Universidad de las Ciencias Informáticas

Facultad 2

Módulo de notificaciones del Sistema Integral de Análisis de Información

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Denis Henry Cruz Figueredo

Tutores: Ing. Lilian Saúco Altuna

Ing. Husseyn Despaigne Reyes

Co-tutor: MSc. Jorge Luis Olmedo Flores

"Año 55 de la Revolución"

Cuidad de la Habana, Junio 2013

Pensamiento

"… No lo sé" se ha convertido en "No lo sé todavía…".

Bill Gates.

DECLARACIÓN DE AUTORÍA

Declaro ser el único autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma.

Para que así conste firmo la presente a los _26_ días del mes de _mayo_ del año _2013_.

Denis Henry Cruz Figueredo

_____

Autor

Ing. Lilian Saúco Altuna

_____

Tutora

Ing. Husseyn Despaigne Reyes

_____

Tutor

MsC. Jorge Luis Olmedo Flores

_____

Co-tutor

DATOS DE CONTACTO

Ing. Lilian Saúco Altuna, lsauco@uci.cu

Ingeniera en Ciencias Informáticas, graduada en la Universidad de Ciencias Informáticas en el año 2008. Trabaja en la Facultad 2 de dicho centro de enseñanza superior con categoría docente de Instructor. Pertenece al Centro de Telemática, específicamente al proyecto Sistema Integral de Análisis de Información, en el cual ocupa el rol de líder.

Ing. Husseyn Despaigne Reyes, hdespaigne@uci.cu

Ingeniero en Ciencias Informáticas, graduado en la Universidad de Ciencias Informáticas en el año 2011. Trabaja en la Facultad 2 de dicho centro de enseñanza superior. Pertenece al Centro de Telemática, específicamente al proyecto Sistema Integral de Análisis de Información, en el cual ocupa los roles de arquitecto y jefe de desarrollo.

AGRADECIMIENTOS

A mis padres Milsi y Osvaldo y a mi hermana Diana, por estar siempre ahí cuando los necesito. A mis abuelos, a mi tía Anita y a Cheo, a mis primos Danay, Dany, Pucho, Puni, El Pombo y los demás… por confiar en mí, por el apoyo, por la amistad, por ser los míos.

A mis tutores, porque más que tutores son mis compañeros de tesis… mis amigos.

A mi familia del proyecto SIAI. Lilian, Husseyn, Martica, Erik, Norbelis, Sandy, Yaili, Dany, Pedro, Arian, Jova, el chami con sus pensamientos, Rogelio, a Olmedo, Alberto, a todos…. Porque me han hecho superarme, estudiar, crecer…

A mis amigos de los grupos en los que he estado en estos años de universidad, especialmente a los de primer año, que fueron los mejores. A los "sobrevivientes" y a los que no también.

A mis amigos del apto. Michel, Dalien (la masacre), Alejandro, Luis. Porque son los mejores.

A los colegas de edificio 15 y del laboratorio 305.

A Yoanni y Ernesto.

A mis maestros de la primaria, Leonardo, Damaris, Zenia, y Ramón que todavía se preocupan por mí y están pendientes de mi carrera.

A los que no pongo en el documento porque son muchos nombres pero que también tengo presentes.

A la UCI….

DEDICATORIA

A mis padres.

A mis amigos.

A ustedes lectores.

RESUMEN

Con el avance de las tecnologías de la informática y las comunicaciones se ha producido un incremento de soluciones informáticas que necesitan de la interacción entre aplicaciones mediante el intercambio de mensajes. Por lo general, para que la comunicación se realice, se requiere que el remitente y el destinatario estén disponibles, lo que trae consigo una necesidad de sincronización, que en dependencia de los requerimientos de la solución pudiera atentar contra su funcionamiento si una de las dos aplicaciones no está disponible en ese instante. El presente trabajo muestra cómo la creación de una herramienta que actúa como intermediario en las comunicaciones entre aplicaciones contribuye a mejorar el funcionamiento de soluciones con estas características y mejora la interoperabilidad entre aplicaciones ya que unifica el mecanismo de comunicación. El módulo de notificaciones del Sistema Integral de Análisis de Información (SIAINTFS) consiste en una aplicación servidora que posibilita el reenvío de mensajes desde un cliente hasta otro, haciendo esta comunicación asíncrona. Los servicios de recepción y entrega de los mensajes se realizan a través de servicios web, pero se incluye soporte para *plugins* mediante los cuales pueden agregarse nuevas formas de entrega de mensajes. La aplicación está desarrollada en lenguaje Java con encriptación de los mensajes usando el algoritmo AES[1]. Para el almacenamiento de los mensajes se utiliza el servidor de colas de mensajes HornetQ y se realizan las conexiones usando el API[2] JMS[3].

