Universidad de las Ciencias Informáticas

Facultad 3

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

Autor:
Jorge Méndez Barreto.

Tutor(es):

Lic. Orlando Arnaldo Valenzuela Aguilera.
Ing. Yuniel Cedeño Mendoza.

Junio, 2012
Habana, Cuba.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al centro CEIGE de 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 _____.

_____ _____

Jorge Méndez Barreto — Ing. Yuniel Cedeño Mendoza

Autor — **Tutor**

_____

Lic. Orlando Arnaldo Valenzuela Aguilera

Tutor

DATOS DE CONTACTO

Síntesis del Tutor: Orlando Valenzuela. Lic. en Ciencias de la Computación, graduado en el año 2007.

Síntesis del Tutor: Yuniel Cedeño Mendoza. Ingeniero en Ciencias Informáticas, graduado en el año 2010.

Sintesis del Autor: Jorge Méndez Barreto. Estudiante de la Universidad de Ciencias Informáticas.

AGRADECIMIENTOS

Quiero agradecer en primer lugar a mis padres por ser padres ejemplares en todos los sentidos, los quiero muchísimo. Gracias por criarme, por ser mis guías, por todos estos años de sacrificio, por brindarme amor y comprensión.

Quiero también agradecer a mi hermano por siempre estar a mi lado apoyándome y guiándome.

Gracias a todos mis familiares más cercanos que siempre están preocupados por mí y mis estudios.

No pueden faltar en estos agradecimientos, mis compañeros que me han apoyado durante estos 5 años de estudio.

DEDICATORIA.

Dedico este trabajo a mis padres y hermano por ser las personas más importantes de mi vida.

RESUMEN.

Con el desarrollo del sistema CEDRUX llevado a cabo por el Centro de Informatización de la Gestión de Entidades, de la Universidad de las Ciencias Informáticas, se consideró que debe poseer como una de sus premisas fundamentales, un componente de gestión de notificaciones, por el cual los usuarios del mismo puedan ser avisados de eventos que ocurran en el marco de trabajo.

Este componente es desarrollado sobre la base de un servidor de Mensajería Instantánea el cual se nombra Openfire, pero este en gran medida se ve imposibilitado de realizar operaciones en conjunto con el Sistema Integral de Seguridad ACAXIA, por lo que se ha planteado el desarrollo de un plugin que permita la interacción entre estos dos sistemas.

Este trabajo tiene como objetivo el desarrollo de un plugin que permita el intercambio de datos de una forma transparente y segura entre el servidor Openfire y ACAXIA. Además de detallar las tecnologías a emplear y las funcionalidades implementadas, con el objetivo de solucionar el problema planteado.

Palabras Claves: tecnologías, plugin.

Contenido.

Índice de figuras:

Índice de Tablas:

Índice de Gráficas:

INTRODUCCIÓN.

La informática sin duda es uno de los logros más grandes del hombre y desde el surgimiento de la misma la sociedad ha ido cambiando gracias a los grandes avances tecnológicos. El auge de las TIC (Tecnologías de la Informática y las Comunicaciones) ha traído consigo que existan empresas y compañías dedicadas enteramente a la producción de software, las cuales han utilizado estos conocimientos para el perfeccionamiento de la producción y su desarrollo, convirtiéndose la producción de software en uno de los principales renglones de la economía de muchos países.

Cuba, aprovechando el gran potencial profesional con el que cuenta, ha creado organismos y empresas para solventar la necesidad creciente que existe de desarrollar productos informáticos, ya sea para la comercialización y la demanda nacional o internacional. Entre las principales entidades productoras de software se encuentra la Universidad de Ciencias Informáticas (UCI).

Desde su concepción, la UCI, se ha convertido en una universidad que prepara y gradúa ingenieros informáticos de alto nivel, y a la vez funciona como una empresa productora de software; que poco a poco va alcanzando prestigio a nivel internacional. En esta existen varios centros productivos en los cuales se desarrollan un gran número de proyectos de software, entre los que se encuentra el Centro Informatización de la Gestión de Entidades (CEIGE).

En el centro CEIGE se crea un marco de trabajo que ha sido nombrado “SAUXE”, basado en el marco de trabajo ZendExt y al cual se le agregan algunas particularidades para enriquecerlo. El mismo es capaz de manejar un gran número de subsistemas, módulos y componentes, además de permitir el desarrollo de una arquitectura web.

Actualmente se pretende desarrollar un componente de gestión de notificaciones para el marco de trabajo SAUXE pero este se ve afectado en gran medida por la arquitectura cliente-servidor empleada por las aplicaciones web. Esto trae consigo que no se pueda notificar a los usuarios de algún evento existente en el servidor en tiempo real, afectando así el tiempo de respuesta a un evento determinado por parte del usuario. Para dar solución a este problema es usado un Servidor de Mensajería Instantánea el cual da solución al inconveniente antes mencionado. Este servidor jabber almacena un gran número de usuarios y necesita mantener la seguridad entre los peldaños más altos de nuestro desarrollo. Por lo antes expuesto es necesario

relacionarlo con el Sistema Integral de Gestión de Seguridad (ACAXIA), que implementa un componente de autenticación regido por el estándar internacional SAML [1], brindando la posibilidad de utilizar firmas, certificados digitales y la implementación de una arquitectura de Single Sign-On.

Por lo antes expuesto surge como **problema a resolver**: ¿Cómo garantizar la autenticidad de los usuarios de Openfire empleando el Sistema Integral de Seguridad ACAXIA? Tomando como **objeto de estudio** Los servidores de comunicación en tiempo real**.** Según lo planteado anteriormente se establece como **Campo de Acción** la modelación e implementación de aplicaciones que garanticen la seguridad de componentes de gestión de notificaciones.

Quedando definido como **objetivo general** Desarrollar un plugin para la autenticación de usuarios. Para el desarrollo de esta investigación se desglosa el objetivo general en los siguientes **Objetivos Específicos**:

1. Realizar el Marco Teórico de la investigación.
2. Análisis y diseño de un plugin para la autenticación de usuarios.
3. Implementar un plugin para la autenticación de usuarios empleando el Sistema Integral de Seguridad ACAXIA.
4. Validar la solución propuesta.

Se definen además, las siguientes **tareas de investigación** a realizar para dar cumplimiento a los objetivos específicos:

- Valoración de la arquitectura de plugin de Openfire.
- Exploración de la expedición de certificados de ACAXIA.
- Obtener los requisitos necesarios para autenticar un usuario de ACAXIA en Openfire.
- Análisis y diseño de un plugin a Openfire que permita la gestión de los usuarios del marco de trabajo desde ACAXIA y su posterior autenticación.
- Desarrollo de un plugin a Openfire que permita la gestión de los usuarios del marco de trabajo desde ACAXIA y su posterior autenticación.
- Validar las funcionalidades de la aplicación a través de los casos de pruebas descritos para la aplicación.

[1] Security Assertion Markup Language. Entorno basado en XML diseñado para facilitar el intercambio de información de autenticación y autorización, entre los diferentes componentes de la infraestructura de seguridad informática.

La **idea a defender** que se define en esta investigación es**:** con el desarrollo de un plugin para Openfire se permitirá brindar al mismo la seguridad que provee ACAXIA para la autenticación de usuarios.

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 casos y técnicas de programación más utilizadas en el desarrollo de este tipo de componentes. Además conllevó 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 método **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 la 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 nos 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 las áreas de CEIGE.

Capítulo 1: Fundamentación Teórica.

Introducción:

En el presente capítulo se desarrolla una investigación para llevar a cabo el desarrollo de un plugin y su estado del arte. Se realiza además un estudio sobre las herramientas, lenguajes y metodologías a utilizar con el fin de darle solución al problema planteado, teniendo en cuenta las particularidades y requerimientos del mismo. Además se abordan las tendencias actuales que dan solución al problema que se enfrenta.

1.1.Técnicas de notificación en tiempo real.

Las alertas a tiempo reales son una potente función que se añade a más aplicaciones cada día; la incorporación de notificaciones a tiempo real puede suponer una gran diferencia en la experiencia del usuario y el rendimiento en el trabajo al utilizar estas aplicaciones. 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. (1)

Para un mejor entendimiento en el tema se explican a continuación los principales elementos que componen estas técnicas:

1.1.1. Web Sockets.

WebSocket es una tecnología que proporciona un canal de comunicación bidireccional y full-duplex sobre un único socket TCP. Está diseñada para ser implementada en navegadores y servidores web, pero puede utilizarse por cualquier aplicación cliente/servidor. Web sockets sucede a través de la capa 3 del modelo de referencia de Interconexión de Sistemas Abiertos (OSI[2], Open System Interconnection). Tanto el servidor y el cliente/s utilizan una aplicación, la cual es responsable de interactuar con los sockets, dándole responsabilidad el sistema operativo de utilizar un proceso para: crear, utilizar y luego cerrar el socket cuando ya se ha transmitido el mensaje. Este

[2] Es un marco de referencia para la definición de arquitecturas de interconexión de sistemas de comunicaciones.

flujo de eventos constituye un uso de recursos innecesario de ambas partes, además de representar una relativa demora, la cual se puede incrementar considerablemente si se usa WebSocket Secure connections (wss[3]), porque el mismo utiliza Transport Layer Security(TLS[4]) y representa un costo adicional en cuanto al tiempo que puede tardar para encriptar. (2)

Aunque el protocolo WebSocket es indiferente a la conexión sobre servidores proxy o cortafuegos, implementa una negociación compatible con HTTP para que los servidores HTTP puedan compartir sus puertos HTTP y HTTPS 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. (2)

Esta técnica no es utilizada para la solución, debido a que es una tecnología reciente, que todavía presenta problemas con su desarrollo en PHP no siendo así en JAVA, además de haber muchos navegadores web que aún no la soportan por problemas de seguridad.

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 Cliente-Servidor. (3)

COMET se basa en que el servidor envía datos aunque el cliente no los pida (HTTP Push), si haces una llamada, el canal se queda abierto y el servidor va mandando 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[5] de larga duración para reducir la latencia con la cual los mensajes son pasados

[3] Encrypted WebSocket connections. Indica que el tráfico a través de conexión es protegido mediante TLS (incluidas las prestaciones estándar de TLS como la confidencialidad de datos y la integridad y la autenticación de punto final).
[4] Seguridad de la Capa de Transporte. Protocolo criptográficos que proporcionan comunicaciones seguras por una red, comúnmente Internet.
[5] Protocolo de Transferencia de Hipertexto. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

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. (3)

1.1.2.1. Ventajas.

Las aplicaciones que implementan el estilo COMET pueden comunicar cambios de estados pagando una pequeña latencia. Esto hace a esta tecnología adecuada para muchas clases de aplicaciones colaborativas multi – usuarios y de monitoreo que de otra forma sería muy difícil o imposible de manejar en un navegador sin la utilización de un plugin. (3)

1.1.2.2. Desventajas.

- Este método podría significar un desperdicio en términos de sockets [6] y threads [7], y también representaría una sobrecarga para los firewalls, los balanceadores de carga.
- Esta técnica al mantener abierta la conexión por un largo periodo, presenta problemas de escalabilidad.
- Las aplicaciones basadas en HTTP realizan pequeñas peticiones de páginas. Esto significa que se pueden manejar pedidos a una mayor cantidad de usuarios en comparación con las técnicas de conexión de larga duración. (3)

COMET no es usado para dar respuesta al problema planteado principalmente por los problemas de escalabilidad que puede representar para la solución.

1.1.3. XMPP.

XMPP es un protocolo totalmente abierto que permite a cualquier desarrollador implementar Jabber/XMPP sin ningún coste. Las primeras tecnologías fueron

[6] Designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada.
[7] Hilo de ejecución, hebra o subproceso de la unidad de procesamiento más pequeña que puede ser planificada por un sistema operativo.

desarrolladas en 1998 y ahora son muy estables. Existen miles de servidores utilizando este protocolo en Internet, y millones de personas utilizándolo para mensajería instantánea para servicios públicos como Google Talk e implementaciones en organizaciones. Es por ello que cuenta con múltiples implementaciones entre clientes, servidores, componentes de servidores y librerías de código. Su arquitectura es similar al correo electrónico, por lo que se puede usar un servidor propio, permitiendo a las organizaciones tener control de su experiencia de comunicaciones. Utiliza seguridad adicional en los formatos SASL y TLS, además se pueden construir funcionalidades personalizadas sobre el núcleo del protocolo. Un amplio rango de compañías y proyectos Open Source utilizan XMPP para construir y desplegar aplicaciones en tiempo real y servicios. (4)

1.1.3.1. Ventajas.

- El intercambio de mensajes se realiza en formato XML.
- Es capaz, mediante pasarelas, de comunicarse con otros sistemas de MI[8]. El valor de un sistema de comunicaciones se incrementa con el número de personas con las que se pueda comunicar.
- Simplifica las responsabilidades del cliente a lo más mínimo.
- Ninguna organización, grupo o servicio controla el sistema. (4)

1.1.3.2. Desventajas.

- Jabber/XMPP presenta problemas debido a que es un protocolo hoy por hoy, inmaduro y en pleno desarrollo. La mayoría de la documentación oficial del estándar Jabber/XMPP está incompleta o sin actualizar y la única respuesta a las preguntas sobre el protocolo es probar el comportamiento del servidor de referencia de Jabber/XMPP.
- Además el protocolo sufre de ineficiencias directamente relacionadas con su naturaleza XML, ya que los formatos binarios de datos reducen el ancho de banda necesario para su transporte por la red, así como su procesamiento

[8] Mensajería instantánea. Es una forma de comunicación en tiempo real entre dos o más personas basada en texto. El texto es enviado a través de dispositivos conectados a una red como Internet.

tanto en el cliente como en el servidor. Además no disponen de la detección de errores y corrección. (4)

Este protocolo fue escogido para darle solución a la problemática ya que posee unas librerías muy potentes llamadas XMPP_PHP y Strophe, lo que permite la unión con el marco de trabajo SAUXE de forma sencilla.

1.1.4. EJABBERD.

EJabberd es un servidor multiplaforma desarrollado en Erlang y de código abierto. La instalación es sencilla, ya que la distribución de Erlang 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). (5)

1.1.4.1. Ventajas.

- Panel de administración web y herramientas de línea de comandos que aseguran una configuración sencilla.
- Soporte para encriptar conexiones para asegurar una comunicación segura. Conferencias.
- Clúster con múltiples servidores.
- Multiplataforma: Windows, Linux, MacOSX and Solaris.
- Recolección de estadísticas para reportes y análisis del sistema.
- Soporta Virtual hosting.
- Soporte para almacenamiento externo.
- Soporte para IPv6. (6)

1.1.4.2. Desventajas.

- 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 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, esta permitirá administrar

las ACL [9] (Access Control List), crear o dar de bajas a usuarios y ver información estadística del servidor, por otro lado no permite la configuración de Salas las cuales deben de ser creadas y configuradas desde los clientes. Por esta razón recomiendan los desarrolladores que se utilicen PSI o Tkabber (clientes para protocolo Jabber) para estas tareas. (5)

La razón por la cual esta tecnología no es escogida para la solución, es por estar implementada en Erlang, lo que traería consigo mayor complejidad a la hora de darle respuesta a la problemática en cuestión, ya que este lenguaje es poco conocido en la Universidad.

