representan gráficamente el resultado de haber aplicado (TOC).



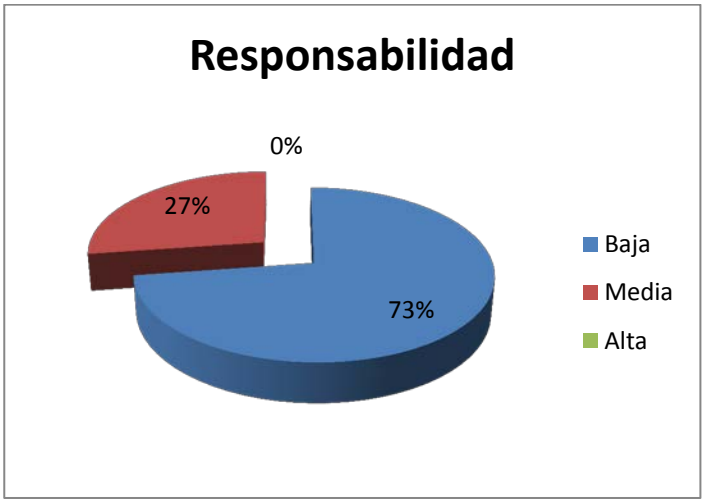

*Gráfica 1 Cantidad de procedimientos por intervalos de procedimientos (TOC).*




*Gráfica 2 Resultados obtenidos de la evaluación de la métrica TOC para el atributo Responsabilidad.*




*Gráfica 3 Resultados obtenidos de la evaluación de la métrica TOC para el atributo Complejidad.*




*Gráfica 4 Resultados obtenidos de la evaluación de la métrica TOC para el atributo Responsabilidad.*

### 3.4.3 Análisis de los resultados obtenidos al evaluar la métrica TOC

Haciendo un análisis de los resultados obtenidos para los atributos de la métrica TOC apreciables en las gráficas de pastel mostradas anteriormente, se puede observar que la mayoría de las clases que conforman el sistema para los atributos responsabilidad y complejidad están dentro de la categoría Media y Baja para un 73% del total, lo que representa que las clases no tienen mucha responsabilidad y no son

tan complejas y en caso de ocurrir algún cambio en el sistema de clases del componente, estaría menos comprometido el funcionamiento correcto del mismo. Mientras que el atributo Reutilización cuenta con igual por ciento pero en las categorías Alta y Media mostrando así que el componente cuenta con una elevada reutilización que permite que las clases puedan ser utilizadas e instanciadas por otras. Por lo que se concluye que los resultados obtenidos según esta métrica son positivos.

### 3.4.4 Relaciones entre clases (RC)

Está dado por el número de relaciones de uso de una clase con otra y evalúa los atributos de calidad que se muestran en la ***Tabla 1***.

| 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. |

*Tabla 5 Atributos de calidad que mide (RC).*

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

| Atributo | Categoría | Criterio |
|---|---|---|
| **Acoplamiento.** | Ninguna | 0 |
| | Baja | 1 |
| | Media | 2 |
| | Alta | >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 |

*Tabla 6 Criterios y categorías para evaluarla métrica (RC).*

### 3.4.5 Resultados obtenidos al aplicar (RC)

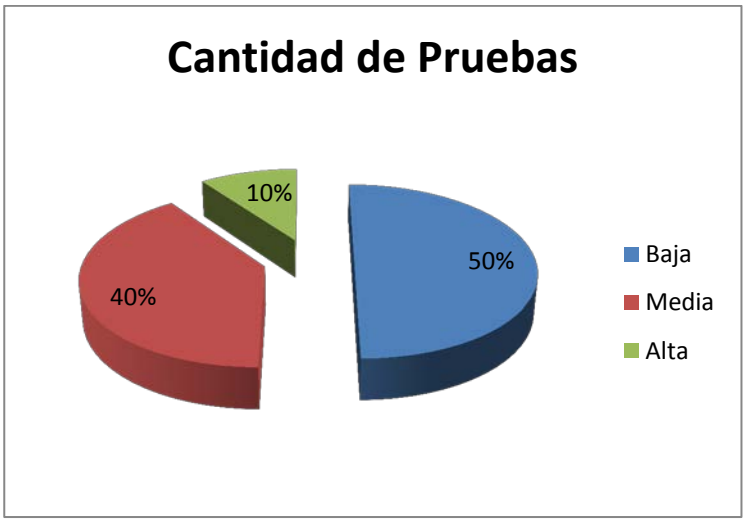
En la siguiente tabla se muestra como datos las 11 clases a las cuales se le aplicó la métrica RC en la columna **Clase**, la cantidad de relaciones que tiene cada una respecto a las demás clases en la columna **Cantidad de Relaciones de Uso**, junto a los atributos de calidad de Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas en sus columnas correspondientes. Al promediar la cantidad de relaciones que tiene cada clase se obtuvo un resultado de relaciones de dependencia por clase de 0.8 el cual se utiliza en la tabla anterior para hallar el valor de la columna criterio.