Palabras claves:
comunicación entre aplicaciones, fiabilidad en el intercambio de mensajes, interoperabilidad, notificación, reenvío de mensajes.

[1] AES: del inglés “Advanced Encryption Standard”, en español “Estándar de encriptación avanzado”.
[2] API: del inglés “Application programming Interface”, en español “Interfaz para programación de aplicaciones”.
[3] JMS: del inglés “Java Message Service”, en español “Servicio de mensajes de Java”.

ÍNDICE

INTRODUCCIÓN

Luego del surgimiento del teléfono en el siglo XIX, debido a la amplia aceptación por parte del público y por su gran utilidad, surgieron empresas que hicieron su negocio brindando el servicio de telefonía a otras compañías y a la población en general. Estas empresas dedican tiempo y dinero a mejorar los diseños, funcionalidades y la calidad de los servicios. Los primeros teléfonos comercializados fueron de tecnología conocida hoy como fija, en la que los mismos están conectados mediante cables a una central telefónica que se encarga de comunicarlos. En la segunda mitad del siglo XX surgen los teléfonos celulares y la tecnología digital que abarcó también a los teléfonos fijos, y mejoró la calidad de las comunicaciones.

Con el avance de las tecnologías informáticas y su facilidad para la automatización, las conexiones entre teléfonos se realizan mediante aplicaciones informáticas que controlan el proceso. Tal es la utilidad de los teléfonos que hoy en día son muy comunes en la población mundial. "*A finales de 2013 es posible que haya tantos o más teléfonos que personas en el mundo"* (1). Estas condiciones son propicias para que muchas personas utilicen o intenten utilizar el servicio de manera ilegal, incurriendo en hechos delictivos que reportan pérdidas económicas para los proveedores de servicios e influyen negativamente en la calidad del servicio brindado.

La Empresa de Telecomunicaciones de Cuba S.A (ETECSA) es la encargada de proveer el servicio de telefonía en el país. Esta entidad ha visto un rápido crecimiento en el servicio que presta a la población cubana. Con este crecimiento también se ha producido un incremento considerable en los delitos telefónicos que reporta pérdidas monetarias, afecta la calidad y la imagen de la empresa frente a sus abonados. Con el objetivo de disminuir las acciones ilegales la empresa cuenta con un grupo de analistas encargados de investigar los posibles casos de fraude y llevar los responsables ante la ley, pero el elevado número de fraudes o a veces pistas falsas hace que el trabajo de estas personas sea demasiado lento en la detección e investigación, lo que atenta contra la calidad de su trabajo y la eficacia de la institución. Por esta razón ETECSA, en convenio con la Universidad de la Ciencias Informáticas (UCI) se encuentra desarrollando un software que ayudará a los analistas de fraude en su trabajo diario. Este software lleva por nombre Sistema Integral de Análisis de Información (SIAI).

En el marco del SIAI, donde coexisten varias aplicaciones y un gran volumen de datos, es un hecho la necesidad de que dichas aplicaciones intercambien información entre ellas o notifiquen a usuarios sobre determinados estados o resultados de sus tareas. Estas notificaciones son el punto de inicio o fin de otros

procesos que podrían ser cruciales para la detección de fraudes o incidentes telefónicos, lo que convierte el envío de notificaciones en un proceso muy delicado y de repercusión muy alta dentro del sistema. Actualmente estas situaciones no pueden ser notificadas automáticamente al personal involucrado y las mismas implican, en su mayoría, desencadenar acciones de continuidad importantes para el proceso investigativo. Tal escenario conlleva demoras en la clasificación de determinados comportamientos como anómalos para su inmediato seguimiento, provocando que pueda aumentar el número de fraudes y por consiguiente pérdidas económicas al país.

Una vez enviadas las notificaciones, existe el riesgo de que los mensajes no lleguen a su destinatario por problemas como: fallos en la conexión, interrupción del servicio eléctrico, fallas de software o hardware, que el usuario al que fue enviado no se encuentre *online*[4], etc. Si estos mensajes se pierden se ve afectado o se detiene un flujo de acciones necesarias para la entidad que podría traducirse en pérdidas de dinero, por lo que surge el problema de garantizar el correcto envío y recepción de estos mensajes o notificaciones de forma *fiable*[5].

Partiendo de la problemática descrita anteriormente se plantea como **problema a resolver:** ¿Cómo garantizar el envío de notificaciones a los usuarios y aplicaciones que interactúan con el SIAI, asegurando su fiabilidad, para el apoyo a las investigaciones realizadas por los analistas de fraude de ETECSA?