1.1.5. OPENFIRE.

Antes llamado Servidor Wildfire es un servidor Jabber/XMPP escrito en Java que provee licencias comerciales y GNU[10]. 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, etc. (7)

1.1.5.1. Ventajas.

- **Protocolo abierto:** Con todas las ventajas del software libre, se puede programar un servidor o un cliente o ver el código, entre otras cosas.
- **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. (7)

[9] Lista de Control de Acceso. Es un concepto de seguridad informática usado para fomentar la separación de privilegios. Es una forma de determinar los permisos de acceso apropiados a un determinado objeto, dependiendo de ciertos aspectos del proceso que hace el pedido.
[10] GNU es un acrónimo recursivo que significa GNU No es Unix.

1.1.5.2. Desventajas

- Requiere Java 5 (JRE 1.5 o superior) o Java 6 (es recomendado) en el servidor.
- Requiere base de datos con driver JDBC.

Características:

- Panel de administración web.
- Interfaz para agregar plugins.
- SSL/TLS.
- Adaptable según las necesidades.
- Conferencias.
- Interacción con MSN, Google Talk, Yahoo messenger, AIM, ICQ.
- Estadísticas del Servidor, mensajes, paquetes, etc.
- Clúster con múltiples servidores.
- Transferencia de Archivos.
- Compresión de datos.
- Mensajes offline.
- Favoritos.
- Autenticación vía Certificados, Kerbeos, LDAP, PAM y Radius.
- Almacenamiento en Active Directory, LDAP, MS SQL, MySQL, Oracle y PostgreSQL.
- SASL: ANONYMOUS, DIGEST-MD5 y Plain. (7)

Este servidor de mensajería esta implementado en JAVA, un lenguaje bien conocido por la comunidad universitaria, lo que trae consigo menor complejidad en el desarrollo de la solución.

1.1.6. cURL.

cURL es una herramienta para usar en un intérprete de comandos para transferir archivos con sintaxis URL, soporta FTP, FTPS, HTTP, HTTPS, TFTP, SCP, SFTP, Telnet, DICT, FILE y LDAP. cURL soporta certificados HTTPS, HTTP POST, HTTP PUT, subidas FTP, Kerberos, subidas mediante formulario HTTP, proxies, cookies, autenticación mediante usuario + contraseña (Basic, DIgest, NTLM y Negotiate para HTTP y kerberos4 para FTP), continuación de transferencia de archivos, tunneling de

proxy http y muchas otras prestaciones. cURL es open source/software libre distribuido bajo la Licencia MIT. (8)

El principal propósito y uso para cURL es automatizar transferencias de archivos o secuencias de operaciones no supervisadas. Es una buena herramienta para simular las acciones de un usuario en un navegador web. Libcurl es la biblioteca/API correspondiente que los usuarios pueden incorporar en sus programas; cURL actúa como un envoltorio (wrapper) aislado para la biblioteca libcurl. Libcurl se usa para proveer capacidades de transferencia de URL a numerosas aplicaciones, tanto libres y open source como así también privativas. La biblioteca "libcurl" se puede usar desde más de 30 lenguajes distintos. (8)

Esta tecnología además de tener un amplio soporte en lenguaje PHP, lo que posibilita su uso en el marco de trabajo, permite hacer peticiones al plugin utilizando el protocolo HTTP para el envío de datos y con esto la unión del sistema ACAXIA con el servidor Openfire a través del plugin.

1.2. Metodologías de desarrollo.

Se conoce como metodología al conjunto de procedimientos, técnicas, herramientas y al soporte documental que ayuda a los desarrolladores a realizar un software. Una metodología define quién debe hacer qué, cuándo y cómo debe hacerlo. (9)

1.2.1. Modelo de desarrollo propuesto.

La propuesta de modelo de desarrollo fue elaborada por el equipo de producción del CEIGE teniendo en cuenta los principales riesgos con los que se cuentan en 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, buenas prácticas propuestas y algunos elementos de las metodologías SCRUM y RUP, fue elaborado teniendo en cuenta las características especiales que presenta la UCI, por ejemplo: casi todos los proyectos están compuestos en su mayoría por estudiantes. Tiene un valor social representativo, ya que implica mejoría en aspectos importantes como la planeación, formación y satisfacción del equipo de trabajo. 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, pudiendo así ahorrar grandes sumas de dinero al país. (10)

1.2.1.1. 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. (10)

- **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. (10)

- **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. (10)

- **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. (10)

1.3. Tecnologías empleadas.

Las tecnologías que ahora serán descritas han sido definidas y adoptadas por el grupo de desarrolladores del centro, de acuerdo con las características del software.

1.3.1. Lenguajes de Programación.

1.3.1.1. Java.

Java es un lenguaje orientado a objetos, eso implica que su concepción es muy próxima a la forma de pensar humana. También posee otras características muy importantes:

- Es un lenguaje multiplataforma: el mismo código java que funciona en un sistema operativo, funcionará en cualquier otro sistema operativo que tenga instalada la máquina virtual java.
- Es un lenguaje seguro: la máquina virtual, al ejecutar el código java, realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras, como por ejemplo los punteros.
- Gracias al API de java se puede ampliar el lenguaje para que sea capaz de, por ejemplo, comunicarse con equipos mediante red, acceder a bases de datos, crear páginas HTML dinámicas, crear aplicaciones visuales al estilo Windows. (11)

1.3.1.2. PHP.

PHP, acrónimo de "Preprocesador de Hipertexto" por sus siglas en inglés, es un lenguaje interpretado de alto nivel embebido en páginas HTML. La mayoría de su sintaxis es similar a C, Java y Perl, con solamente un par de características PHP específicas. La meta de este lenguaje es permitir escribir a los creadores de páginas web, páginas dinámicas de una manera rápida y fácil. Una de sus características más potentes es su soporte para gran cantidad de bases de datos. Entre su soporte pueden mencionarse InterBase, mSQL, MySQL, Oracle, Informix, PostgreSQL, entre otras. PHP también ofrece la integración con varias bibliotecas externas. Como producto de código abierto, goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y se reparan rápidamente. El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP. (12)

1.3.2. Librerías y Marcos de trabajo.

1.3.2.1. Zend Framework.

Zend Framework no es más que un framework MVC (Modelo-Vista-Controlador) open-source 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, aunque todos estos en conjunto conforman un potente y extensible framework para aplicaciones web. Brinda una alta abstracción de bases de datos, haciendo extremadamente simple la interacción con estas, sin necesidad de escribir ninguna consulta SQL. Cuenta con módulos para el manejo de ficheros PDF, canales RSS, entre otros. Cuenta con clientes para el acceso a WS[11] y robustas clases para la autenticación y el filtrado de entrada; completa documentación y test de alta calidad. (13)

1.3.2.2. SAUXE.

Sauxe 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. Al iniciar un desarrollo sobre Sauxe, se tiene garantizado aspectos como la seguridad, la multi-entidad, el multi-tema, el multi-idioma, la auditoría, la integración, la interoperabilidad, la concurrencia, la administración de transacciones, entre otros. Por las ventajas que proporciona se reutiliza en más de 20 proyectos de la UCI y 10 entidades desarrolladoras de software. (14)

1.4. Herramientas del Desarrollo.

Las herramientas de desarrollo son aquellas aplicaciones que tienen cierta importancia en el desarrollo de un programa, como es el caso de las que se exponen a continuación con respecto a la solución propuesta.

[11] Servicio web. Es una pieza de software que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.

1.4.1. XAMPP.

XAMPP es un servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor Web Apache y los intérpretes para lenguajes de script: PHP y Perl. El nombre proviene del acrónimo de X (para cualquiera de los diferentes sistemas operativos), Apache, MySQL, PHP, Perl. El programa está liberado bajo la licencia GNU y actúa como un servidor Web libre, fácil de usar y capaz de interpretar páginas dinámicas. Actualmente XAMPP esta disponible para Microsoft Windows, GNU/Linux, Solaris, y MacOS X. (15)

1.4.2. Eclipse 4.07.

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Eclipse dispone de un Editor de texto con resaltado de sintaxis. La compilación es en tiempo real. Tiene pruebas unitarias con JUnit, control de versiones con CVS, integración con Ant, asistentes (wizards) para creación de proyectos, clases, tests, etc., y refactorización. (16)

1.4.3. Apache Ant.

Apache Ant es una herramienta usada en programación para la realización de tareas mecánicas y repetitivas, normalmente durante la fase de compilación y construcción (build). Es, por tanto, un software para procesos de automatización de compilación, similar a Make pero desarrollado en lenguaje Java y requiere la plataforma Java, así que es más apropiado para la construcción de proyectos Java. Esta herramienta, hecha en el lenguaje de programación Java, tiene la ventaja de no depender de las órdenes del Shell de cada sistema operativo, sino que se basa en archivos de configuración XML y clases Java para la realización de las distintas tareas, siendo idónea como solución multi-plataforma. La diferencia más notable entre Ant y Make es que Ant utiliza XML para describir el proceso de generación y sus dependencias, mientras que Make utiliza formato makefile. Por defecto, el archivo XML se denomina build.xml. Ant es un proyecto de la Apache Software Foundation. Es software open source, y se lanza bajo la licencia Apache Software. (17)

1.4.4. PostgreSQL 8.3.

PostgreSQL versión 8.3 es un gestor de bases de datos relacional orientado a objetos, libre y gratuito. Presenta las siguientes propiedades:

- Atomicidad: asegura la realización de una operación, por lo que ante un fallo del sistema, esta no queda a medias.
- Consistencia: posibilita la ejecución de aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos.
- Aislamiento: mediante un sistema de acceso concurrente multiversión (Multiversion concurrency control, MVCC[12]) asegura que una operación no pueda afectar a otras, de esta manera dos transacciones sobre la misma información no genera error. (18)

1.4.5. Visual Paradigm 5.0.

Visual Paradigm es una herramienta CASE: Ingeniería de Software Asistida por Computación. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Soporta las últimas versiones del mismo, (Lenguaje de Modelado Unificado) y la Notación y Modelado de Procesos de negocios, además de un generador de mapeo de objetos-relacionales para los lenguajes de programación Java .NET y PHP. Se integra con las siguientes herramientas de java: Eclipse, WebSphere, JBuilder, NetBeans IDE, Oracle JDeveloper, BEA Weblogic. (19)

[12] Control de Concurrencia Multiversión (MVCC). Es un método de control de concurrencia comúnmente utilizado por los sistemas de gestión de bases de datos para proporcionar acceso simultáneo a la base de datos.

1.5. Conclusiones Parciales.

Una vez concluido el marco teórico de la investigación se arribó a las siguientes conclusiones:

El protocolo XMPP ha tenido en los últimos tiempos un gran auge con respecto a la mensajería instantánea, debido que es una solución sencilla, que evita en su mayor parte las dependencias de plataformas, fabricantes y además es una de las más usadas a nivel mundial.

El servidor de mensajería instantánea Openfire basado en el protocolo XMPP representa un potencial para dar solución al problema planteado por ser desarrollado por la comunidad de software libre y por sus características que ampliamente satisfacen el desarrollo para dar respuesta al problema planteado.

Las herramientas y tecnologías descritas anteriormente resuelven el desarrollo de la solución.

Por lo antes expuesto se propone la implementación de un plugin para el servidor de mensajería instantánea Openfire que permita el proceso de gestión de usuarios y autenticación de los mismos a partir de la seguridad que brinda ACAXIA.

Capítulo 2: Características, análisis y diseño del sistema.

Introducción.

El presente capítulo está dedicado a las características del sistema, es por esto que se lleva a cabo un análisis de la propuesta de solución, así como de conceptos basados en esta, como es el caso de plugin. Se demuestra cómo se realizará la interacción del sistema con el servidor de mensajería Openfire y ACAXIA a través de un mapa conceptual. Se mencionan los requisitos funcionales y no funcionales que debe cumplir la aplicación y además, se explican los patrones utilizados como es el caso del Modelo-Vista-Controlador y el diagrama de clases del diseño realizado para la aplicación.

2.1. Propuesta de Solución.

Como propuesta de solución se presenta la creación de un plugin para el servidor de mensajería instantánea Openfire, el cual debe enviar mensajes a los usuarios registrados a partir del protocolo XMPP (Extensible Messaging and Presence Protocol). El plugin recibirá peticiones vía HTTP a través de una clase llamada OpenFire.php que permitirá al Sistema Integral de Gestión de Seguridad ACAXIA, enviar los datos necesarios para gestionar los usuarios de Openfire y su posterior autenticación.

2.1.1. Concepto de Certificado de Seguridad.

Son una medida de confianza adicional para las personas que visitan y hacen transacciones en su página web, le permite cifrar los datos entre el ordenador del cliente y el servidor que representa a la página. El significado más preciso de un certificado de seguridad es que con él logramos que los datos personales sean encriptados y así imposibilitar que sean interceptados por otro usuario. Actualmente es muy común ver en los navegadores el protocolo de seguridad https; mediante éste, básicamente se dice que la información que se envía a través de internet, entre el navegador del cliente y el servidor donde está alojada la página, se encripta de forma

que es casi imposible que otra persona reciba, vea o modifique los datos confidenciales del cliente. (20)

2.1.2. Concepto de Plugin.

Un plugin es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande. Un programa puede tener uno o más conectores. Son muy utilizados en los navegadores para ampliar sus funcionalidades. (21)

2.1.3. Concepto de Notificación.

Una notificación está vinculada a una comunicación, aviso o alerta. Al enviar una notificación, una empresa, organización o persona pretende establecer determinada resolución que se ha tomado o que se tomará en un futuro. (22)

2.1.4. Plugin de Openfire.

Openfire es un servidor de Mensajería Instantánea con muy poco años de experiencia, esto no quiere decir que no presente una fuerte comunidad de usuarios que apoyan su desarrollo, es por esto que cuenta con una vasta librería de plugin que han sido la manera más eficiente de suplir las carencias que aun presenta. Todo esto puede ser posible a partir de la interfaz gráfica llamada consola de administración del Openfire que permite agregar plugins al mismo en formato jar. La creación de los plugins está dada en dependencia de necesidades de los usuarios.