| Clase | Cantidad de Relaciones de Uso | Acoplamiento | Complejidad Mant. | Reutilización | Cantidad de Pruebas |
|---|---|---|---|---|---|
| AplicacionController | 1 | Baja | Media | Media | Media |
| GestionarController | 2 | Media | Alta | Baja | Alta |
| GestsubsistemaemitenController | 1 | Baja | Media | Media | Media |
| GestsubsistemarecibenController | 1 | Baja | Media | Media | Media |
| DatBuzonNotifModel | 1 | Baja | Media | Media | Media |
| DatMessagebusModel | 2 | Alta | Alta | Baja | Alta |
| GestSubsistemaEmitenModel | 0 | Ninguno | Bajo | Alta | Bajo |

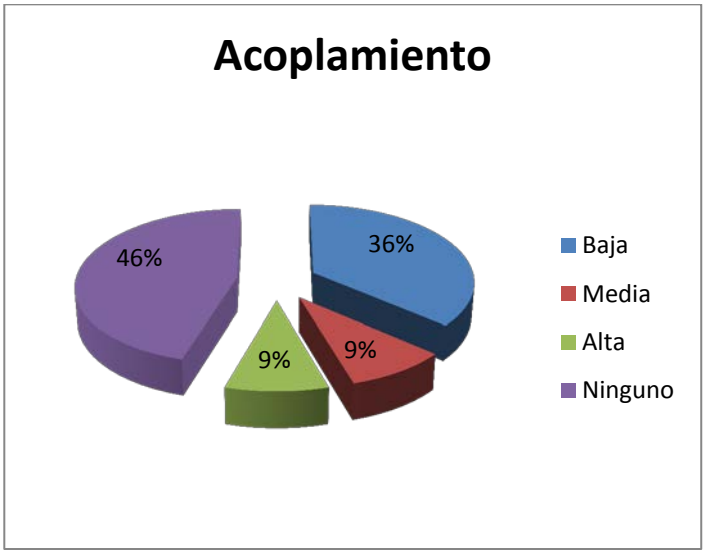
| | | | | | |
|---|---|---|---|---|---|
| GestSubsistemaRecibenModel | 0 | Ninguno | Bajo | Alta | Bajo |
| SubsistemaeventoModel | 0 | Ninguno | Bajo | Alta | Bajo |
| DatBuzonNotif | 0 | Ninguno | Bajo | Alta | Bajo |
| DatMessagebus | 0 | Ninguno | Bajo | Alta | Bajo |

*Tabla 7 Evaluación de la métrica (RC).*

Los elementos: ***Gráfica 6***, ***Gráfica 7***, ***Gráfica 8***, ***Gráfica 9*** guardan relación con la ***Tabla 6***, y representan gráficamente el resultado de haber aplicado (RC).

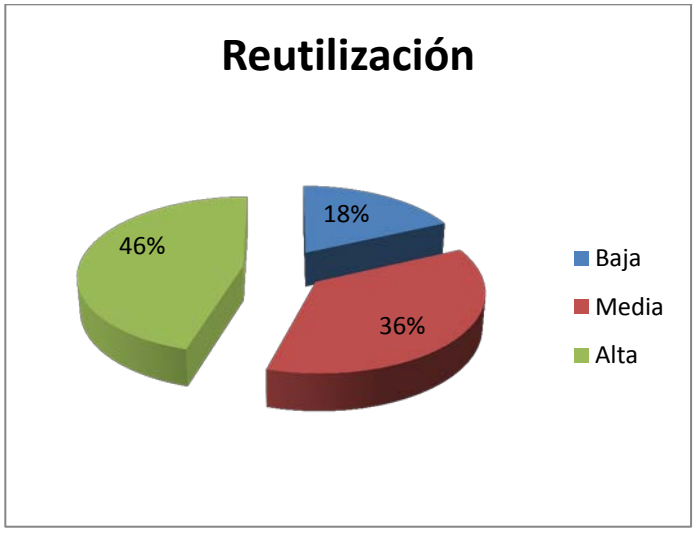



*Gráfica 5 Resultados obtenidos de la evaluación de la métrica RC para el atributo Cantidad de pruebas.*