Como propuesta de solución al problema se plantea la automatización del proceso de envío de notificaciones a los usuarios y aplicaciones que interactúan con el SIAI, teniendo como **objeto de estudio:** el proceso de intercambio de mensajes entre aplicaciones informáticas y el envío de notificaciones a usuarios**,** enmarcado en el **campo de acción:** el proceso de intercambio de mensajes entre aplicaciones informáticas y el envío de notificaciones a usuarios por medio del correo electrónico, vinculados a la detección de fraudes telefónicos en ETECSA. Se tiene como **objetivo general:** Desarrollar un módulo que garantice el envío de notificaciones a los usuarios y aplicaciones que interactúan con el SIAI, asegurando su fiabilidad.

Las **tareas a desarrollar** para dar cumplimiento al objetivo general son las siguientes:

- ✓ Elaboración de la fundamentación teórica del trabajo.

[4] Online: Término usado para referirse a un usuario que interactúa con una aplicación en un instante de tiempo determinado.
[5] Fiable: Término usado para referirse al proceso de envío de notificaciones garantizando la entrega del mensaje a su destinatario.

- ✓ Estudio de mecanismos para garantizar la interoperabilidad entre sistemas informáticos para conocer su estado del arte y tendencias a nivel nacional e internacional.
- ✓ Selección y estudio de herramientas a utilizar para desarrollar el módulo de notificaciones.
- ✓ Identificación de los procesos y reglas del negocio.
- ✓ Elaboración del modelo de negocio de los procesos que tienen lugar en el organismo en cuestión, para la comprensión del flujo normal para realizar el envío de notificaciones entre usuarios y aplicaciones que interactúan con el SIAI.
- ✓ Descripción de los procesos representados en el modelo de negocio.
- ✓ Definición de requerimientos funcionales y no funcionales de la solución de software para realizar el envío de notificaciones entre usuarios y aplicaciones que interactúan con el SIAI.
- ✓ Descripción de los requerimientos funcionales y no funcionales de la solución de software.
- ✓ Elaboración del modelo de diseño de la solución de software.
- ✓ Implementación de un módulo de notificaciones modular y escalable que permita enviar avisos a usuarios y aplicaciones.
- ✓ Realización de pruebas de funcionalidad a la solución de software.
- ✓ Realización de los manuales de usuarios necesarios para guiar a los futuros beneficiarios del módulo de notificaciones en un correcto uso y explotación de las diferentes funcionalidades del mismo.
- ✓ Elaboración de una adecuada documentación de la solución en todas sus etapas (estudio del estado del arte, modelamiento del negocio, diseño, implementación y prueba).

Métodos científicos utilizados:

Método teórico "Modelación"**:** Durante la investigación y desarrollo serán modelados los diferentes estados en que podría caer la aplicación en su ciclo de vida para tener una idea lo más clara posible de las repercusiones que podría tener cada uno de estos estados.

Método teórico "Histórico-lógico": Permite estudiar el historial y evolución de los sistemas de notificación, hacer un estudio del arte y compararlas.

Método empírico “Experimento”: En la etapa de desarrollo se utilizará la experimentación con las tecnologías y herramientas actuales para comprobar su validez y correcto desempeño en la solución del problema planteado.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En el presente capítulo se analizarán conceptos fundamentales relacionados con los sistemas de notificaciones, las tecnologías y tendencias actuales en el uso de los mismos, así como sus características, el estudio de conceptos elementales para su comprensión, ventajas y desventajas; con el objetivo de poder contar con elementos sólidos que permitan la selección de una o varias alternativas adecuadas para usar en la realización del Módulo de notificaciones del SIAI.

1.2. Estado del arte

A continuación se muestran algunos sistemas desarrollados que de alguna manera ejecutan el envío de notificaciones a usuarios y aplicaciones.

1.2.1. Sistemas de notificaciones a nivel mundial

NetSupport Notify

Es una solución propietaria desarrollada especialmente enviar notificaciones y alertas instantáneas y fiables a escritorios Windows, Mac y Linux desde un ordenador con sistema operativo Windows o un dispositivo Apple. Su filosofía es ofrecer una solución sencilla y eficaz para que los administradores de una organización puedan enviar notificaciones importantes al personal sin importar el diseño de la red, además de ofrecer acuses de recibo de entrega y confirmación de los mensajes (2).

Está compuesta por una aplicación servidora y todos los clientes que sean necesarios. Los interesados en recibir los mensajes deben instalar las aplicaciones clientes (agentes de notificación) que estarán conectadas al servidor para recibir las notificaciones y mostrarlas directamente cuando sean enviadas. Permite establecer niveles de prioridad para los mensajes como son; crítico, aviso, técnico o noticias y personalizar estos con el logo de la empresa y sonidos asociados para cada caso. Además permite que los mensajes lleven una solicitud para confirmar su recepción. No posibilita que las notificaciones sean enviadas desde una aplicación externa ni que sean enviadas a otra aplicación que no sean los agentes de notificación que provee la propia herramienta, lo que limita su utilización en otros entornos y bajo otras condiciones como es el caso del SIAI (2).