2.1.5. Modelo conceptual.

El modelo conceptual es una representación de conceptos en un dominio del problema. Este modelo muestra asociaciones entre conceptos y atributos de conceptos. Se puede ver como un modelo que comunica los términos importantes y cómo se relacionan entre sí. (23)

2.1.6. Representación del modelo conceptual.

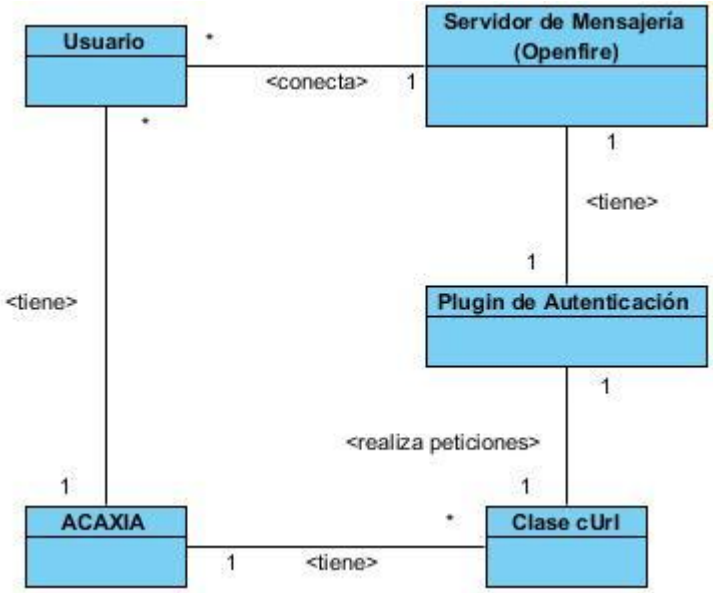


Figura 1: Modelo conceptual.

2.1.6.1. Entidades, conceptos y relaciones.

Para desarrollar un correcto modelo conceptual es necesario realizar una investigación de los principales conceptos utilizados en el entorno de trabajo. Por lo que a continuación se presentan algunos que son de vital importancia para el desarrollo del plugin.

Usuario: es quien va a interactuar con la aplicación y realizar la solicitud de autenticación.

Servidor de mensajería Instantánea Openfire: permite el envío de mensajes a los usuarios registrados a partir de la autenticación.

Plugin de autenticación: es el encargado de permitir la autenticación de los usuarios a partir del nombre del usuario y los servicios que brinda ACAXIA.

cURL: librería para PHP que permite intercambiar datos entre el plugin a desarrollar y ACAXIA a través del protocolo HTTP.

ACAXIA: es el sistema de gestión de seguridad que brinda los servicios web pertinentes para poder autenticar a los usuarios del sistema que ya se encuentran registrados.

2.1.7. Especificación de los requerimientos.

La ingeniería de requisitos facilita el mecanismo apropiado para comprender lo que quiere el cliente, analizando necesidades, confirmando su viabilidad, negociando una solución razonable, especificando la solución sin ambigüedad, validando la especificación y gestionando los requisitos para que se transformen en un sistema operacional. Los requerimientos para un sistema de software determinan lo que hará el sistema y definen las restricciones de su operación e implementación. **(24)**

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. (23)

2.1.7.1. Requisitos funcionales.

De acuerdo a la experiencia de los analistas del centro CEIGE, se identificaron un total de 5 requisitos funcionales que debe cumplir el plugin, a continuación se listan dichos requisitos funcionales:

- Gestionar usuarios en el servidor:
    - Adicionar usuario.
    - Modificar usuario.
    - Eliminar usuario.
- Autenticar usuario.
- Gestionar IPs que hacen peticiones al servidor:
    - Adicionar uno o varios IP
    - Modificar uno o varios IP
    - Eliminar uno o varios IP
- Generar y modificar clave secreta.
- Activar o desactivar el plugin.

Tabla 1. Especificación de Requisito Adicionar usuario.

Precondiciones	El servidor Openfire debe estar iniciado. Debe existir conexión de red.
Flujo de eventos	
Flujo básico Insertar Usuario	
1	El sistema permitirá insertar todos los datos del usuario.
2	El sistema inserta los datos que intervienen en el momento de insertar el usuario: Datos de usuario username, password, secret y type.
3	El sistema muestra un mensaje de confirmación.
4	Concluye el requisito.
Pos-condiciones	
1	Se registraron los datos del usuario.
Flujos alternativos	
Flujo alternativo 2.a Campos vacíos o erróneos	
1	El sistema muestra un mensaje de error, notificando que existen campos vacíos o erróneos e indica cuáles son estos.
2	Continúa en el paso 2 del flujo básico.
Pos-condiciones	
1	N/A
Validaciones	
1	Se validan los datos según lo establecido en el Modelo conceptual <<Referencia al modelo conceptual en cuestión>>.
Conceptos	**Usuario** Utilizados internamente: o username o password o secret o type
Requisitos especiales	N/A
Asuntos pendientes	Posibles mejoras al requisito.

La descripción de los demás requisitos mencionados se incluyen en el Anexo 1.

2.1.7.2. Requisitos no funcionales.

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. De esta manera los requisitos no funcionales a los que se debe acoger la aplicación a desarrollar son los que fueron establecidos por el centro al inicio del proceso de desarrollo. A continuación son descritos algunos de los más importantes.

Software:

A nivel del cliente o usuario el plugin puede ser accedido desde cualquier plataforma o sistema operativo, solo es necesario contar el servidor Openfire instalado así como una buena conexión a la red y un navegador web. Aunque se recomienda el uso del navegador Mozilla Firefox 3.0 o superior para su funcionamiento óptimo. (25)

Hardware:

Para el servidor:

- Requerimientos mínimos: Procesador Pentium IV a 2GHz de velocidad de procesamiento y 1Gb de memoria RAM.
- Al menos 40Gb de espacio libre en disco duro.
- Tarjeta de red. (25)

Para el cliente:

- Requerimientos mínimos: Procesador Pentium III a 1GHz con 256Mb de memoria RAM.
- Tarjeta de red. (25)

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. (25)

Confiabilidad:

El plugin deberá contar con un máximo de disponibilidad por lo que podrá ser usado siempre que el servidor Openfire esté ejecutándose. El tiempo medio de reparación se estima que sea menor de 1 día.

Requerimientos del diseño y la implementación:

- Se debe implementar en el lenguaje JAVA.
- Para el análisis y el diseño del plugin se utilizará la metodología definida por el centro, usando el lenguaje de modelación UML y como herramienta para llevar a cabo el modelado Visual Paradigm.
- El IDE que se utilizara para la construcción de dicho plugin es ECLIPSE.

2.1.8. Prototipo de interfaz de usuario.

Para un mejor entendimiento y desarrollo de la implementación del plugin se comienza a definir y diseñar las principales vistas que van a interactuar con el usuario final, en busca de aminorar la complejidad y abstracción que puede conllevar la implementación de un software para los desarrolladores y también con el objetivo de evaluar el nivel de satisfacción del cliente y garantizar el entendimiento con este. A continuación se muestra el prototipo desarrollado para la interfaz principal (Figura 2). (23)

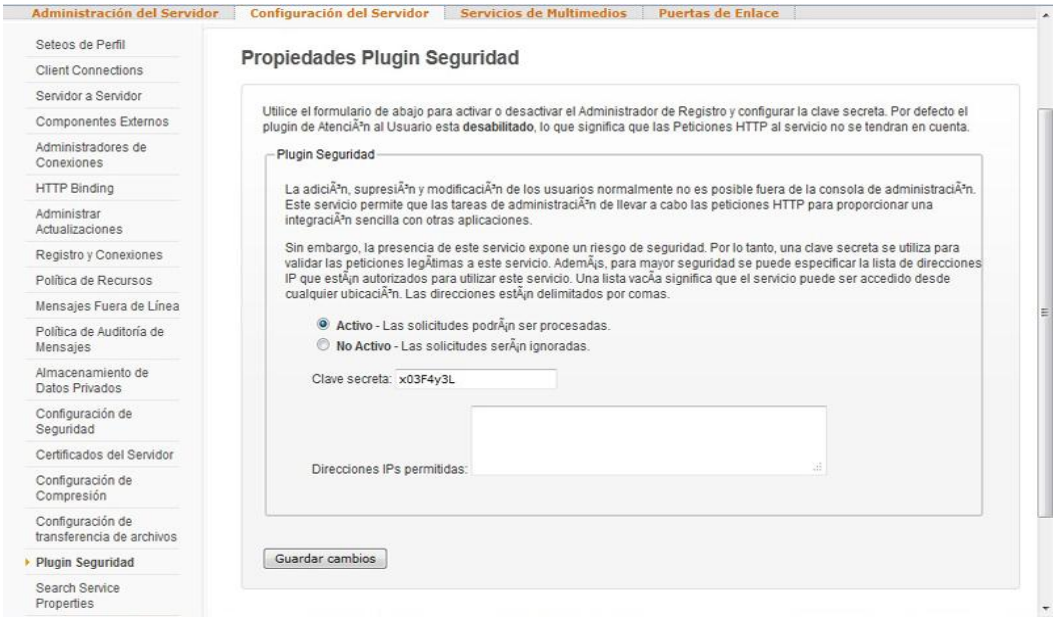


Figura 2. Interfaz principal

2.2.Diagrama de clases del diseño.

El diagrama de clases del diseño describe la estructura del sistema, mostrando sus clases, asociaciones, atributos, métodos y relaciones entre ellos. (23). A continuación en la Figura 3 se muestra el diagrama de clases del diseño con estereotipos web perteneciente a la solución:

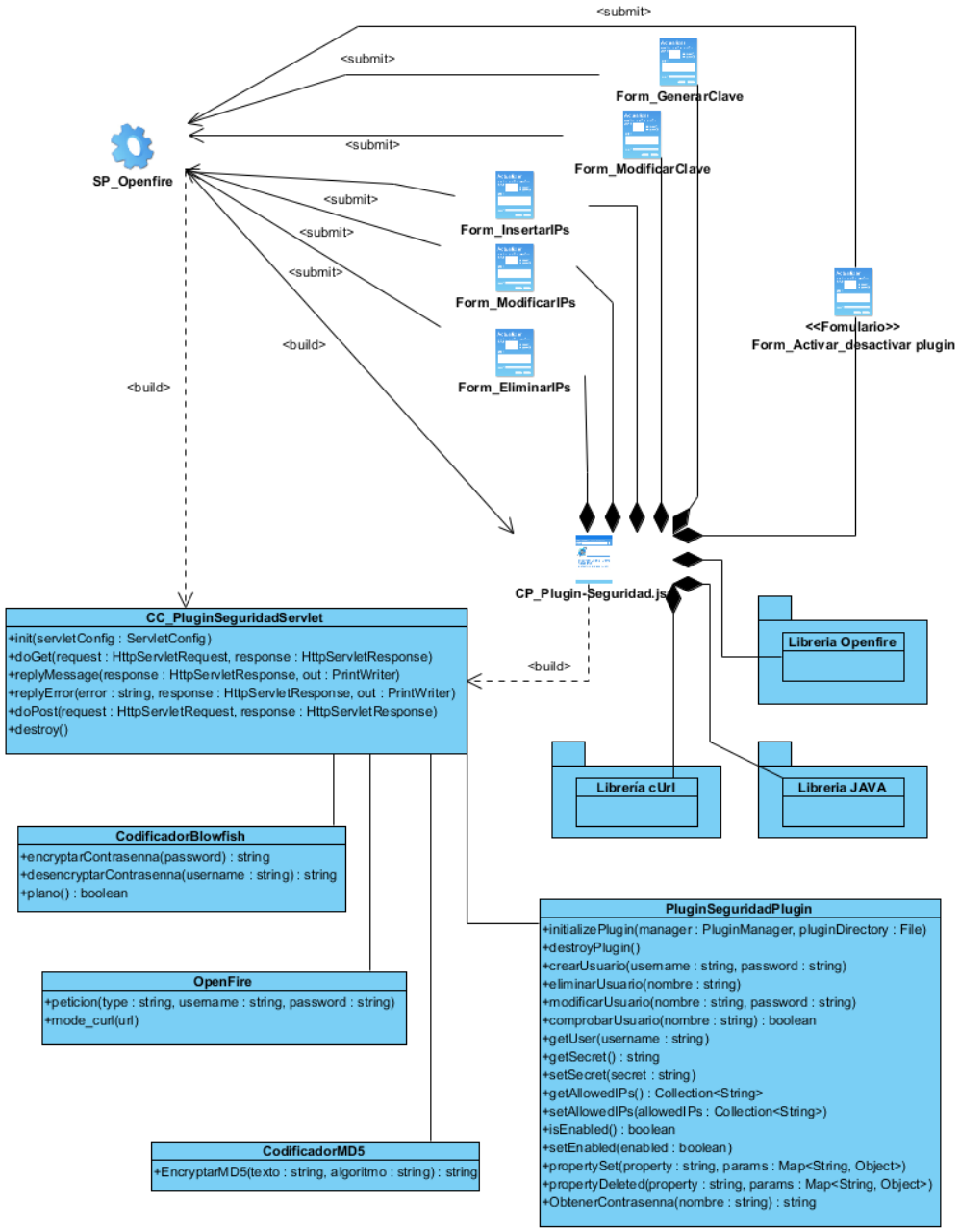


Figura 3. Diagrama de clases del diseño con estereotipos web

2.2.1. Patrones de diseño utilizados.

Un patrón de diseño es una abstracción de una solución en un nivel alto. Los patrones solucionan problemas que existen en muchos niveles de abstracción. Hay patrones que abarcan las distintas etapas del desarrollo; desde el análisis hasta el diseño y desde la arquitectura hasta la implementación. (26)

Con la aparición del J2EE, todo un nuevo catálogo de patrones de diseño apareció. Desde que J2EE es una arquitectura por si misma que involucra otras arquitecturas, incluyendo servlets, JavaServer Pages, Enterprise JavaBeans, y más, merece su propio conjunto de patrones específicos para diferentes aplicaciones empresariales. (26)

De acuerdo al libro "J2EE PATTERNS Best Practices and Design Strategies", existen 5 capas en la arquitectura J2EE: Cliente, Presentación, Negocios, Integración y Recurso. El libro explica 15 patrones J2EE que están divididos en 3 de las capas: presentación, negocios e integración. (26)

Capa de presentación:

Intercepting Filter: un objeto que está entre el cliente y los componentes Web. Este procesa las peticiones y las respuestas. (26)

Composite view: un objeto vista que está compuesto de otros objetos vista. Por ejemplo, una página JSP que incluye otras páginas JSP y HTML usando la directiva include o el action include es un patrón Composite View. (26)

Front Controller: Un objeto que acepta todos los requerimientos de un cliente y los direcciona a manejadores apropiados. El patrón Front Controller podría dividir la funcionalidad en 2 diferentes objetos: el Front Controller y el Dispatcher. En ese caso, El Front Controller acepta todos los requerimientos de un cliente y realiza la autenticación, y el Dispatcher direcciona los requerimientos a manejadores apropiada. (26)

Capa de Negocio:

Business Delegate: un objeto que reside en la capa de presentación y en beneficio de los otros componentes de la capa de presentación llama a métodos remotos en los objetos de la capa de negocios. (26)

Capa de Integración:

Service Activator: Se utiliza para recibir peticiones y mensajes asíncronos de los clientes. Cuando se recibe un mensaje, el Service Activator localiza e invoca a los métodos de los componentes de negocio necesarios para cumplir la petición de forma asíncrona. (26)

2.2.1.1. Patrón arquitectónico Modelo-Vista-Controlador (MVC).

El Modelo Vista Controlador (**MVC**) es un patrón arquitectónico que tiene como principal característica separar los datos de una aplicación en tres módulos identificables y con funcionalidades bien definidas: el Modelo, las Vistas y el Controlador, permitiendo flexibilidad y facilidad a la hora de hacer futuros cambios. (23)

El **modelo** es un conjunto de clases que representan la información del mundo real que el sistema debe procesar. (27) En el mismo se encuentran todas las funciones para consultar, insertar y actualizar información de la base de datos.