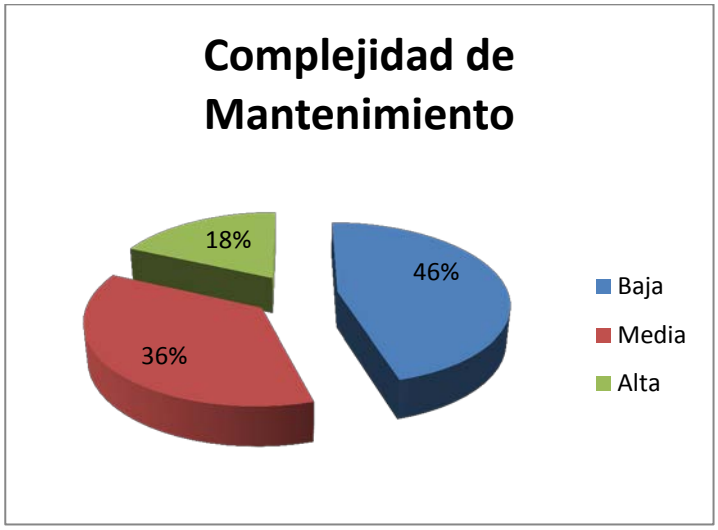



*Gráfica 6 Resultados obtenidos de la evaluación de la métrica RC para el atributo Acoplamiento.*

.




*Gráfica 7 Resultados obtenidos de la evaluación de la métrica RC para el atributo Reutilización.*




*Gráfica 8 Resultados obtenidos de la evaluación de la métrica RC para el atributo Mantenimiento.*

### 3.4.6 Análisis de los resultados obtenidos al evaluar la métrica RC

Los resultados obtenidos durante la evaluación del instrumento de medición de la métrica RC demuestran que las clases del diseño poseen un bajo acoplamiento, ya que para este atributo las categorías Ninguno y Bajo sumaron un 82% del total, mostrando igual por ciento en la categoría alta y media del atributo reutilización. Los atributos complejidad de mantenimiento y cantidad de pruebas, sumaron un 90% y 82 % en las categorías baja y media respectivamente, lo que demuestra que no es necesario un elevado esfuerzo en el momento de realizar cambios, rectificaciones y pruebas al software.

## 3.5 Pruebas de software

Las pruebas de software consisten en la dinámica de la verificación del comportamiento de un programa en un conjunto finito de casos de prueba, debidamente seleccionados de por lo general infinitas ejecuciones de dominio, contra la del comportamiento esperado. Son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador; probando el comportamiento del mismo.

La prueba de software es un elemento crítico para la garantía del correcto funcionamiento del software. Entre sus objetivos están:

1. Detectar defectos en el software.
2. Verificar la integración adecuada de los componentes.
3. Verificar que todos los requisitos se han implementado correctamente.
4. Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente.
5. Diseñar casos de prueba que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo (28).

### 3.5.1 Pruebas de caja blanca

La prueba de caja blanca se basa en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos. Mediante la prueba de la caja blanca el ingeniero del software puede obtener casos de prueba que:

1. Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
2. Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
3. Ejecuten todos los bucles en sus límites operacionales.
4. Ejerciten las estructuras internas de datos para asegurar su validez.

Es por ello que se considera a la prueba de Caja Blanca como uno de los tipos de pruebas más importantes que se le aplican a los software, logrando como resultado que disminuya en un gran porciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad (28).

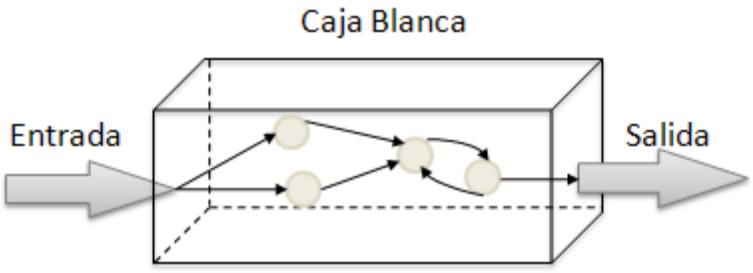



*Figura 14 Interpretación gráfica de caja blanca.*

#### 3.5.1.1 Camino Básico

La prueba del camino básico es una técnica de prueba de la Caja Blanca propuesta por Tom McCabe. Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico (28).

La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática. Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa (28).

A continuación después de haber mencionado los pasos para efectuar la prueba de caja blanca camino básico. Se procede a explicar los mismos, detallando algunas de las técnicas que utilizan y los conceptos fundamentales. El objetivo es comprender como se realiza esta técnica.

**Notación de Grafo de Flujo**

Para aplicar la técnica del camino básico se debe introducir una sencilla notación para la representación del flujo de control, el cual puede representarse por un Grafo de Flujo. Cada nodo del grafo corresponde a una o más sentencias de código fuente. Todo segmento de código de cualquier programa se puede traducir a un Grafo de Flujo. Para construir el grafo se debe tener en cuenta la notación para las instrucciones. Un Grafo de Flujo está formado por 3 componentes fundamentales que ayudan a su elaboración, comprensión y nos brinda información para confirmar que el trabajo se está haciendo adecuadamente (28).