C2DM – Android Cloud to Device Messaging:

Es un servicio brindado por Google[6] que permite a los desarrolladores enviar notificaciones a sus aplicaciones en terminales móviles Android[7]. Es un servicio gratuito mientras no se sobrepasen los límites de volumen de tráfico de datos establecidos por Google. Al usar este servicio las notificaciones se realizan a través de los servidores de Google, lo que podría comprometer la seguridad de los datos del SIAI, ya que se involucraría a un tercero en la comunicación. Al ser un servicio específico, si en el futuro se desea enviar notificaciones de algún tipo que no sea soportado, los desarrolladores del SIAI tendrían que buscar otra vía para hacerlo, lo que obligaría a cambiar la estructura del sistema de notificaciones o incluir otro servicio o herramienta paralelamente para satisfacer el nuevo requerimiento, continuando con la misma problemática que se intenta resolver con esta investigación (3).

DELL AlertFind(TM):

"Es una plataforma de notificación empresarial que provee entrega automatizada de mensajes y seguimiento de respuestas, utiliza múltiples canales de comunicación como son:

- ✓ *Llamadas de voz*
- ✓ *Correo electrónico*
- ✓ *Mensajes SMS*
- ✓ *Buscador de personas*
- ✓ *Fax*

Es un servicio brindado por Dell (TM) y entregado como un servicio basado en suscripción, por lo que no requiere ninguna inversión en hardware o software por parte de los clientes" (4).

Brinda una API basada en servicios web para integrarse con otras aplicaciones y permitir el envío automatizado de notificaciones. Permite definir qué usuarios pueden enviar y/o recibir mensajes a/desde otros usuarios.

[6] Google: Compañía estadounidense especializada en búsquedas por internet y publicidad.
[7] Android: Sistema Operativo basado en GNU/Linux, para dispositivos móviles.

La entrega de los mensajes se realiza en dependencia de los "Créditos de mensajes" mensuales contratados por el cliente, y por el destino de estos mensajes ya que varía la cantidad de créditos consumidos en dependencia del país, región de destino y el canal de comunicación utilizado (4).

A pesar de ser muy completo y contar con el respaldo de una empresa de prestigio internacional como Dell (TM), no constituye una opción viable para ETECSA ya que utilizar Dell AlertFind (TM) implicaría que cada mensaje intercambiado por las aplicaciones o enviado a un usuario sería transmitido a través de una entidad externa, algo que va en contra de las políticas de la empresa. Otro punto en contra es el hecho de que los mensajes enviados a Cuba consumen una gran cantidad de créditos lo que constituiría un gasto de dinero importante para ETECSA, además de la inseguridad de saber si en el futuro se podría continuar utilizando el servicio debido a las leyes del gobierno de EE.UU con respecto a Cuba.

No obstante el estudio de algunas de las características de este sistema aportó información relevante que contribuyó a concebir y diseñar cómo sería el funcionamiento del servidor de notificaciones desarrollado en este trabajo. De AlertFind (TM) se tomó la idea de abstraer a los clientes del servicio de construir distintas estructuras en dependencia del tipo de mensaje; estos solo serán responsables de enviar un mensaje en un único formato y es el servidor el que los diferencia según el canal utilizado para enviarlo. Además se siguió el principio de crear un API de alto nivel para que dichos clientes no interactúen directamente con el servidor, abstrayéndolos de los detalles de bajo nivel en la implementación de la vía de comunicación. Por otra parte, AlertFind (TM) utiliza un sistema de administración donde pueden configurarse usuarios, grupos y canales de comunicación, pudiendo definirse una serie de políticas de autorización para el consumo de los servicios que brinda este sistema. Este último aspecto también fue de utilidad para la creación del servidor de notificaciones del SIAI. En la Figura 1 se ilustra el funcionamiento básico de AlertFind (TM).