Las **vistas** son el conjunto de clases que se encargan de presentar al usuario la información contenida en el modelo. Una vista está asociada a un modelo, pudiendo existir varias vistas asociadas al mismo modelo. Estas pueden ser una página web o una parte de una página y obtiene del modelo solamente la información que necesita para desplegar y se actualiza cada vez que el modelo del dominio cambia por medio de notificaciones. (23)

El **controlador** es un objeto que se encarga de dirigir el flujo del control de la aplicación debido a mensajes externos, como datos introducidos por el usuario u opciones del menú seleccionadas por él. A partir de estos mensajes, el controlador se encarga de modificar el modelo o de abrir y cerrar vistas. El controlador tiene acceso al modelo y a las vistas, pero las vistas y el modelo no conocen de la existencia del controlador. (28)

2.2.2. Conclusiones parciales.

Con la realización de este capítulo se lograron los objetivos trazados, al realizar un profundo análisis y diseño de la solución. En el mismo se exponen los diferentes artefactos realizados como son la descripción de los procesos de negocios, el modelo conceptual, la descripción de los requisitos funcionales, no funcionales, el diagrama de clases del diseño con estereotipos web, así como los patrones de diseño utilizados en la solución, además de mostrar los prototipos de interfaces. Una vez concluida esta fase de la solución se procederá a la realización de los flujos de implementación y prueba de la misma.

Capítulo 3: Implementación y Pruebas.

Introducción.

En el siguiente capítulo se muestran en primer lugar los estándares de codificación, así como los diferentes artefactos que se desarrollaron en la fase de implementación, como el diagrama de componente y el diagrama de despliegue. Y se especifica el conjunto de validaciones y pruebas que evalúan la calidad del componente desarrollado.

Estándares de Codificación.

Se define como estándar de codificación a un estilo de programación en un proyecto determinado permitiendo que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea mantenido. (23)

Debido al numeroso personal involucrado en la implementación del sistema CEDRUX, su complejidad y el alto nivel de integración, se define un estándar de codificación para JAVA con el fin de obtener un estándar en la implementación que permita asegurar la calidad del software y de esta forma obtener un código más legible y reutilizable. A continuación se describe el estándar empleado en la implementación del Plugin. (23)

3.1. Estándar de Codificación de JAVA.

3.1.1. Nombre de los archivos.

Los nombres de archivos deben ser sustantivos y comenzar con letra mayúscula, si está compuesto por más de una palabra se forma el nombre con todas las palabras juntas y la primera letra de cada palabra con letra mayúscula. Solamente se permite el uso de palabras completas, no se pueden usar abreviaturas y el nombre debe describir correctamente al archivo.

Ejemplo: Empresa, EmpresaAzucarera.

3.1.2. Organización de archivos.

Cada archivo contiene varias secciones (funciones) y deben separarse con un espacio en blanco entre ellas.

Los archivos con más de 1000 líneas de códigos deben ser evitados debido a que dificultarían un posterior mantenimiento.

3.1.2.1. Comentarios al inicio.

Todos los archivos deben empezar con un comentario donde se especifique el autor del fichero y de manera opcional se puede poner una breve descripción de su propósito.

3.1.3. Identación.

Se usará 1 tab como unidad de identación.

3.1.3.1. Longitud de línea.

Evitar líneas mayores de 80 caracteres.

3.1.3.2. Estructura de las líneas.

Cuando una expresión no quepa en una sola línea, se debe dividir siguiendo los siguientes principios:

- Dividir después de una coma.
- Dividir después de un operador (lógico o aritmético).
- Alinear la nueva línea al principio al mismo nivel que la expresión de la línea anterior.

3.1.4. Comentarios.

Existen dos tipos de comentarios, los comentarios de implementación y los comentarios de documentación. Los de implementación se delimitan con /*……*/ y con //; mientras que los de documentación se delimitan con /**……*/.

Los comentarios de implementación se usan para explicar partes de código que sean un poco difíciles de entender (muy útil y absolutamente necesario a la hora de dar mantenimiento al código). Los comentarios de documentación se usan para ofrecer una descripción de las funciones.

Es de carácter obligatorio en el proyecto que se realicen los comentarios de documentación de todas las funciones. Los comentarios de implementación son a

criterio del programador según la complejidad del código escrito, pero a la vez se deben evitar comentarios innecesarios.

3.1.5. Declaraciones.

3.1.5.1. Cantidad de declaraciones por línea.

Es recomendado que solamente exista una declaración por línea, ello facilita la posibilidad de añadirle comentarios a cada línea en caso de ser necesario.

3.1.5.2. Lugar de las declaraciones.

Poner todas las declaraciones al principio de cada bloque de código (se denota bloque de código al área que está entre dos llaves "{" y "}").

La única excepción a esa regla es la declaración de las variables de control en los ciclos.

3.1.5.3. Inicialización de variables.

Siempre tratar de inicializar las variables en el mismo lugar donde son declaradas. La única razón para no hacerlo es que su valor inicial dependa de alguna operación computacional previa.

3.1.5.4. Declaración de funciones y de clases.

Se deben seguir las siguientes reglas a la hora de declarar clases y funciones en Java:

- No debe haber espacios entre el nombre de las funciones y el paréntesis "(" que da inicio a la lista de parámetros.
- La llave de apertura "{" al final de la misma línea de la declaración.
- La llave de cierre "}" empieza una línea por ella misma al mismo nivel de identación que la línea donde se abrió, excepto cuando no haya nada entre ambas llaves, estaría la del cierre a continuación de la de apertura.
- Las funciones se separan por una línea en blanco.

3.1.6. Expresiones.

3.1.6.1. Expresiones simples.

Cada línea debe contener como máximo una sola expresión. No se puede hacer lo siguiente:

cantidadUsad++; cantidadDisponible--; //ERROR!!!

3.1.6.2. Expresiones compuestas.

Las expresiones compuestas son expresiones que contienen una lista de expresiones agrupadas entre llaves "{…}". Se debe cumplir con las siguientes regulaciones:

- Cuando se usan estructuras de control (if-else, for, switch, etc.) siempre se debe poner las llaves aunque sea una sola expresión lo que llevan dentro. Esto evita posibles errores futuros si se añaden nuevas expresiones internas y se olvidan los paréntesis.
- La llave de apertura se debe poner al final de la línea donde comienza la expresión compuesta, la llave de cierre debe empezar una línea nueva y estar identada con el principio de la línea de la expresión compuesta.
- Las expresiones internas deben estar identadas un nivel más que la expresión compuesta.

3.1.7. Espacios vacíos.

3.1.7.1. Líneas en blanco.

Las líneas en blanco mejoran la legibilidad del código al separar secciones lógicas dentro del mismo.

Se debe usar una línea en blanco siempre que pase alguna de las siguientes situaciones:

- Entre funciones.
- Entre secciones lógicas dentro de una misma función para mejorar la legibilidad.
- Después de las declaraciones de las variables.

3.1.7.2. Espacios en blanco.

Se deben usar los espacios en blanco en las siguientes circunstancias:

- Una palabra reservada seguida por un paréntesis debe separarse con un espacio en blanco, asimismo, la llave de apertura "{" siempre debe estar precedida por espacio en blanco. Ejemplo:

```
while (true) {
  ...
}
```

- Después de las comas en una lista de argumentos debe usarse un espacio en blanco.
- Todos los operadores binarios excepto el "**.**" deben separarse de sus operandos con un espacio en blanco. Por el contrario, con los operadores unarios ("++", "--") nunca se debe usar los espacios en blanco para separar. Ejemplos:

```
a += c + d;
a = (a + b) / (c * d);
while (d++ == s++) {
  n++;
}

alert("la longitud es " + longitud + "\n");
```

- Las expresiones de la instrucción *for* se deben separar con un espacio en blanco. Ejemplo:

```
for (expresion1; expresion2; expresion3)
```

3.1.8. Convenciones de nombres.

Todos los nombres que se usen tienen que ser en español, no se admite ningún nombre en otro idioma.

3.1.8.1. Clases.

Las clases deben tener el mismo nombre del archivo y por lo tanto cumplir con las normativas especificadas en el Capítulo 3.1.1.

3.1.8.2. Funciones.

Las funciones deben ser verbos, comenzar siempre con letra minúscula y en caso de ser varias palabras, se pondrían todas seguidas siendo la primera letra de la primera palabra minúscula y la primera letra de cada una de las restantes palabras sería mayúscula. No se puede usar abreviaturas y el nombre debe describir completamente la acción principal de la función.

3.1.8.3. Variables.

Se cumple lo mismo que en las funciones., aunque para el caso que se usen variables temporales se pueden usar abreviaturas (i, j, k, etc.).

3.1.8.4. Constantes.

Las constantes se escriben con mayúscula la palabra completa y si son más de una palabra se separan con un guión bajo "_".

3.2. Diagrama de componentes.

Un diagrama de componentes permite visualizar con más facilidad la estructura general del sistema y el comportamiento de los servicios que estos componentes proporcionan y utilizan a través de las interfaces. (23)

La solución consta de un solo componente denominado Plugin de seguridad que interactúa con el servidor de mensajería instantánea Openfire y otros del marco de trabajo Sauxe. A continuación en la Figura 4 se muestra el diagrama de componentes.

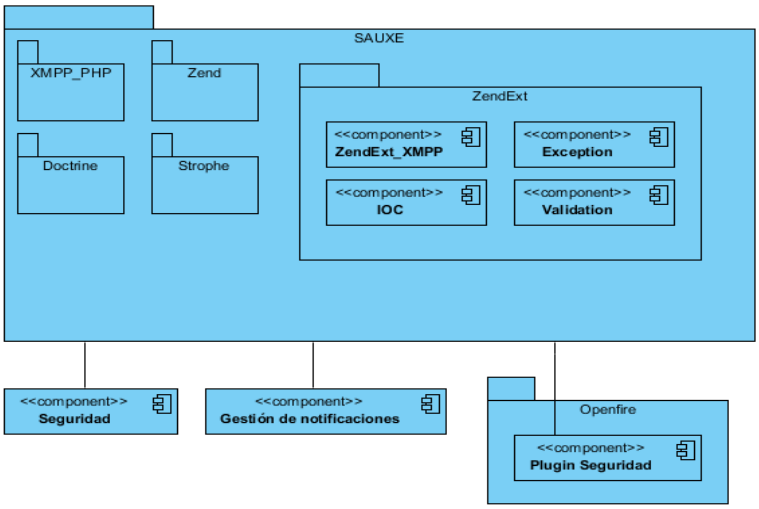


Figura 4. Diagrama de Componente.

3.3.Diagrama de despliegue.

Un diagrama de despliegue es un tipo de diagrama UML con el cual queda representado un modelado del hardware utilizado en la elaboración del software. En la Figura 5 se describen los dispositivos necesarios para la utilización del producto, es decir, desde dónde se encuentra instalado, dónde se puede utilizar, hasta dónde se conecta para tomar los datos. Es una vista panorámica que describe los requisitos de hardware y software mínimos, necesarios para que esta herramienta funcione. (23)

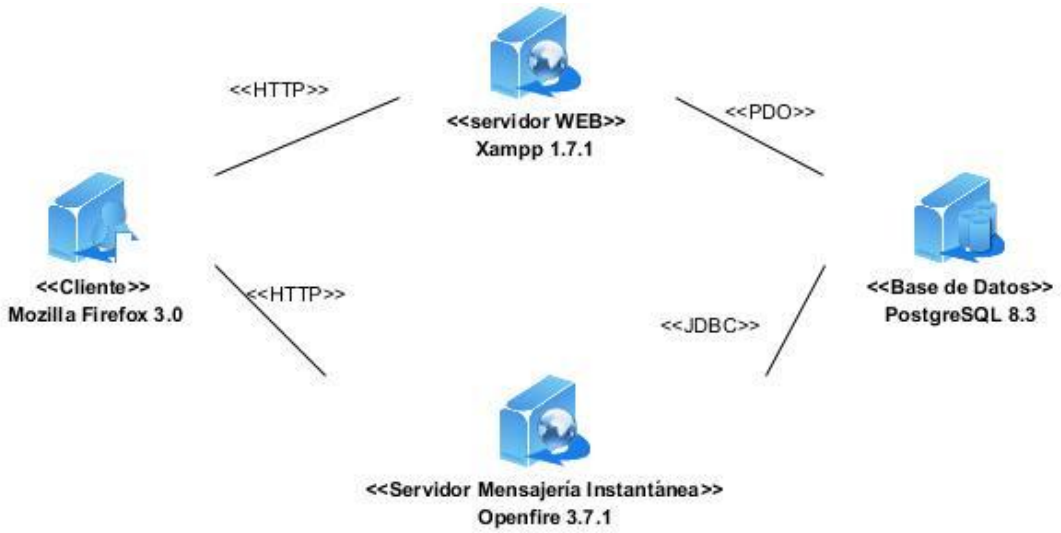


Figura 5. Diagrama de Despliegue.