**Nodo**

Cada círculo representado se denomina nodo del Grafo de Flujo, el cual representa una o más secuencias procedimentales. Un solo nodo puede corresponder a una secuencia de procesos o a una sentencia de decisión. Puede ser también que hayan nodos que no se asocien, se utilizan principalmente al inicio y final del grafo.

**Aristas**

Las flechas del grafo se denominan aristas y representan el flujo de control, son análogas a las representadas en un diagrama de flujo. Una arista debe terminar en un nodo, incluso aunque el nodo no represente ninguna sentencia procedimental.

**Regiones**

Las regiones son las áreas delimitadas por las aristas y nodos. También se incluye el área exterior del grafo, contando como una región más. Las regiones se enumeran. La cantidad de regiones es equivalente a la cantidad de caminos independientes del conjunto básico de un programa. Cualquier representación del diseño procedimental se puede traducir a un grafo de flujo. Cuando en un diseño se encuentran condiciones compuestas (uno o más operadores AND, NAND, NOR lógicos en una sentencia condicional), la generación del grafo de flujo se hace un poco más complicada

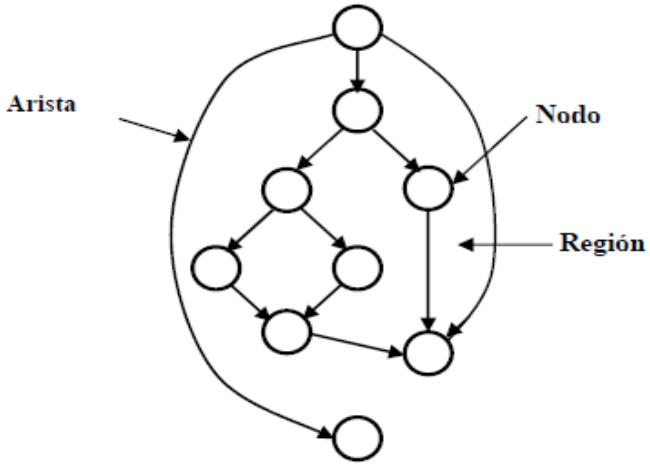



*Figura 15 Características de los grafos.*

**Complejidad Ciclomática**

La complejidad ciclomática es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Un camino independiente es cualquier camino del programa que introduce por lo menos un nuevo conjunto de sentencias de procesamiento o una nueva condición. El camino independiente se debe mover por lo menos por una arista que no haya sido recorrida anteriormente.

**Derivación de casos de prueba**

Luego de tener elaborados los Grafos de Flujos y los caminos a recorrer, se preparan los casos de prueba que forzarán la ejecución de cada uno de esos caminos. Se escogen los datos de forma que las condiciones de los nodos predicados estén adecuadamente establecidas, con el fin de comprobar cada camino.

Ejemplos de casos de prueba para cada camino:

Camino 1: 1-2-3-5-6. Escoger algún X y Y tales que cumpla X >= 0 AND Y >= 0. X = 10 AND Y = 20.
Camino 2: 1-2-4-6. Escoger algún X tal que se cumpla X < 0. X = -15.

Luego de confeccionar los casos de prueba se ejecutan cada uno de estos y se comparan los resultados con los esperados. Una vez terminados todos los casos de prueba, se estará seguro de que todas las sentencias del programa se han ejecutado por lo menos una vez. Es importante considerar que algunos caminos no se pueden probar de forma aislada. O sea, la combinación de datos requeridos para recorrer el camino no se puede obtener con el flujo normal del programa. En tales casos, estos caminos se prueban como parte de otra prueba de camino (28).

Luego de explicar en qué consisten los pasos para realizarla prueba de caja blanca Camino Básico y haber mencionado los conceptos fundamentales que contiene este tipo de prueba, se selecciona el método: **cargarCodigonotificaciones** de la clase **SubsistemaeventoModel** como ejemplo. Este método tiene la responsabilidad de cargar las notificaciones que puede enviar un subsistema, para lo cual obtiene

los datos directamente del fichero “subemitennotif.xml”. Recibe1sola variable como parámetro, el nombre del subsistema el cual contiene las notificaciones que es capaz de emitir. A continuación se muestra el código fuente de dicho método.