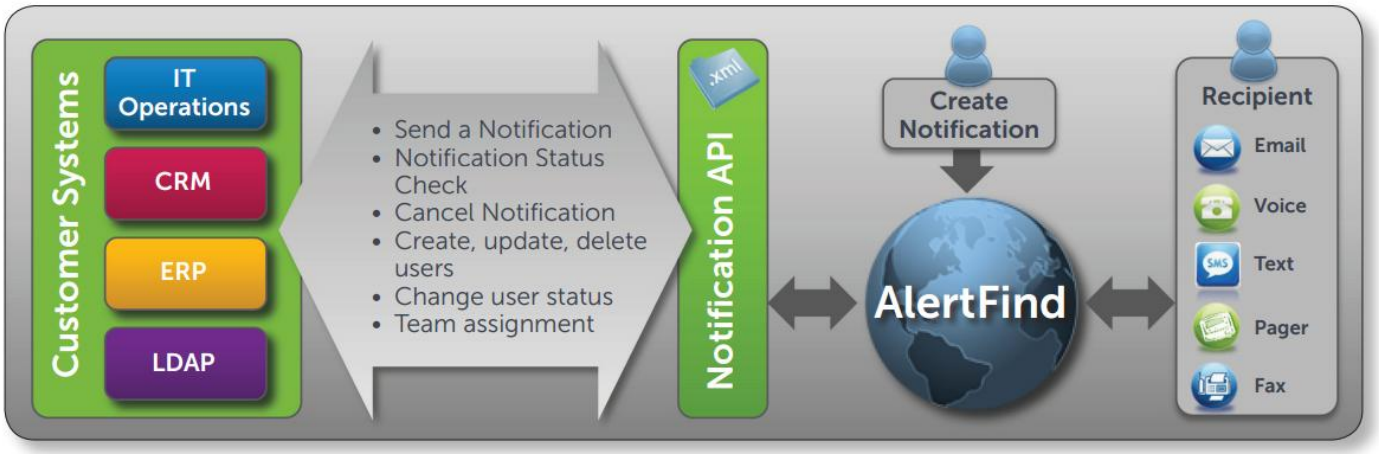

Figura 1. Funcionamiento de AlertFind (TM) (4).

1.2.2. Sistemas de notificaciones en la UCI.

En la UCI con el paso del tiempo se han desarrollado algunas aplicaciones que entre sus módulos cuentan con la posibilidad de enviar correos a usuarios e incorporan otras funcionalidades relacionadas con el envío de notificaciones. Como ejemplos de estas están:

Sistema de notificación de eventos:

Este es un sistema que forma parte del **Sistema de video vigilancia** desarrollado por la UCI para el MINTUR[8]. Este sistema desarrolla el envío de notificaciones por diferentes vías como son el correo electrónico, SMS[9], Beeper[10] y Socket[11]. Está desarrollado de forma tal que es posible agregarle funcionalidades mediante *plugins* pero no garantiza la fiabilidad en el envío de las notificaciones, por lo que ante una falla en la comunicación los mensajes se pierden, lo que constituye el centro del problema a resolver en el presente trabajo (5).

Sistema de notificaciones de RHODA[12] 2.1:

[8]MINTUR: Ministerio del turismo.
[9]SMS: Del inglés Short message service, en español Servicio de mensajes cortos.
[10]Beeper: Dispositivo de comunicación que permite la recepción de mensajes escritos y emite un pitido (beep) cuando se recibe el mensaje. El mensaje puede ser enviado a través de internet o desde otro Beeper.
[11]Socket: Combinación de una dirección IP (la dirección numérica única de cuatro o seis partes que identifica a un ordenador particular en Internet) y un número de puerto (numero de 16 bits que sirve como punto de conexión con una aplicación).
[12]RHODA: Repositorio de Objetos de aprendizaje.

Sistema para mejorar las notificaciones del repositorio RHODA, permitiendo enviar correos y mantener actualizados a los usuarios interesados sin necesidad de que visiten el repositorio mediante el uso de RSS[13].En esta aplicación a los usuarios definidos en el sistema se le envían notificaciones ante algún cambio en el repositorio pero no se garantiza la fiabilidad de la comunicación ni la entrega del mensaje si hay fallas en la comunicación. El sistema está hecho para el escenario específico del Repositorio de objetos de aprendizaje lo que dificulta su uso en otros escenarios y no tiene soporte para agregar nuevas funcionalidades sin implementarlas directamente en el código original (6).

Componente de notificación y mensajería para productos del área temática sistemas de apoyo a la salud:

Componente que permite la notificación y el intercambio de mensajes entre usuarios y sistemas de apoyo a la salud. Logra integrar los servicios de mensajería síncrona y asíncrona, modelando servicios web XML[14], posibilitando a sistemas externos el envío de notificaciones a los usuarios que interactúan con la información gestionada por estos. Fue obtenido un prototipo no funcional refinado que permite una correcta comprensión de los requerimientos de software especificados, elemento significativo para el proceso de implementación. En dicho trabajo no se llega a la implementación de la herramienta y según la descripción no se valora la posibilidad de hacer que la entrega de las notificaciones sea fiable ni la posibilidad de incluir nuevas funcionalidades sin modificar el código original de la herramienta (7).