Para el buen funcionamiento de la solución, es necesario tener una PC cliente que contenga Mozilla Firefox como navegador web en su versión 3.0 o superior, para poder relacionarse con el servidor Openfire mediante el protocolo HTTP y también con el servidor web utilizado, el cual se nombra Xampp. La conexión de Openfire y el servidor de base de datos PostgreSQL sucede a través de la API[13] que permite la ejecución de operaciones sobre bases de datos, conocida como JDBC[14]. El Xampp y el PostgreSQL se conectan mediante PDO[15].

[13] Interfaz de Programación de Aplicaciones. Es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
[14] Java Database Connectivity. Es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java.
[15] PHP Data Objects. es una extensión que provee una capa de abstracción de acceso a datos para PHP 5.

3.4.Métricas de software.

En el libro “Ingeniería del software, un enfoque práctico”, Pressman plantea: “El proceso del software y las métricas del producto son una medida cuantitativa que permite a la gente del software tener una visión profunda de la eficacia del proceso del software y de los proyectos que dirigen utilizando el proceso como un marco de trabajo. Se reúnen los datos básicos de calidad y productividad. Estos datos son entonces analizados, comparados con promedios anteriores, y evaluados para determinar las mejoras en la calidad y productividad. Las métricas son también utilizadas para señalar áreas con problemas de manera que se puedan desarrollar los remedios y mejorar el proceso del software”. (29)

Debido a que este software se realizó bajo la programación orientada a objetos (POO) y las clases constituyen la unidad básica y fundamental de un sistema orientado a objetos, es indiscutible que la validación del mismo se centró en la aplicación de métricas dirigidas a sus clases de forma individual, sus jerarquías y colaboraciones. (23) Las métricas empleadas fueron Tamaño Operacional de Clase (TOC) y Relaciones entre Clases (RC) las cuales están diseñadas para evaluar los siguientes atributos de calidad:

- **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.

A continuación se encuentran desarrolladas dichas métricas:

Tamaño operacional de clase (TOC):

El tamaño general de una clase se puede determinar empleando las medidas siguientes:

- El número total de operaciones (tanto operaciones heredadas como privadas de la instancia) que están encapsuladas dentro de la clase. (23)

- El número de atributos (tanto atributos heredados como atributos privados de la Instancia) que están encapsulados en la clase. (23)

Si existen valores grandes de TC éstos mostrarán que una clase puede tener demasiada responsabilidad, lo cual reducirá la reutilización de la clase y complicará la implementación y la comprobación, por otra parte cuanto menor sea el valor medio para el tamaño, más probable es que las clases existentes dentro del sistema se puedan reutilizar ampliamente. (23)

Tabla 2. Tamaño operacional de clase (TOC). (23)

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

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

Tabla 3. Rango de valores para los criterios de evaluación de la métrica Tamaño Operacional de Clase (TOC). (23)

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

Relaciones entre Clases (RC)

Esta métrica está dada por el número de relaciones diferentes con que cuenta y evalúa los siguientes atributos de calidad:

Tabla 4. Atributos de calidad que se evalúan en la métrica RC. (23)

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. Criterios de evaluación para la métrica RC. (23)

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

A continuación se realizará el análisis de los resultados obtenidos al aplicar las métricas al diseño planteado anteriormente.

3.4.1. Tamaño Operacional de Clases (TOC).

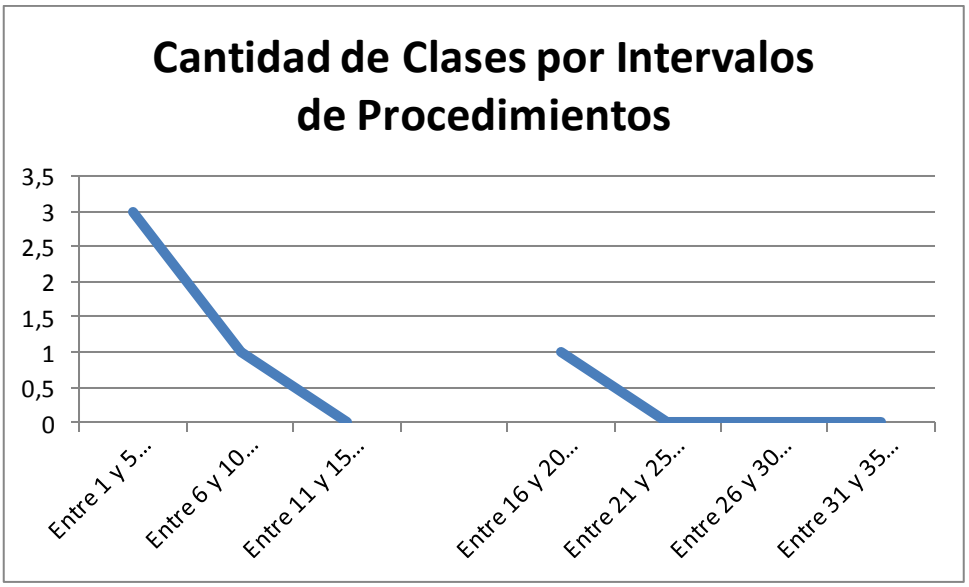
Los valores grandes para esta métrica, indican que la clase debe tener bastante responsabilidad. Esto reducirá la reutilización de esta clase y complicará la implementación y las pruebas. Se pueden calcular los promedios para el número de atributos y operaciones de clase. Cuando menor sea el valor del promedio para el tamaño será más posible que las clases dentro del sistema puedan ser reutilizadas.

El tamaño general de una clase se puede determinar empleando medidas para saber el número total de operaciones (tanto operaciones heredadas como privadas de la instancia) que están encapsuladas dentro de la clase así como encontrando el número de atributos (tanto atributos heredados como atributos privados de la Instancia) que están encapsulados en la clase. Si existen valores grandes de TOC éstos mostrarán que una clase puede tener demasiada responsabilidad, lo cual reducirá la reutilizabilidad de la clase y complicará la implementación y la comprobación, por otra parte cuanto menor sea el valor medio para el tamaño, más probable es que las clases existentes dentro del sistema se puedan reutilizar ampliamente. (23)

A continuación se muestra la tabla de resultados de la evaluación técnica y su influencia en los parámetros de calidad Responsabilidad, Complejidad de Implementación y Reutilización.

Tabla 6. Resultados de la evaluación técnica (TOC).

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
PluginSeguridadPlugin	16	Alta	Alta	Baja
PluginSeguridadServlet	6	Media	Media	Media
Openfire	2	Baja	Baja	Alta
CodificadorMD5	1	Baja	Baja	Alta
CodificadorBlowfish	3	Baja	Baja	Alta



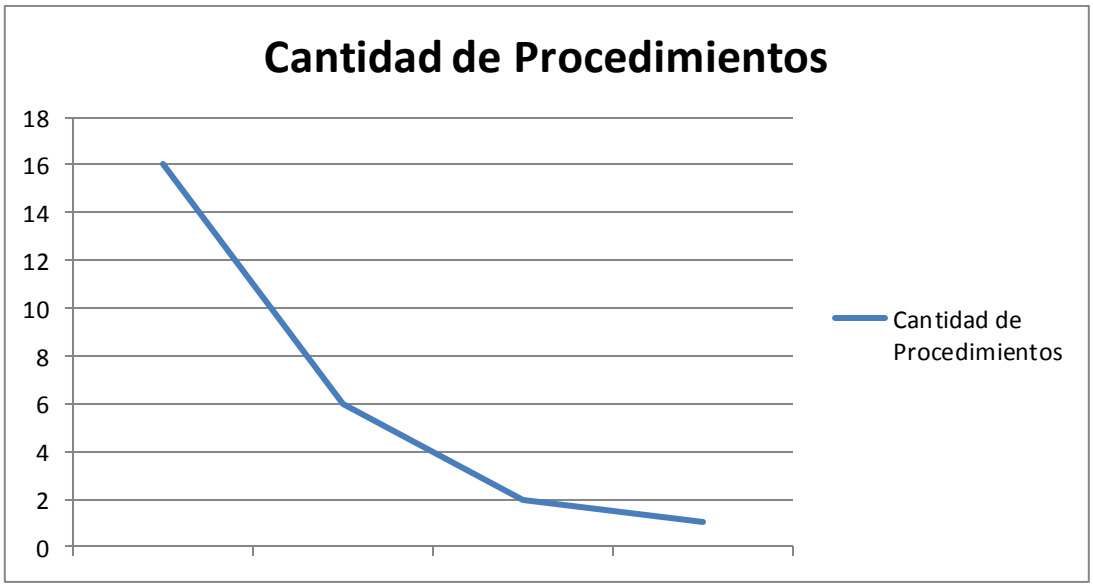
Gráfica 1: Representación de las clases según cantidad de operaciones.

Teniendo en cuenta las medidas o umbrales de referencia en lo que respecta al número de operaciones y/o atributos de las clases se establece que un tamaño de clase pequeño es aquel que tiene un valor menor o igual que 3, un tamaño medio para aquellas clases que tengan entre 4 y 10 operaciones y un tamaño de clase grande es aquel que es mayor o igual que 16. Se concluye que el plugin cuenta con 5 clases, para un promedio de cantidad de operaciones de 5.6. Los valores de tamaño quedan distribuidos de la siguiente manera:

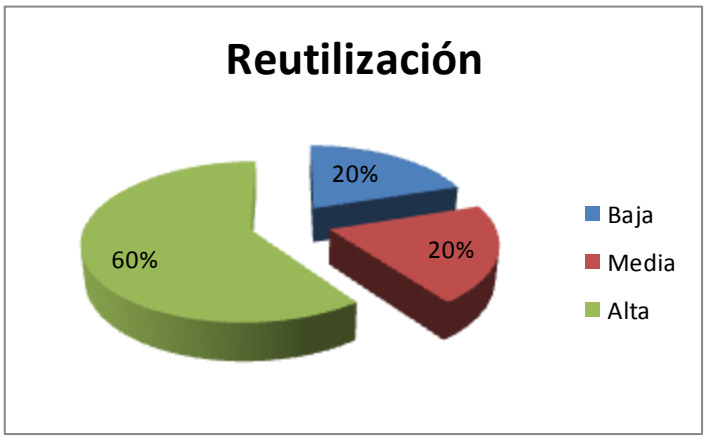
Tabla 7. Tamaño de Clases. (23)

Umbral	Tamaño	Cantidad de Clases
Pequeño	<=3	3
Medio	4<=x<=10	1
Grande	>=16	1

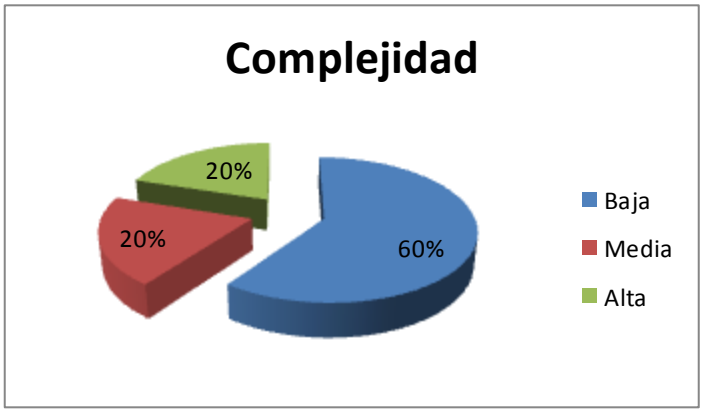
Como se puede observar el 60% de las clases están clasificadas en pequeñas, el 20% están en el umbral de clases de tamaño medio y el 20% en el umbral de clases grandes, lo cual brinda un resultado positivo según los parámetros de calidad Responsabilidad, Complejidad de Implementación y Reutilización propuestos para esta métrica tratada en este sub-epígrafe.



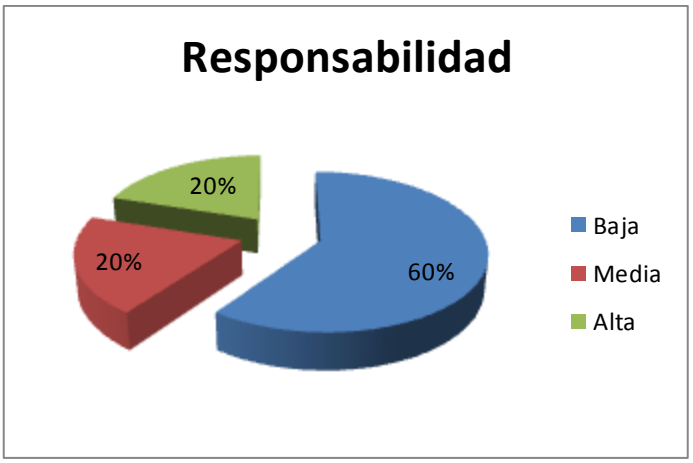
Gráfica 2: Cantidad de operaciones.



Gráfica 3: Reutilización.



Gráfica 4: Complejidad.



Gráfica 5: Responsabilidad.

Tabla 8: Intervalos definidos para procedimientos.

Intervalos	Cantidad de Clases
Entre 1 y 5 procedimientos	3
Entre 6 y 10 procedimientos	1
Entre 11 y 15 procedimientos	0
Entre 16 y 20 procedimientos	1
Entre 21 y 25 procedimientos	0
Entre 26 y 30 procedimientos	0
Entre 31 y 35 procedimientos	0

3.4.2. Relaciones entre Clases (RC).

Los valores grandes de RC implican un aumento del Acoplamiento, en la complejidad del Mantenimiento de la clase y en la cantidad de Pruebas de unidad necesarias para probar una clase por lo que traería consigo una mayor complejidad. Un valor grande de RC implicaría una disminución en el grado de Reutilización de la clase.

A continuación se muestra la tabla de resultados de la evaluación técnica y su influencia en los parámetros de calidad Acoplamiento, Complejidad de Mantenimiento, Cantidad de Pruebas y Reutilización.

Tabla 9: Resultados de la evaluación técnica. RC

- Clase – C
- Cantidad de Relaciones de Uso – CRU
- Acoplamiento – A
- Complejidad de Mantenimiento – CM

- Reutilización – R
- Cantidad de Pruebas - CP

C	CRU	A	CM	R	CP
PluginSeguridadPlugin	1	Baja	Baja	Alta	Baja
PluginSeguridadServlet	4	Alta	Alta	Baja	Alta
Openfire	1	Baja	Baja	Alta	Baja
CodificadorMD5	1	Baja	Baja	Alta	Baja
CodificadorBlowfish	1	Baja	Baja	Alta	Baja

Tabla 10: Cantidad de Dependencias por Clases.