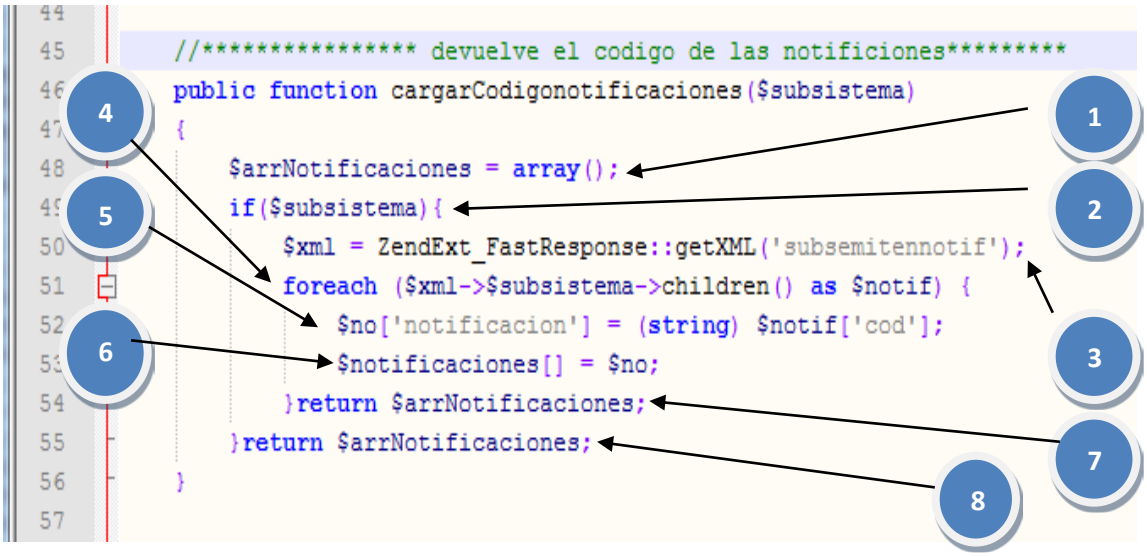



*Figura 16 Código fuente de la función escogida para aplicar la prueba de caja blanca.*

**Paso1** A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.

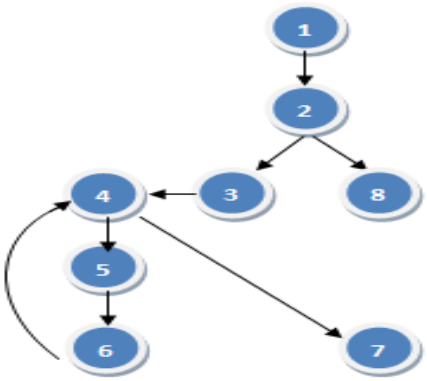



*Figura 17 Grafo de flujo asociado.*

**Paso 2** Se calcula la complejidad ciclomática.

Existen 3 vías para calcular la complejidad ciclomática, *V (G)*, de un grafo de flujo asociado:

1. La cantidad de nodos predicado que existan en el grafo.
2. V (G) = A - N + 2.

Donde: *A* es el número de aristas del grafo y *N* es el número de nodos.

3. La complejidad ciclomática, *V(G)*, también se define como:

V (G) = P + 1

Donde: *P* es el número de nodos predicado contenidos en el grafo *G*.

Se procede a calcular la complejidad ciclomática del grafo mediante la segunda vía V (G) = A - N + 2. Tras evaluar en la función queda de la siguiente forma:

V (G) =8 – 8 + 2

Lo cual da como resultado 2, por tanto la complejidad ciclomática del grafo de la ***Figura 27*** es 2, que coincide con la cantidad de caminos independientes y por tanto es 2, el límite superior para la cantidad de casos de prueba que se pueden realizar para forzar la ejecución de la función, por cada uno de los caminos independientes que contiene el grafo.

**Paso 3** Determinar un conjunto básico de caminos independientes.

Un camino independiente se debe mover por lo menos por una arista que no haya sido recorrida anteriormente. Por lo tanto los caminos independientes para este caso son:

Camino1: 1-2-8

Camino2: 1-2-3-4-5-6-4-7

**Paso 4** Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Camino1: 1-2-8

Después de haber extraído los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para este procedimiento, se debe realizar al menos un caso de prueba por cada camino básico. Para realizarlos es necesario cumplir con las siguientes exigencias:

- **Descripción:** Se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.
- **Condición de ejecución:** Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.
- **Entrada:** Se muestran los parámetros que entran al procedimiento.