1.3. Conceptos asociados al dominio del problema.

1.3.1. Servicio de mensajería.

Un servicio de mensajería proporciona un sistema garantizado de envío y entrega de mensajes, constituyendo un medio sólido para llevar a cabo la mayor parte de los procesos de una empresa. En este contexto, un mensaje es una unidad de datos enviada entre dos equipos que puede ser muy sencillo y constar de una simple cadena de texto, o más complejo e incluir, por ejemplo, objetos incrustados. Los mensajes se envían a colas, las cuales no son más que un contenedor que alberga mensajes mientras están en tránsito. El principal objetivo de una cola es proporcionar enrutamiento y garantizar la entrega de los mensajes; si el destinatario no está disponible en el momento de enviarse un mensaje, la cola lo

[13]RSS: del inglés "Really Simple Syndication", en español "Sindicación Realmente Simple".
[14] XML: del inglés "Extensible Markup Languaje", en español "Lenguaje de marcado extensible".

guarda hasta que pueda entregarse correctamente. El administrador o servidor de colas actúa como intermediario en la transmisión de un mensaje desde su origen hasta su destino, para lo cual se utiliza un API de mensajería como es el caso de JMS (8).

Para el desarrollo del presente trabajo se decidió emplear un servicio de mensajería para la transferencia y persistencia de las notificaciones, permitiendo utilizar un conjunto de herramientas y tecnologías con un marcado prestigio en este tipo de escenarios.

1.3.2. Java Message Service.

Es una API de mensajería de Java que permite a las aplicaciones crear, enviar, leer y recibir mensajes. Define un conjunto de interfaces que guían el trabajo de los desarrolladores para crear aplicaciones de calidad siguiendo buenas prácticas. Cuenta con el mecanismo de transacciones que permite tratar a un grupo de operaciones como un todo, algo muy útil cuando se tiene que llevar a cabo un proceso que incluye varias acciones y solamente tiene sentido dar por terminado el proceso si cada acción se ejecutó como estaba previsto (9).

Estas transacciones son especialmente útiles en el servidor de notificaciones cuando se extraen los mensajes de las colas para ser enviados a sus destinos ya que si no es posible enviar el mensaje en ese momento el proceso no está completo y es necesario retornar el mensaje a su cola de origen.

1.3.3. Plugin.

Es un componente que se añade a una aplicación para agregarle funcionalidades a la misma. Los *plugins* deben cumplir con una o varias interfaces que controlarán la comunicación de estos con el programa principal que hará uso de ellos (10). La aplicación principal proporciona servicios que el *plugin* puede utilizar. Los *plugin* dependen de los servicios prestados por la aplicación de acogida y no suelen funcionar por sí mismos. Por el contrario, la aplicación principal funciona independientemente de ellos, lo que permite a los usuarios finales añadir y actualizar los *plugin* de forma estática o dinámica sin necesidad de hacer cambios a la misma.

Como se detalla más adelante en el capítulo 3, para realizar el servidor de notificaciones se utilizó una arquitectura basada en *plugins,* que permite que le sean agregadas funcionalidades y garantizar de esta forma, que pueda extenderse su utilidad y brindar nuevas vías de notificación en el futuro sin necesidad de modificar el código interno del mismo.

1.3.4. Algoritmo de encriptación RSA[15].

Este es un algoritmo de encriptación de clave pública (PKI[16]) que fue seleccionado para su utilización debido a que es uno de los más antiguos y seguros que existen hoy en día, lo que permitirá a la aplicación utilizar un mecanismo de seguridad altamente probado y confiable sin arriesgar el soporte por parte de los diferentes lenguajes de programación en que pudieran desarrollarse las aplicaciones clientes. Dentro de los algoritmos de clave pública, RSA es una de las opciones más rápidas en cuanto a tiempo de encriptación y desencriptación. El hecho de ser un algoritmo de clave pública permite que sea utilizado para garantizar el intercambio seguro de una clave simétrica entre aplicaciones de manera automatizada por un canal que pudiera no ser seguro.

1.3.5. Algoritmo de encriptación AES[17].

"Es un algoritmo de encriptación simétrico aceptado como estándar en el año 2002" (11). Es utilizado mundialmente en todo tipo de aplicaciones y escenarios. Principalmente por su alto grado de seguridad y velocidad de encriptación/decriptación. *"Es un algoritmo calificado por el gobierno de los EE.UU como suficientemente seguro para proteger información clasificada hasta el nivel de Alto secreto que es el de mayor seguridad"* (12). Fue seleccionado para su utilización debido a las características expuestas anteriormente y al hecho de poder encriptar datos de cualquier tamaño, lo que junto al uso de RSA brindará a la aplicación un alto grado de seguridad en el envío de los datos manejados a través de la red.