Intervalos	Cantidad de Clases
0 dependencias	**0**
1 dependencias	**4**
> 1 dependencias	**1**

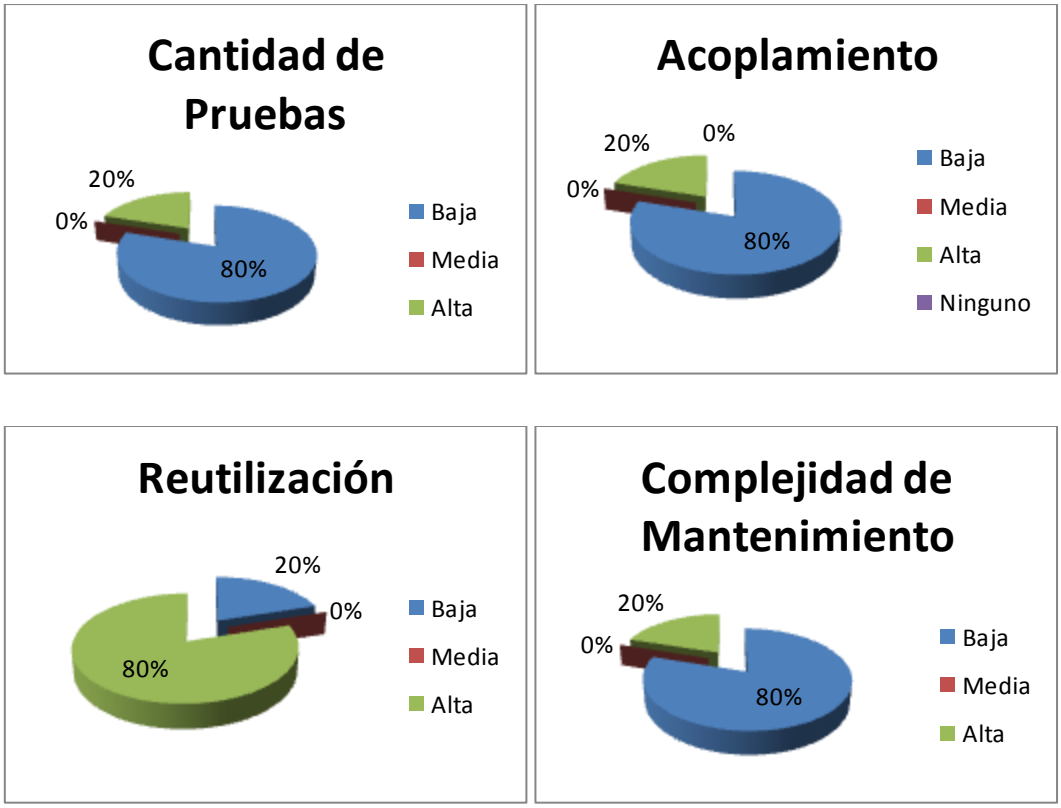
Teniendo en cuenta el número de dependencias de las clases se establece que un valor de relaciones pequeño es aquel que tiene un valor menor o igual que 1, un valor medio para las que están compendiadas entre uno y menores o incluyendo 3 dependencias y un valor de relaciones grande es aquel que es mayor que tres.

Tabla 11: Cantidad de relaciones. (23)

Umbral	Relaciones	Cantidad de Clases
Pocas	<=1	4
Medias	>1 y <= 3	0
Muchas	>3	1

Como se puede observar el 80% de las clases están clasificadas con pocas relaciones, y el 20% en el umbral de las clases con muchas relaciones, lo cual brinda

un resultado positivo según los atributos de calidad Complejidad del Mantenimiento, Cantidad de Pruebas, Reutilización y Acoplamiento.

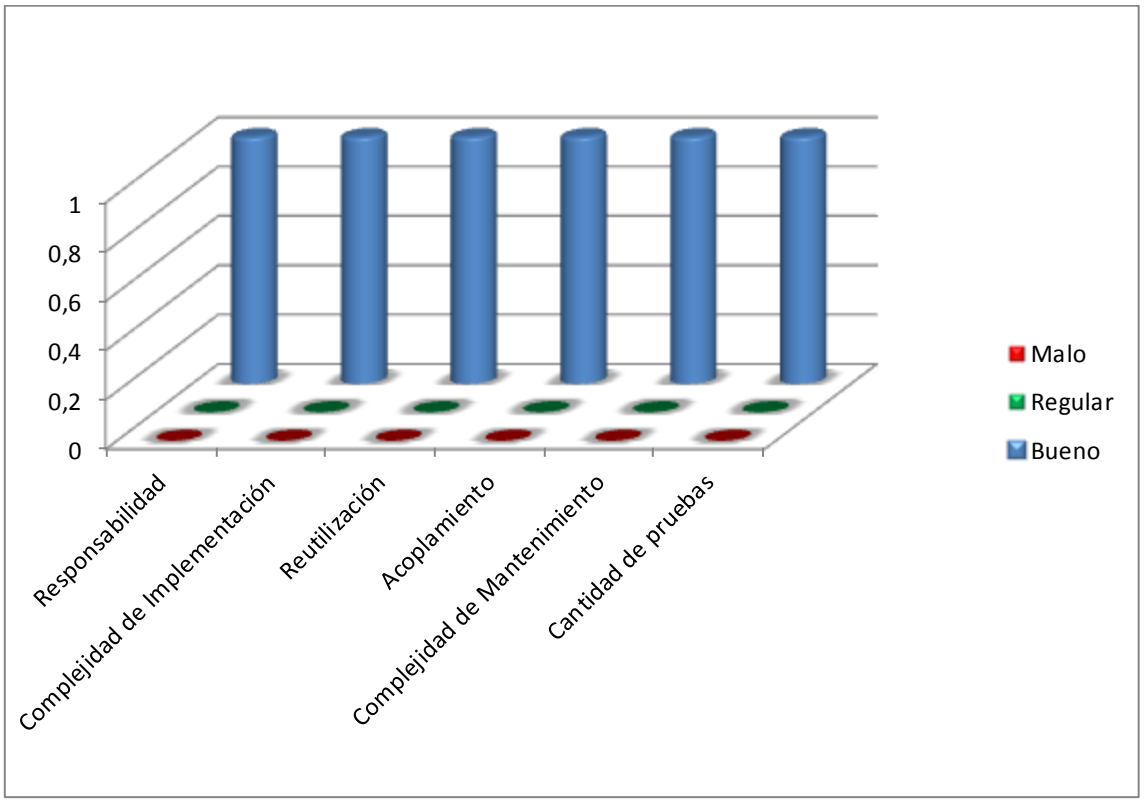


Gráfica 6: Cantidad de pruebas, acoplamiento, reutilización y complejidad de mantenimiento.

3.5. Matriz de cubrimiento de los atributos de calidad evaluados con las métricas propuestas.

La matriz de cubrimiento o matriz de inferencia de indicadores de calidad es el resumen de los resultados obtenidos al aplicar las métricas mencionadas en el epígrafe anterior. Esta matriz es una representación estructurada de los atributos de calidad para evaluar la calidad del diseño de la solución propuesta. (23) Esta matriz tomará valores de 0 o 1 según los criterios evaluativos de Bueno. Malo o Regular para cada uno de los atributos de calidad. Debido a las características del plugin desarrollado, el cual cuenta con 5 clases, si alguno de estos atributos sobrepasa el 50% de la totalidad de los demás criterios (Bajo, Medio, Alto, Ninguno) se tomará esta como predominante. Para tener una evaluación de Bueno el criterio predominante debe ser de Alto, o que la suma de los criterios Medio y Alto sea mayor que el de Bajo o Ninguno y mayor que el 50 % del total. Para obtener una evaluación de Regular deberá predominar el criterio Medio o la suma de Medio y Bajo mayor que Alto y

Ninguno y mayor que el 50% del total. Para obtener la evaluación de Malo tendrá que predominar el criterio Bajo o Ninguno o que la suma de estos dos sea mayor que el 50% del total de los criterios obtenidos en el atributo.



Gráfica 7: Matriz de cubrimiento para las métricas realizadas al diseño.

3.6. Pruebas de Software.

La prueba de software es un elemento esencial y crítico para la garantía de la calidad del software. El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado. La prueba se enfoca sobre la lógica interna del software y las funciones externas. Es considerado un proceso que tiene como objetivo la ejecución de un programa con la intención de descubrir un error. Un buen caso de prueba es aquel que tiene alta probabilidad de mostrar un error no descubierto hasta entonces.

3.6.1. Pruebas de caja blanca.

Permiten examinar la estructura interna del programa. Se diseñan casos de prueba para examinar la lógica del programa. Es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar casos de prueba que garanticen que:

- Se ejercitan todos los caminos independientes de cada módulo.

- Se ejercitan todas las decisiones lógicas.
- Se ejecutan todos los bucles.
- Se ejecutan las estructuras de datos internas.

```
protected void doGet(HttpServletRequest request,HttpServletResponse response) throws ServletException, IOException {

        PrintWriter out = response.getWriter();1

        if (!plugin.getAllowedIPs().isEmpty()) {2
            String ipAddress = request.getHeader("x-forwarded-for");4
            if (ipAddress == null) {5
                ipAddress = request.getHeader("X_FORWARDED_FOR");7
                if (ipAddress == null) {8
                    ipAddress = request.getHeader("X-Forward-For");9
                    if (ipAddress == null) {10
                        ipAddress = request.getRemoteAddr();11
                    }
                }
            }
            if (!plugin.getAllowedIPs().contains(ipAddress)) {6
                Log.warn("User service rejected service to IP address: "+ ipAddress);12
                replyError("RequestNotAuthorised", response, out);
                return;
            }
        }

        String username = request.getParameter("username");3
        String password = request.getParameter("password");
        String secret = request.getParameter("secret");
        String type = request.getParameter("type");
```

Figura 6: Código fuente de la funcionalidad doGet de la clase PluginSeguridadServlet.

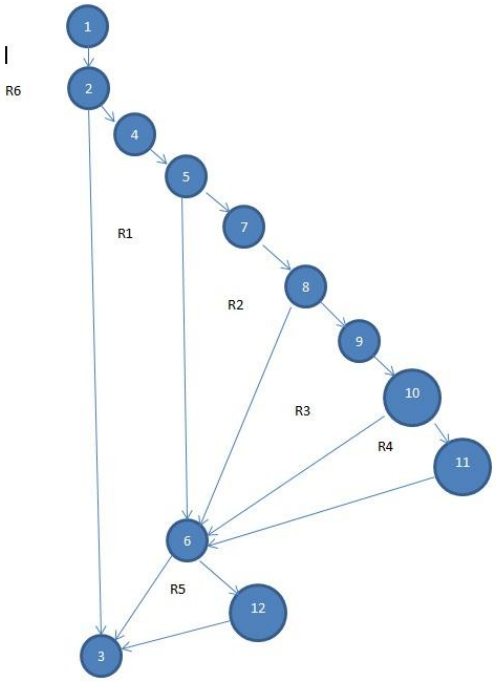


Figura 7: Muestra el grafo de flujo asociado a la funcionalidad.

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. (23)

Complejidad ciclomática de la funcionalidad doGet.

Existen tres formas fundamentales de calcular la complejidad:

1. V (G) = (A – N) + 2

V (G) = (16-12) + 2

V (G) =6

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

2. V (G) = P + 1

V (G) =5+1

V (G) =6

Donde P es el número de nodos predicado contenido en el grafo G.

3. V (G) = R

Donde "R" es la cantidad total de regiones, para cada formula "V (G)".

V (G) =6

Con el cálculo se obtuvo valor 6, seguidamente se representan los caminos básicos por los que puede transitar el flujo:

Camino #1: (1-2-3).

Camino #2: (1-2-4-5-6-3).

Camino #3: (1-2-4-5-7-8-6-3).

Camino #4: (1-2-4-5-7-8-6-12-3).

Camino #5: (1-2-4-5-7-8-9-10-6-3).

Camino #6: (1-2-4-5-7-8-9-10-6-12-3).

Derivación de casos de prueba de la funcionalidad doGet.

Luego de elaborar 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.

Casos de prueba para cada camino.

- Camino #1: (1-2-3).

Si los IPs permitidos están vacíos:

if (!plugin.getAllowedIPs().isEmpty())

- Camino #2: (1-2-4-5-6-3).

Si los IPs permitidos no están vacíos:

String ipAddress = request.getHeader("x-forwarded-for");

if (ipAddress == null) {

- Camino #3: (1-2-4-5-7-8-6-3).

Si las direcciones Ips son nulas:

if (ipAddress == null) {

ipAddress = request.getHeader("X_FORWARDED_FOR");

- Camino #4: (1-2-4-5-7-8-6-12-3).

Si las direcciones Ips son nulas:

if (ipAddress == null) {

ipAddress = request.getHeader("X-Forward-For");

- Camino #5: (1-2-4-5-7-8-9-10-6-3).

Si las direcciones Ips son nulas:

if (ipAddress == null) {

ipAddress = request.getRemoteAddr();

- Camino #6: (1-2-4-5-7-8-9-10-6-12-3).

Si los Ips permitidos no son Ips verdaderos:

if (!plugin.getAllowedIPs().contains(ipAddress)) {

Resultados esperados: Al ejecutar el procedimiento se lograron cargar los Ips permitidos.

Evaluación de los resultados obtenidos: Los resultados obtenidos de la realización del caso de prueba de caja blanca para el método doGet, fueron satisfactorios ya que se validó que cada camino fuera recorrido al menos una vez a partir de cada condición de ejecución y se obtuvieron los parámetros esperados.

Estas mismas pruebas fueron realizadas a las demás funcionalidades arrojando resultados satisfactorios.

3.6.2. Pruebas de Caja Negra.

Se denominan pruebas funcionales o Functional Testing, a las pruebas de software que tienen por objetivo probar que los sistemas desarrollados, cumplan con las funciones específicas para los cuales han sido creados. Es común que este tipo de pruebas sean desarrolladas por analistas de pruebas con apoyo de algunos usuarios

finales, esta etapa suele ser la última etapa de pruebas y al dar conformidad sobre esta el paso siguiente es el pase a producción. (30)

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.

A continuación se muestra uno de los casos de prueba que se definieron para cada uno de los requisitos del sistema, los demás se encuentran en la Tabla 22. Su objetivo principal es demostrar la reacción que corresponderá por parte del sistema luego de realizar alguna acción en el mismo. Para un mejor entendimiento de las respuestas o posibles funcionalidades que brindará el sistema, según la necesidad del usuario.

Tabla 12. Caso de Prueba: Activar o desactivar el plugin.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Activar o desactivar el plugin	Este permite la activación o desactivación del plugin para que el mismo realice las tareas necesarias.	EP 1.1: Se presiona el botón guardar cambios habiendo seleccionado la opción Activo.	– Se activa la ventana Propiedades del plugin de seguridad. – Se marca el RadioButton Activo en el cual se define que las solicitudes podrán ser procesadas. – Se guardan los cambios. – Se muestra un mensaje: Plugin de seguridad, propiedades editadas con éxito.
		EP 1.2: Se presiona el botón guardar cambios habiendo seleccionado la opción No Activo.	– Se activa la ventana Propiedades del plugin de seguridad. – Se marca el RadioButton No Activo en el cual se define que las solicitudes no podrán ser procesadas. – Se guardan los cambios. – Se muestra un mensaje: Plugin de seguridad, propiedades editadas con éxito.