Resultados Esperados: Se expone resultado que se espera que devuelva el procedimiento.

| Número de Camino | Descripción | Entrada | Resultados Esperados |
|---|---|---|---|
| 1 | El parámetro subsistema está vacío. | Vacío | Muestra un mensaje informando al usuario “Error del sistema. Contacte con el administrador. ” |
| 2 | El parámetro subsistema no está vacío | seguridad | Se muestra las notificaciones que pertenecen al sistema seguridad. |

### 3.5.2 Resultado de las pruebas de Caja blanca

Para la realización de las pruebas de caja blanca al sistema fueron analizados en total 35 funcionalidades, cada clase tiene de 6 a 9 funcionalidades lo que conllevó a obtener resultados positivos en el 80% las funciones, en el otro 20% los resultados no fueron los deseados, los servicio no cumplían con su funcionalidad y no devolvían los datos necesarios usados en otras funciones, en ocasiones los datos no eran insertados en la base de datos o no cumplían con el objetivo esperado, De esta forma se demuestra que al emplear este tipo de pruebas se tiende a que disminuya en un gran porciento el número de errores

existentes en los sistemas y por ende una mayor calidad y confiabilidad del código fuente y por lo tanto de la aplicación.

### 3.5.3 Pruebas de caja negra

Cuando se utiliza el término pruebas de caja negra se está haciendo referencia a las pruebas que se llevan a cabo sobre la interfaz del software sin tener en cuenta el código, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene.

El diseño de estas pruebas tiene el propósito de detectar (47):

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.


*Figura 18 Interpretación gráfica Caja negra.*

Algunas técnicas utilizadas en la prueba de caja negra son:

**Partición de equivalencia:** Técnica que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

**Análisis de valores límite:** La experiencia muestra que los casos de prueba que exploran las condiciones límite producen mejor resultado que aquellos que no lo hacen. Las condiciones límite son aquellas que se hallan en los márgenes de la clase de equivalencia, tanto de entrada como de salida. Por ello, se ha desarrollado el análisis de valores límite como técnica de prueba. Esta técnica lleva a elegir los casos de prueba que ejerciten los valores límite.

**Grafos de causa-efecto:** Es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones (47).

De estas técnicas, la seleccionada fue **partición de equivalencia** la cual permite examinar los valores válidos e inválidos de las entradas existentes en el software. Para la aplicación de esta técnica se realizan los diseños de casos de prueba los cuales se basan en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica (47).
Las clases de equivalencia se pueden definir de acuerdo con las siguientes directrices:

- Si un parámetro de entrada debe estar comprendido en un cierto rango, aparecen 3 clases de equivalencia: por debajo, por encima y en el rango.
- Si una entrada requiere un valor concreto, aparecen 3 clases de equivalencia: por debajo, por encima y en el rango.
- Si una entrada requiere un valor de entre los de un conjunto, aparecen 2 clases de equivalencia: en el conjunto o fuera de él.
- Si una entrada es booleana, hay 2 clases: sí o no.
- Los mismos criterios se aplican a las salidas esperadas: hay que intentar generar resultados en todas y cada una de las clases.

Se realizaron 10 diseños de casos de prueba para validar el componente atendiendo a la técnica partición de equivalencia.

Las pruebas de caja negra a la solución fueron desarrolladas por el Grupo de Aseguramiento de la Calidad del Centro de Informatización para la Gestión de Entidades (CEIGE). Estas se realizaron en dos

iteraciones, donde finalmente se comprobó en la segunda iteración que el componente estaba libre de no conformidades culminando así la fase de pruebas internas.

La tabla que se muestra a continuación ofrece detalles de los resultados obtenidos hasta el momento, desglosando las no conformidades (NC) detectadas en significativas (S) que incluyen errores de validación, de función y excepciones, las que no proceden (NP) y las no significativas NS que incluyen los errores en los casos de prueba.

| **Agrupación de Requisitos** | **No conformidades** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Iteración 1 | | | | Iteración 2 | | | |
| | NC | NP | S | NS | NC | NP | S | NS |
| **Gestionar las notificaciones que envía cada sistema** | 3 | | 2 | 1 | 0 | 0 | 0 | 0 |
| **Gestionar las notificaciones que recibe cada sistema** | 2 | | | 2 | 0 | 0 | 0 | 0 |
| **Gestionar notificaciones por Rol** | 1 | | | 1 | 0 | 0 | 0 | 0 |
| **Eliminar mensaje del buzón de notificaciones** | 0 | | | | 0 | 0 | 0 | 0 |

En las mismas se encontraron mayormente errores de faltas de ortografías en las interfaces, de validaciones, los mensajes de información estaba mal elaborados, faltaban los iconos de los botones, etc.

Luego de ser corregidos los errores encontrados en las pruebas, se pudo comprobar que el flujo de trabajo de las interfaces estaban correcto ya que cumplían con las condiciones necesarias que se habían

planteado para las funcionalidades descritas.

## Conclusiones parciales

El componente de notificaciones cumple con las normas de codificación propuestas, lo cual contribuye a entender y organizar el código fuente del mismo, Otros artefactos generados por el modelo de desarrollo de software utilizado, como el diagrama de componentes brindan una visión global del funcionamiento de los sistemas, librerías y ficheros del sistema que intervienen, junto con el modelado de los sistemas de hardware observables en el diagrama de despliegue.