1.3.6. Servicio Web.

"Conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web" (13). Los servicios web brindan la posibilidad de interoperabilidad entre aplicaciones desarrolladas en un gran número de lenguajes de programación y se abstraen de la plataforma en que estas se ejecuten, algo muy importante ya que uno de los objetivos de este trabajo es brindar el servicio independientemente de la plataforma en que estén los clientes. Utilizan las bondades de

[15] RSA: Siglas de los apellidos de los inventores del algoritmo (Ron Rivest, Adi Shamir, Leonard Adleman).
[16] PKI: del inglés "Public Key Infraestructure", en español "Infraestructura de clave pública", se refiere a algoritmos de encriptación que usan 2 claves para la encriptación/desencriptación. Una pública con la cual solo se puede encriptar mensajes y otra privada con la cual solo se puede desencriptar los mensajes.
[17] AES: del inglés "Advanced Encryption Standard" en español "Estándar de encriptación avanzado".

la comunicación HTTP[18]. Reducen la complejidad por medio del encapsulamiento ya que los solicitantes y los proveedores del servicio se preocupan por las interfaces necesarias para interactuar. Como resultado, un solicitante de servicio no sabe cómo fue implementado el servicio por parte del proveedor, y éste a su vez, no sabe cómo utiliza el cliente el servicio. Estos detalles se encapsulan en los solicitantes y proveedores. El encapsulamiento es crucial para reducir la complejidad (14).

Los servicios web fueron utilizados como mecanismo de comunicación para que los clientes envíen los mensajes al servidor de notificaciones y para realizar las tareas de administración del servidor desde una aplicación externa.

1.4. Técnica para el modelado del negocio y lenguaje de programación a utilizar.

1.4.1. Lenguaje de definición Integrado (IDEF0).

IDEF0 fue seleccionado principalmente por ser la técnica de modelación de negocio utilizada para modelar los procesos del SIAI, pero su uso se justifica además porque permite representar los modelos de manera que sean fácilmente comprensibles por personas no vinculadas a la ingeniería informática. Permite detectar problemas de organización en el negocio, lo que evita informatizar procesos con flujos incorrectos, algo que fácilmente se pasaría por alto en un proyecto de gran tamaño como SIAI. Incorpora en la misma vista los datos de las actividades, los mecanismos y las reglas de negocio. Es muy usado en sistemas complejos y dinámicos.

1.4.2. Java.

Es un lenguaje de programación libre, con la ventaja de poder ser usado en cualquier plataforma que cuente con la máquina virtual de Java[19] instalada. Cuenta con una amplia documentación disponible para la comunidad de manera gratuita y una gran variedad de API que facilitan el trabajo de los desarrolladores. Se seleccionó para la realización del servidor de notificaciones además porque es el lenguaje para el que mayor documentación brinda el servidor de colas de mensajes HornetQ, al ser multiplataforma permite el despliegue del servidor en cualquier plataforma, lo que hace posible una mayor

[18] HTTP: del inglés "Hypertext Transfer Protocol", en español "Protocolo de Transferencia de Hipertexto".
[19] Máquina Virtual de java: Entorno en el que se ejecutan los programas escritos en Java, su misión principal es la de garantizar la portabilidad de las aplicaciones Java. Se le llama "máquina virtual" porque, sin importar el tipo de computador en el cual se esté ejecutando el programa, crea un computador simulado que proporciona la plataforma correcta para ejecutar estas aplicaciones.

flexibilidad a los beneficiarios de la herramienta. Con Java se pueden publicar servicios web sin necesidad de utilizar servidores externos. Brinda una API muy completa para la interacción con servidores de correo. Permite crear aplicaciones basadas en *plugins* con gran facilidad. Por último, es uno de los lenguajes de programación más utilizados en el mundo por los desarrolladores de software, lo que da una prueba fiable de la popularidad y utilidad de este lenguaje para el desarrollo de aplicaciones informáticas que abarcan un número significativo de escenarios (15).

1.5. Herramientas a utilizar.

1.5.1. Eclipse Indigo.

Es una plataforma de código abierto que se ejecuta sobre la máquina virtual de Java, lo que la convierte en multiplataforma. Permite la integración de herramientas que le adicionan funcionalidades mediante la tecnología de plugins como es el caso de soporte para interpretar y compilar código en lenguajes como Python, PHP o Java, algo que será muy útil durante el desarrollo del servidor, los plugins y las API para los distintos lenguajes desde los que se pudiera hacer uso del servidor de notificaciones. Cuenta con el soporte para realizar interfaces visuales SWT que permite crear interfaces de manera sencilla. Además es el IDE[20] utilizado en el desarrollo del SIAI lo que facilita la estandarización de la documentación y estilos propios del proyecto (16).