El componente fue probado por el Departamento de Calidad del CEIGE donde se comprobó el buen funcionamiento del mismo y queda avalado por el acta de liberación expuesta en la Figura 8 y Figura 9.

3.7. Conclusiones Parciales.

En el capítulo fueron expuestos los artefactos generados en la etapa de implementación de la solución, tales como los estándares de codificación, el diagrama de componente y el diagrama de despliegue. Además se aplicaron métricas dirigidas a evaluar la calidad del diseño del software, basadas en diferentes atributos de calidad que avalaron sus resultados. Se realizaron las pruebas del camino básico como parte de las pruebas de caja blanca para garantizar el buen funcionamiento del código implementado, así como su eficiencia y solidez. Y por último para demostrar el cumplimiento de los requisitos definidos, se aplicaron pruebas de caja negra guiándose por los casos de prueba que se definieron para cada uno de los requisitos del sistema.

CONCLUSIONES.

Al desarrollar el plugin utilizando las herramientas y tecnologías propuestas, se logró la unión del Sistema integral de seguridad ACAXIA y el servidor de mensajería instantánea Openfire. Todo esto fue posible gracias al estudio realizado del estado del arte sobre las herramientas, tecnologías y técnicas aplicadas a nivel mundial que corroboran a solucionar la problemática, lo cual sentó las condiciones para su posterior análisis, diseño e implementación. Los resultados obtenidos fueron analizados mediante la validación de su diseño y una etapa de pruebas que reveló el cumplimiento de los requerimientos propuestos. El plugin permitirá gestionar los usuarios de Openfire desde ACAXIA y su posterior autenticación, lo que trae consigo el buen funcionamiento del componente Gestión de Notificaciones y el de CEDRUX como producto clave para el desarrollo de nuestro país.

RECOMENDACIONES.

- Una vez concluido el proceso de desarrollo se recomienda la realización de una versión posterior donde se incluyan otras funcionalidades que pueden servir de apoyo al componente de gestión de notificaciones.

- Se recomienda el uso del plugin para el intercambio entre el Servidor de Mensajería instantánea Openfire y el Sistema integral de seguridad ACAXIA.

ANEXOS.

Anexo 1:

Tabla 13. Especificación de Requisito Modificar Usuario.

Precondiciones	Existencia de usuarios insertados.	
Flujo de eventos		
Flujo básico		
1.	El sistema permitirá modificar cualquier usuario ya insertado.	
2.	Se registra en el sistema las modificaciones hechas del usuario.	
3.	El sistema actualiza los datos modificados del usuario.	
4.	El sistema valida los nuevos datos.	
5.	El sistema muestra un mensaje de confirmación.	
6.	Concluye el requisito.	
Pos-condiciones		
1.	Se modificó un nuevo usuario.	
Flujos alternativos		
Flujo alternativo 3.a Campos vacíos o erróneos.		
1	El sistema muestra un mensaje de error notificando que existen campos vacíos o erróneos e indica cuáles son estos.	
2	Continúa en el paso 3 del flujo básico.	
Pos-condiciones		
1	N/A	
Validaciones		
1	Se validan los datos según lo establecido en el Modelo conceptual <<Referencia al modelo conceptual en cuestión>>.	
Conceptos	**Usuario**	Utilizados internamente: ○ username ○ password ○ secret ○ type
Requisitos especiales	N/A	
Asuntos pendientes	Posibles mejoras al requisito.	

Tabla 14. Especificación de Requisito Eliminar Usuario.

Precondiciones	Existencia de usuarios insertados.
Flujo de eventos	
Flujo básico	
1.	El sistema permitirá eliminar cualquier usuario insertado anteriormente.
2.	El sistema dará la posibilidad de seleccionar el usuario que se quiere eliminar de una lista de usuarios.
3.	Se selecciona la opción eliminar usuario.
4.	El sistema valida los datos.
5.	El sistema muestra un mensaje de confirmación de la eliminación.
6.	Concluye el requisito.
Pos-condiciones	
1.	Se eliminó un usuario.
Flujos alternativos	
Flujo alternativo	
Pos-condiciones	

1	N/A
Validaciones	
1	Se validan los datos según lo establecido en el Modelo conceptual <<Referencia al modelo conceptual en cuestión>>.

Conceptos	**Usuario**	Utilizados internamente: o Username o Secret o Type
Requisitos especiales	N/A	
Asuntos pendientes	Posibles mejoras al requisito.	

Tabla 15. Especificación de Requisito Autenticar Usuario.

Precondiciones	Existencia del usuario en el servidor.
Flujo de eventos	
Flujo básico	
1.	El sistema permitirá autenticar los usuarios, mientras ya existan en el servidor.
2.	El sistema registra los datos que intervienen en el momento de autenticar el usuario: Datos de usuario username y password.
3.	El sistema valida los datos.
4.	El sistema muestra un mensaje de confirmación.
5.	Concluye el requisito.
Pos-condiciones	
1.	Se autenticaron los usuarios.

Flujos alternativos

Flujo alternativo 4.a Campos vacíos o erróneos.

1	El sistema muestra un mensaje de error, notificando que existen campos vacíos o erróneos e indica cuáles son estos.
2	Continúa en el paso 2 del flujo básico.

Flujos alternativos

Flujo alternativo 5.a.1 Usuario no registrado aún.

3	El sistema muestra un mensaje de error, notificando que el usuario no se encuentra registrado aún.
4	Continúa en el paso 2 del flujo básico.

Pos-condiciones

1	N/A
Validaciones	
1	N/A

Conceptos	**Usuario**	Utilizados internamente: Username password
Requisitos especiales	N/A	
Asuntos pendientes	Posibles mejoras al requisito.	

Tabla 16. Especificación de Requisito Adicionar IP.

Precondiciones	El servidor openfire debe estar iniciado. Debe existir conexión de red.	
Flujo de eventos		
Flujo básico		
1.	El sistema permitirá adicionar los IPs correspondientes.	
2.	Se registra en el sistema los nuevos IPs.	
3.	El sistema actualiza los IPs nuevos adicionados.	
4.	El sistema valida los datos.	
5.	El sistema muestra un mensaje de confirmación.	
6.	Concluye el requisito.	
Pos-condiciones		
1.	Se Adicionaron los IPs correspondientes.	
Flujos alternativos		
Flujo alternativo 6.a Campos vacíos o erróneos.		
1	El sistema muestra un mensaje notificando que existen campos vacíos o erróneos e indica cuáles son estos.	
2	Continúa en el paso 2 del flujo básico.	
Pos-condiciones		
2	N/A	
Validaciones		
2	Se validan los datos según lo establecido en el Modelo conceptual <<Referencia al modelo conceptual en cuestión>>.	
Conceptos	**IP**	Visibles en la interfaz: o Direcciones IPs permitidas.
Requisitos especiales	N/A	
Asuntos pendientes	Posibles mejoras al requisito.	

Tabla 17. Especificación de Requisito Modificar IP.

Precondiciones	Existencia de IPs adicionados al servidor.
Flujo de eventos	
Flujo básico	
1.	El sistema permitirá modificar los IPs ya agregados al servidor.
2.	Se registra en el sistema las modificaciones hechas.
3.	El sistema actualiza las modificaciones hechas.
4.	El sistema valida los datos.
5.	El sistema muestra un mensaje de confirmación.
6.	Concluye el requisito.

Pos-condiciones		
1.	Se modificó el IP.	
Flujos alternativos		
Flujo alternativo 7.a Campos vacíos o erróneos.		
3	El sistema muestra un mensaje notificando que existen campos vacíos o erróneos e indica cuáles son estos.	
4	Continúa en el paso 2 del flujo básico.	
Pos-condiciones		
1	N/A	
Validaciones		
1	N//A	
Conceptos	**IP**	Visibles en la interfaz: o Direcciones IPs permitidas
Requisitos especiales	N/A	
Asuntos pendientes	Posibles mejoras al requisito.	

Tabla 18. Especificación de Requisito Eliminar IP.

Precondiciones	Existencia de IPs adicionados anteriormente.	
Flujo de eventos		
Flujo básico		
1.	El sistema permitirá eliminar cualquier IP agregado anteriormente.	
2.	Se selecciona la opción eliminar documento.	
3.	El sistema muestra un mensaje de confirmación de la eliminación.	
4.	Concluye el requisito.	
Pos-condiciones		
2.	Se eliminó el IP.	
Flujos alternativos		
Flujo alternativo		
Pos-condiciones		
1	N/A	
Validaciones		
1	Se validan los datos según lo establecido en el Modelo conceptual <<Referencia al modelo conceptual en cuestión>>.	
Conceptos	**IP**	Visibles en la interfaz: Direcciones IPs permitidas
Requisitos especiales	N/A	
Asuntos pendientes	Posibles mejoras al requisito.	

Tabla 19. Especificación de Requisito Generar Clave secreta.

Precondiciones		El servidor openfire debe estar iniciado. Debe existir conexión de red.
Flujo de eventos		
Flujo básico		
1.	El sistema permitirá generar en la interfaz la clave secreta que ingrese el usuario.	
2.	Concluye el requisito.	
Pos-condiciones		
3.	Se generó la clave secreta.	
Flujos alternativos		
Flujo alternativo		
Pos-condiciones		
2	N/A	
Validaciones		
2	N/A	
Conceptos	**Clave Secreta**	Visibles en la interfaz: ○ Clave secreta
Requisitos especiales	N/A	
Asuntos pendientes	Posibles mejoras al requisito.	

Tabla 20. Especificación de Requisito Modificar Clave secreta.

Precondiciones	Existencia de claves secretas adicionadas anteriormente.
Flujo de eventos	
Flujo básico	
1.	El sistema permitirá modificar las claves secretas de cualquier usuario ya agregado al sistema.
2.	Se registra en el sistema la nueva clave secreta.
3.	El sistema valida los datos.
4.	Concluye el requisito.
Pos-condiciones	
1.	Se modificó una clave secreta al sistema.
Flujos alternativos	
Flujo alternativo 10.a Campos vacíos o erróneos.	
5	El sistema notifica que existen campos vacíos o erróneos e indica cuáles son estos.
6	Continúa en el paso 2 del flujo básico.
Pos-condiciones	
1	N/A
Validaciones	
1	N/A

Conceptos	**Documento**	Visibles en la interfaz: o Clave secreta
Requisitos especiales	N/A	
Asuntos pendientes	Posibles mejoras al requisito.	

Tabla 21. Especificación de Requisito Activar o desactivar plugin.

Precondiciones		El servidor openfire debe estar iniciado. Debe existir conexión de red.
Flujo de eventos		
Flujo básico Conformar expediente		
1.	El sistema permitirá Activar o desactivar peticiones.	
2.	El sistema dará la posibilidad de seleccionar Activo: “las peticiones serán procesadas”, No Activo: las solicitudes serán ignoradas”.	
3.	Se selecciona la opción deseada.	
4.	Concluye el requisito.	
Pos-condiciones		
1.	Se activó o desactivó plugin.	
Flujos alternativos		
Flujo alternativo		
Pos-condiciones		
1	N/A	
Validaciones		
1	Se validan los datos según lo establecido en el Modelo conceptual <<Referencia al modelo conceptual en cuestión>>.	
Conceptos	**Documento**	Visibles en la interfaz: Activo No Activo
Requisitos especiales	N/A	
Asuntos pendientes	Posibles mejoras al requisito.	

Anexo 2

Tabla 22. Caso de prueba: Generar y modificar clave secreta.

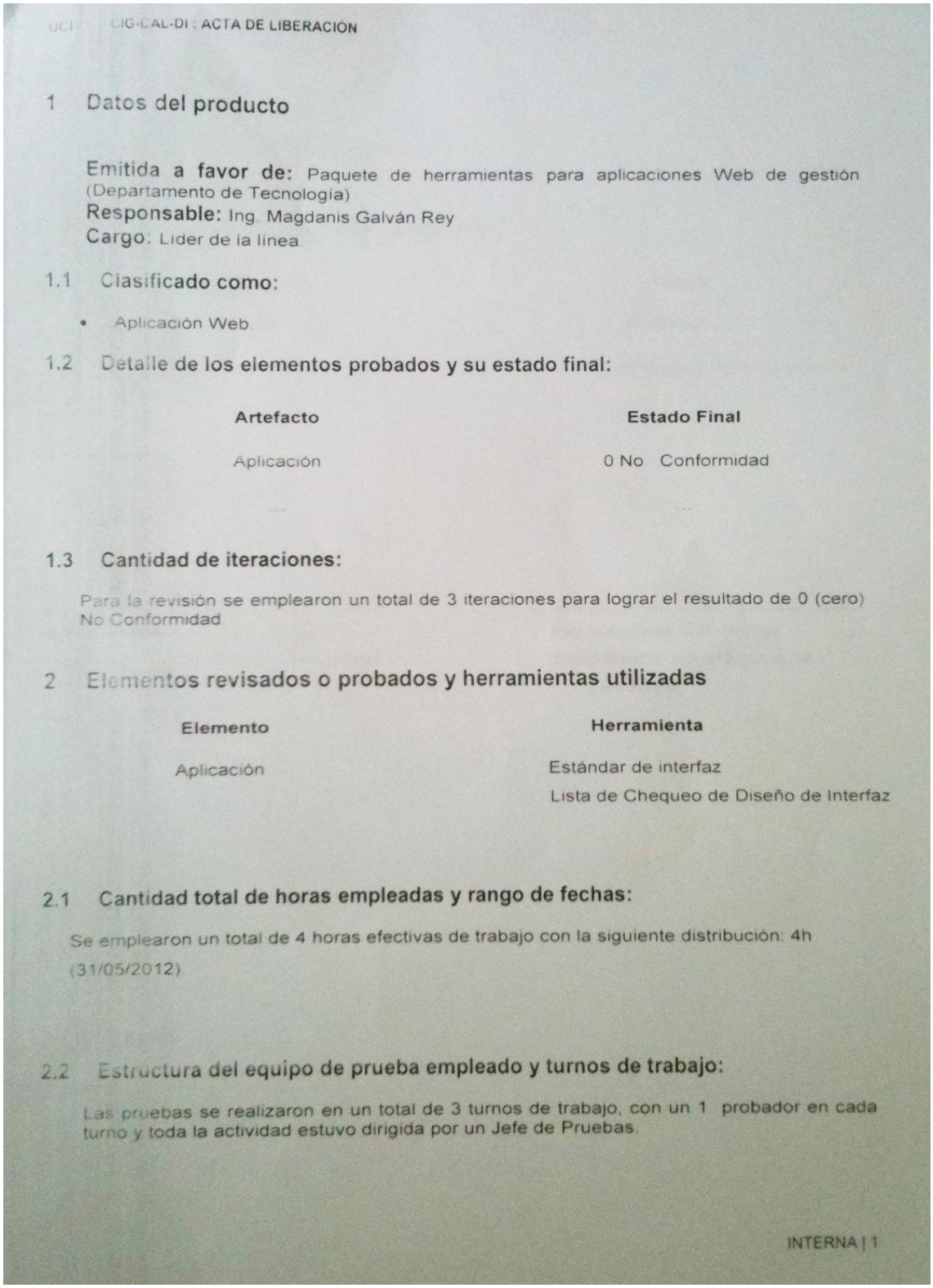
Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Generar y modificar clave secreta.	Este permite modificar o generar la clave secreta del plugin para que el mismo entienda las peticiones que se le harán.	EP 1.1: Se presiona el botón guardar cambios habiendo modificado la clave secreta. Esta debe contener de 1 a 8 caracteres.	– Se activa la ventana Propiedades del plugin de seguridad. – Se selecciona la clave secreta en la cual se define la misma. – Se guardan los cambios. – Se muestra un mensaje: Plugin de seguridad, propiedades editadas con éxito.
		EP 1.2: Se presiona el botón guardar cambios habiendo generado la clave secreta.	– Se activa la ventana Propiedades del plugin de seguridad. – Se guardan los cambios. – Se muestra un mensaje: Plugin de seguridad, propiedades editadas con éxito.
		EP 1.3: Se presiona el botón guardar cambios sin introducir la clave secreta.	– Se activa la ventana Propiedades del plugin de seguridad. – Se guardan los cambios. – Se muestra un mensaje: Plugin de seguridad, propiedades editadas con éxito.