De acuerdo con los resultados de las métricas utilizadas que evalúan el diseño de las clases que intervienen en la solución a través de sus atributos de calidad se puede tener una noción de cuan eficiente es el mismo y establece la medida para poder ser comparado con los estándares que establece CEIGE en cuanto a su funcionamiento. Además las pruebas de software realizadas contribuyeron a eliminar la mayor cantidad de errores que pudiera tener el componente y constataron que el mismo cumple con las expectativas de los clientes.

# Conclusiones generales

Con la realización y culminación del componente para la gestión de notificaciones del marco de trabajo Sauxe se puede afirmar lo siguiente:

- Después recopilar información y hacer un estudio detallado sobre las técnicas de notificación en tiempo real utilizadas en aplicaciones web, se obtuvo como resultado que el protocolo XMPP a través del servidor Openfire cumplía de mejor manera con el objetivo general de esta investigación.
- Se determinaron y analizaron las herramientas necesarias para el desarrollo de la solución.
- Luego de la implementación se obtuvo un producto funcional acorde a los requisitos identificados.
- El componente fue probado por la subdirección de Calidad de CEIGE y por el grupo de trabajo del Dpto. de Tecnología, los cuales constataron que el componente cumple el objetivo general determinado al comienzo de la investigación.

- El componente para la gestión de notificaciones para el marco de trabajo Sauxe, permite que los usuarios del mismo conocer una determinada información en el instante que se genera o cambia del lado del servidor, sin la necesidad de que el usuario actualice manualmente la fuente de información, impregnando de esta forma instantaneidad al proceso y contribuyendo así a economizar el tiempo y beneficiar el proceso de toma de decisiones.

# Recomendaciones.

- Implementar una funcionalidad que permita configurar el fichero XML que contiene la configuración del servidor jabber desde una interfaz visual.

- Agregar soporte a través del cual se pueda enviar notificaciones mediantes otros canales de comunicación como, correo electrónico, servicio de jabber de la universidad y mensajes de texto aprovechando la telefonía celular.

# Bibliografía

1. **Herrera, Diaz Francisco Paul.** s.l. : http://paulhdcpp.wordpress.com/2010/04/12/deberes/., 2011.

2. **Baryolo, Ing. Oiner Gómez.** *TRABAJO ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE SOFTWARE.*

3. **Wellings, Burns.** *Real-time systems and programming languages.3ª ed.* s.l. : London Addison Wesley., 1997.

4. The free dictionary - tiempo real. [En línea] [Citado el: 25 de febrero de 2012.] http://es.thefreedictionary.com/tiempo.

5. Master magazine. [En línea] [Citado el: 21 de febrero de 2012.] http://www.mastermagazine.info/termino/6891.php.

6. **Werdmuller, Ben.** *Build a web-based notification tool with XMPP.* 2010.

7. **Barros, Laura.** *Programacion Concurrente y Distribuida.*

8. **Oviedo Chaparro, Luis Enrique.** *COMET: UN SIGUIENTE PASO AL AJAX MOVIENDO DE LAS APLICACIONES WEB TRADICIONALES A UN NUEVO ESTILO.* s.l. : Facultad de Ciencias y Tecnología, Universidad Católica de Asunción.

9. Mensajeria Instantanea Libre jabber.org. [En línea] marzo de 2009. [Citado el: 3 de noviembre de 2011.] http://www.jabberes.org/glosario.

10. sagt.cnti.gob.ve. [En línea] [Citado el: 27 de marzo de 2012.] http://sagt.cnti.gob.ve/otrs/public.pl?Action=PublicFAQZoom;ItemID=429.

11. www.fing.edu.uy. [En línea] [Citado el: 3 de noviembre de 2011.] http://www.fing.edu.uy/~asabigue/prgrado/2004eofgl/contenido/archivos/Anexo-III.pdf .

12. desarrolloweb. [En línea] [Citado el: 2012 de Febrero de 12.] http://www.desarrolloweb.com/articulos/449.php.

13. definicion.de. [En línea] [Citado el: 2012 de Febrero de 12.] http://definicion.de/xml/.

14. Ecured. [En línea] [Citado el: 12 de 12 de 2011.] http://www.ecured.cu/index.php/Modelo_de_dominio.

15. **Sommerville, I.** *Ingeniería de Software, Pearson Educación.* 2002.

16. **M, S Perez.** *Propuesta de modelo de desarrollo de software tecnológico del Centro de Soluciones de Gestión.* Cuidad de la Habana : s.n., 2009.

17. **Escribano, Gerardo Fernández.** *Introducción a Extreme Programing.* 2002.

18. *Manual de PHP.* Cuidad Habana : s.n., 2011.