Tabla 23. Caso de prueba: Gestionar IPs que hacen peticiones al servidor.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Gestionar IPs que hacen peticiones al servidor:	Este permite gestionar los IPs que pueden hacer peticiones al plugin y así poder agregar los datos que se necesitan.	EP 1.1: Se presiona el botón guardar cambios sin haber seleccionado ningún IP.	– Se activa la ventana Propiedades del plugin de seguridad. – Se selecciona el TextField en el cual se define que los IPs que serán procesados. – Se guardan los cambios. – Se muestra un mensaje: ¨Plugin de seguridad, propiedades editadas con éxito.¨
		EP 1.2: Se presiona el botón guardar cambios para adicionar uno o varios IPs.	– Se activa la ventana Propiedades del plugin de seguridad. – Se marca el TextField en el cual se define que los IPs que serán procesados. – Se adicionan los IPs o los rangos de IPs que se desea. – Se guardan los cambios. – Se muestra un mensaje: ¨Plugin de seguridad, propiedades editadas con éxito.¨
		EP 1.3: Se presiona el botón guardar cambios para modificar uno o varios IPs.	– Se activa la ventana Propiedades del plugin de seguridad. – Se marca el TextField en el cual están definidos los IPs que serán procesados. – Se modifican los IPs o los rangos de IPs que se desea. – Se guardan los cambios. – Se muestra un mensaje: ¨Plugin de

		seguridad, propiedades editadas con éxito.¨
	EP 1.4: Se presiona el botón guardar cambios para eliminar uno o varios IPs.	– Se activa la ventana Propiedades del plugin de seguridad. – Se marca el TextField en el cual están definidos los IPs que serán procesados. – Se eliminan los IPs o los rangos de IPs que se desea. – Se guardan los cambios. – Se muestra un mensaje: ¨Plugin de seguridad, propiedades editadas con éxito.¨

Anexo 3



UCI CIG-CAL-DI : ACTA DE LIBERACIÓN

1 Datos del producto

Emitida a favor de: Paquete de herramientas para aplicaciones Web de gestión (Departamento de Tecnología)
Responsable: Ing. Magdanis Galván Rey
Cargo: Líder de la línea

1.1 Clasificado como:

- Aplicación Web

1.2 Detalle de los elementos probados y su estado final:

Artefacto	Estado Final
Aplicación	0 No Conformidad

1.3 Cantidad de iteraciones:

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

2 Elementos revisados o probados y herramientas utilizadas

Elemento	Herramienta
Aplicación	Estándar de interfaz
	Lista de Chequeo de Diseño de Interfaz

2.1 Cantidad total de horas empleadas y rango de fechas:

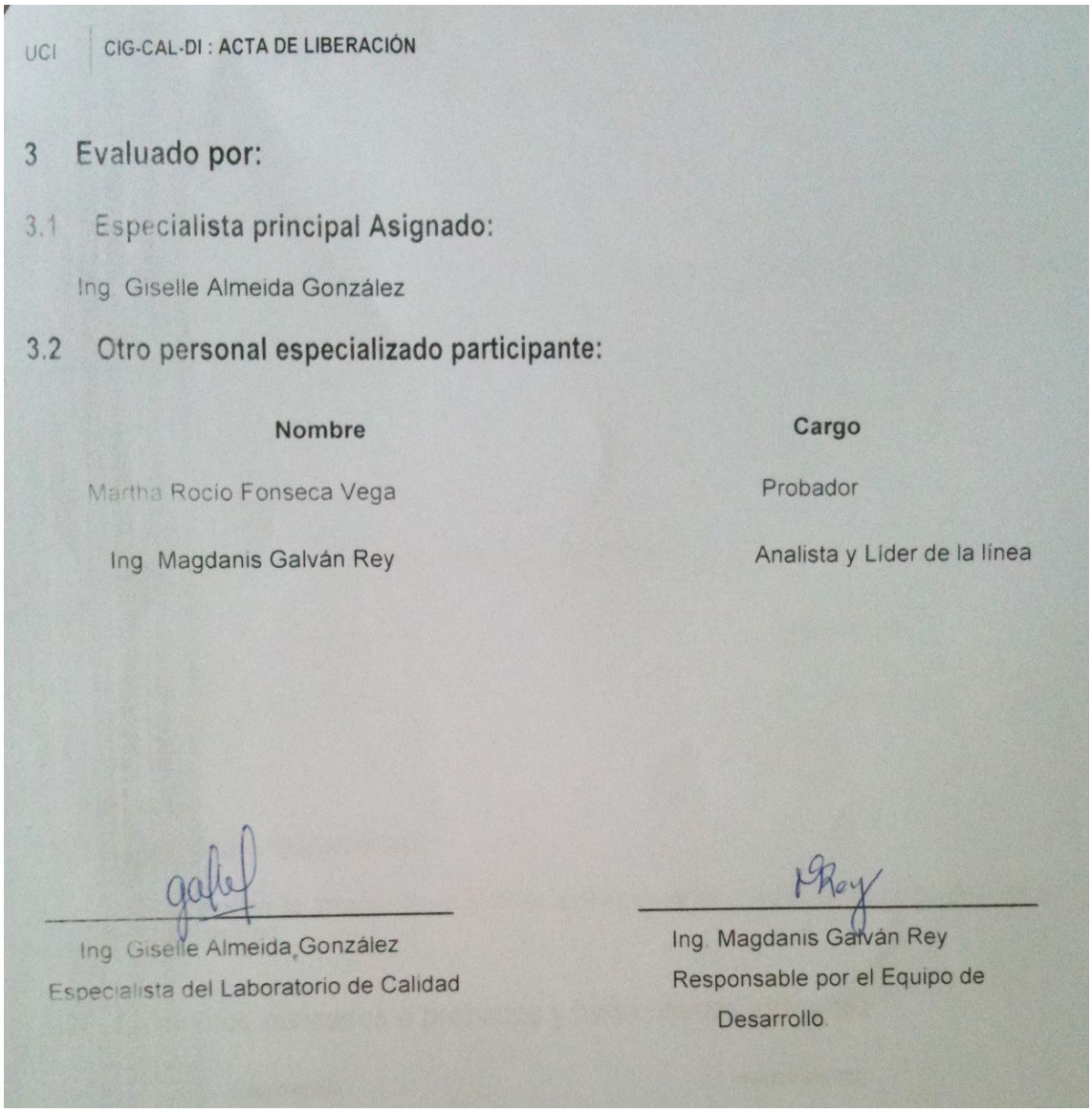
Se emplearon un total de 4 horas efectivas de trabajo con la siguiente distribución: 4h (31/05/2012)

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

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

INTERNA | 1

Figura 8. Acta de Liberación del Plugin de seguridad.



UCI | CIG-CAL-DI : ACTA DE LIBERACIÓN

3 Evaluado por:

3.1 Especialista principal Asignado:

Ing. Giselle Almeida González

3.2 Otro personal especializado participante:

Nombre	Cargo
Martha Rocío Fonseca Vega	Probador
Ing. Magdanis Galván Rey	Analista y Líder de la línea

Ing. Giselle Almeida González
Especialista del Laboratorio de Calidad

Ing. Magdanis Galván Rey
Responsable por el Equipo de Desarrollo

Figura 9. Acta de Liberación del Plugin de seguridad.

Bibliografía

1. MasterMagazine. Tendencias informáticas. [En línea] [Citado el: 04 10, 2012.] http://www.mastermagazine.info/termino/6891.php.

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

3. **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, 2006.

4. www.ecured.cu. *http://www.ecured.cu/index.php/XMPP.* [En línea] 02 21, 2012.

5. www.fing.edu.uy. *(Estudio del Open/Free (GNU/Linux) como plataforma de servicios de red en entornos empresariales.* [En línea] 12 18, 2011. http://www.fing.edu.uy/~asabigue/prgrado/2004eofgl/contenido/archivos/Anexo-III.pdf.

6. **Scavone, Federico.** Scribd. [En línea] [Citado el: 05 05, 2012.] http://es.scribd.com/doc/94978894/Xmpp.

7. **Conocimiento, OpenFire - Base de Conocimiento.** Scribd. [En línea] [Citado el: 05 05, 2012.] http://es.scribd.com/doc/90201703/OpenFire-Base-de-Conocimiento.

8. **PHP.** PHP. [En línea] [Citado el: 05 05, 2012.] http://php.net/manual/es/book.curl.php.

9. **Ivar Jacobson, Grady Booch, James Rumbaugh.** *El Proceso Unificado de Desarrollo de Softaware. 2005.*

10. **Pérez, Mileidys Sarduy.** *Propuesta de modelo de desarrollo de software tecnológico del Centro de Soluciones de Gestión. Habana.* 2009.

11. Curso de Java. [En línea] 02 28, 2012. http://tikal.cifn.unam.mx/~jsegura/LCGII/java3.htm.

12. **PHP., Grupo de documentación de.** *Manual de PHP.* 2001.

13. **Esser, Stefan.** *Secure Programming with the Zend-Framework. Dutch PHP Conference.* Amsterdam : s.n., 2009.

14. **Gómez, Oiner.** ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE SOFTWARE. La Habana, Cuba : s.n., 2012.

15. **Rojas, Prof. Jesús.** slideshare. [En línea] UNIVERSIDAD NACIONAL EXPERIMENTAL "FRANCISCO DE MIRANDA" VICE-RECTORADO ACADÉMICO ÁREA CIENCIAS DE LA EDUCACIÓN DEPARTAMENTO DE INFORMÁTICA Y TECNOLOGÍA EDUCATIVA SISTEMAS DE INFORMACIÓN, 02 10, 2009. http://www.slideshare.net/jesus25dite/servidor-xampp.

16. **Echevarría, Raúl Eduardo.** Slideshare. *Ide Eclipse, Breve Guía.* [En línea] [Citado el: 06 03, 2012.] http://www.slideshare.net/Benedeti/ide-eclipse-breve-gua-201399.

17. **Loughran, Steve, Hatcher, Erik.** *Ant in Action.* s.l. : Manning Publications, 12 de julio de 2007. 2nd edición.

18. **PostqreSQL.** [En línea] 02 18, 2011. http://www.postgresql.org/docs/8.3/static/release-8-3.html.

19. **Uml, Visual Paradigm For.** [En línea] 02 28, 2011. http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.

20. **CertSuperior.com.** [En línea] 02 23, 2012. http://www.certsuperior.com/CertificadosSeguridad.aspx.

21. Definición.org. [En línea] http://www.definicion.org/plug-in.

22. Definición.de. [En línea] [Citado el: Abril 23, 2012.] http://definicion.de/notificacion/.

23. **González, Oscar Oramas.** *Interoperabilidad del proceso inventario en el sistema Cedrux.* La Habana : s.n., 2011.

24. mitecnologico. [En línea] 02 27, 2012. http://www.mitecnologico.com/Main/EspecificacionesDeRequerimientos.

25. **Pupo, Yanisleydi Cañete.** Libro de Ayuda del Marco de Trabajo Sauxe, En su versión 2.0. La Habana : s.n., 2010.

26. Ciberaula. *Patrones de Diseño en aplicaciones Web.* [En línea] [Citado el: 04 20, 2012.] http://www.ciberaula.com/.

27. **Deacon, John.** [En línea] [Citado el: Abril 24, 2012.] Model-View-Controller (MVC) Architecture.

28. **Burbeck, Steve.** Applications Programming in Smalltalk-80(TM), How to use Model-View-Controller (MVC). [En línea] [Citado el: Abril 24, 2012.] http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html.

29. **Pressman, Roger S.** Ingenieria del Software. *Un Enfoque Practico.* s.l. : McGraw-Hill Companies, 2002.

30. **Oré, Ing. Alexander.** Calidad y software. [En línea] 5 29, 2012. http://www.calidadysoftware.com/testing/pruebas_funcionales.php.

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

32. **Palacio, Juan.** *Gestión de proyectos ágil: conceptos básicos.* 2006.

33. **Amaro Calderón, Sarah Dámaris, Valverde Rebaza y Jorge Carlos.** *Metodologías Ágiles.* Trujillo – Perú : s.n., 2007.

34. **Loureiro, Tania Teresa.** *Análisis y diseño de la solución informática para el subsistema de Caja, del sistema de gestión empresarial Cedrux. 2009.* 2009.

35. **GSINNOVA.** *Grupo de Soluciones. http://www.rational.com.ar/index.html .* 2011.

36. *visual-paradigm http://www.visual-paradigm.com/ visualparadigm.* 2010.

37. **Daniele, Ing.Marcela.** *El arte de modelar.* 2007.

38. Definición.org. [En línea] http://www.definicion.org/plug-in;http://www.definicion.org/plug-in.