19. **H, Ty.** Manuales. [En línea] [Citado el: 5 de noviembre de 2011.] http://max-alva.webs.com/javascript.htm.

20. Visual Paradigm. Visual Paradigm for UML - UML tool for software application development. . [En línea] [Citado el: 23 de enero de 2012.] http://www.visual-paradigm.com/product/vpuml/.

21. **O Gomes Baryolo, Mariela Tenrero Cabrera, Nemuris Silega Martinez.** *Plantilla Registro de propiedad Intelectual (Sauxe).* La habana : s.n., 2008.

22. **S. FRederick, Ramsay , Colin y Blades, Steves' Cutter.** *Learning EXT JS.*

23. *Secure Programming with Zend Framework .* **Esser, S.** Amsterdan : s.n., 2009.

24. *, Implementación de módulo de Contabilidad General del sistema integral de gestión Cedrux. .* **Aquino, Yasser A.a.L.** Cuidad de la Habana : s.n., 2009.

25. Strophe - librería para XMPP. [En línea] [Citado el: 24 de febrero de 2012.] http://strophe.im/.

26. **Ing. Adonis Ricardo Rosales García, Ing. Yasirys Terry González.** *COMPONENTE DE COMUNICACIÓN EN TIEMPO REAL DE VALORES DE LOS KPIS DE UN DASHBOARD.* Cuidad de la habana : s.n., 2010.

27. xmpphp: The xmpphp library. [En línea] [Citado el: 24 de febrero de 2012.] http://code.google.com/p/xmpphp/.

28. Ecured. [En línea] [Citado el: 25 de Marzo de 2012.] http://www.ecured.cu/index.php/Pruebas_de_software.

29. Mozilla Firefox. [En línea] [Citado el: 23 de noviembre de 2011.] http://www.mozilla-europe.org/es/firefox/3.0/releasenotes/..

30. **Equipo de desarrollo de PostgreSQL.** undersecurity. [En línea] 1996. http://lib.undersecurity.net.

31. Visual-Paradigm. Visual Paradigm for UML 8.3 Model-Code-Deploy Platform. [En línea] [Citado el: 12 de Diciembre de 2011.] http://www.visual-paradigm.com/product/vpuml/..

32. NetBeans.org. [En línea] [Citado el: 12 de diciembre de 2011.] http://NetBeans.org.

33. **Sawyer, I.S y P.** *Requirements Engineering: A good practice guide.*

34. **Ivar Jacobson, Grady Booch, James Rumbaugh.** *El Proceso Unificado de Desarrollo de software.* 2000.

35. **María José Escalona, Nora Koch.** *Ingeniería de Requisitos en Aplicaciones para la Web – Un estudio comparativo.* Sevilla : s.n., 2002.

36. *Ingeniería de Requisitos en Aplicaciones para la Web. Un estudio comparativo.* Sevilla : s.n., 2002.

37. **González, Oscar Oramas.** *Interoperabilidad del proceso inventario en el sistema Cedrux.* 2011.

38. **Martínez, Alejandro Martínez y Raúl.** ebookbrowse.com. [En línea] [Citado el: 23 de enero de 2012.] http://ebookbrowse.com/tema-1-normalizaci%C3%B3n-bibliograf%C3%ADa-libro-de-bd-de-rosa-maria-pdf-d91217792..

39. **Frank Buschmann, R.M., hans Rohnert, Peter Sommerlad, Michael Stal.** *Pattern-Oriented Software Architecture: A system of patterns.* s.l. : JOHN WlLEY & SONS, 2001. vol1.

40. **Tedeschi, Nicolás.** Microsoft MSDN. *¿Qué es un Patrón de Diseño?* [En línea] [Citado el: 13 de marzo de 2012.] http://msdn.microsoft.com/es-es/library/bb972240.aspx..

41. **Alexander, Christopher.** *The Timeless Way of Building. .* 1979.

42. **Larman, Craig.** *Applying UML & Patterns: Introduction to Object-Oriented Analysis & Design, & Iterative Development.* 1999.

43. **Microsoft.** Revisiones de código y estándares de codificación. [En línea] [Citado el: 15 de Mayo de 2012.] http://msdn.microsoft.com/es-es/library/aa291591%28v=vs.71%29.aspx.

44. **Corporation, Rational Software.** *Glosario de Rational Unified Process.* 2003.

45. Conceptos de procesos. [En línea] [Citado el: 15 de Mayo de 2012.] http://es.scribd.com/doc/49282475/Conceptos-Procesos.

46. **Gonzalez, DH.** *Métricas en el desarrollo del Software.*

47. **Pressman, Roger.** *Ingeniería de lSoftware. Un enfoque prático. Interamericana de España.* 2002.