1.5.2. HornetQ.

HornetQ es un servidor de colas de mensajes implementado en lenguaje Java y por tanto se integra completamente con este lenguaje. Tiene una extensa documentación que facilita el trabajo de los desarrolladores, permite el manejo de mensajes de tipos como **bytes**, **mapas**, **objetos de Java**, **stream** o **texto**, brinda una API de administración para Java y es compatible con JMS 1.1 (característica que no cumplen otros como RabbitMQ). Es notablemente más rápido que otros servidores de colas de mensajes como son OpenMQ, SwiftMQ y ActiveMQ según pruebas de rendimiento realizadas cuando los mensajes son grandes (17), (18). Hace uso de la función IO[21] asíncrono para guardar los datos en disco (requiere la librería "libaio[22]"), disponible en sistemas Linux que será la plataforma en la que será desplegado el servidor dentro del SIAI. También puede ser desplegado en sistemas Windows aunque no podrá hacer

[20] IDE: del inglés "Integrated Development Environment", en español "Entorno de Desarrollo Integrado".
[21] IO: Del inglés in/out, en español entrada/salida.
[22] Libaio: Librería que permite el uso del sistema de entrada/salida asíncrono de Linux.

uso de la característica IO asíncrono y esta será sustituida por la NIO[23] de Java que brinda velocidades de lectura-escritura bastante rápidas aunque un poco más lentas que la anterior.

1.5.3. Visual Paradigm.

Visual Paradigm para UML[24] es una herramienta CASE[25] profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Proporciona abundantes tutoriales de UML y la forma en que está organizada permite trabajar con ella de manera intuitiva. Se integra con el IDE Eclipse (19). El uso de esta herramienta en el desarrollo del presente trabajo permitió dibujar todos los diagramas de clases, generar diagramas a partir del código fuente y generar la base de datos SQLite a partir del modelo físico de datos.

1.5.4. SQLite.

SQLite es un pequeño sistema de gestión de base de datos (SGBD) multiplataforma que opera sobre un único archivo binario. Permite que la información de la base de datos sea portable y es realmente útil cuando la cantidad de datos y/o acceso a estos datos es relativamente pequeña. Con SQLite se evitan las conexiones al estilo de los servidores de bases de datos tradicionales, *"se elimina la latencia en las conexiones y la velocidad de acceso a la información contenida es superior a la de sistemas como MySQL o PostgreSQL para pequeños volúmenes de datos"* (20). No es necesario configurarlo ni instalarlo para hacer uso de las bases de datos lo que facilita su despliegue y se evita la necesidad de iniciar o detener servicios para poder utilizarlo. Además el tamaño del archivo con los datos es generalmente pequeño ya que solamente se reserva en el disco duro el espacio ocupado por los datos. Cuenta con cláusulas de conflicto que permiten ejecutar determinadas sentencias indicando un algoritmo específico a ejecutar en caso de violaciones de restricción en la BD, lo que evita la necesidad de tratamiento de excepciones innecesarias del lado del cliente o programación del lado de la base de datos. Las cláusulas de conflicto constituyen una característica destacable de SQLite ya que otros SGBD, incluso de los tradicionales no cuentan con esta potencialidad. Por otra parte cuenta con un driver para ser utilizado desde Java (21).

[23] NIO: Colección de APIs de java que proporcionan soporte para operaciones IO de bajo nivel en los sistemas operativos modernos.
[24] UML: del Inglés "Unified Model Languaje", en español "Lenguaje Unificado de Modelado".
[25] CASE: del inglés "Computer Aided Systems Engenieering", en español "Ingeniería de Sistemas Ayudado por Computadoras".

En el servidor de notificaciones se realizó un modelo de datos relacional para la gestión de los clientes, las colas y otras configuraciones necesarias para controlar el flujo de las comunicaciones. Dichas configuraciones generan un volumen de información relativamente pequeño. De utilizar un SGBD tradicional como PostgreSQL, Oracle u otro, se incurriría en un consumo de recursos innecesario, además de que gestionar la conectividad con el servidor de base de datos sería otro aspecto a tener en cuenta para garantizar la fiabilidad en el reenvío de las notificaciones. Por otra parte se sacrificaría en cierta medida la portabilidad, de ahí que el uso de SQLite haya sido conveniente para el desarrollo del trabajo.

1.6. Metodología de desarrollo a utilizar.

1.6.1. Rational Unified Process (RUP).

Partiendo del hecho de que el Módulo de notificaciones forma parte del SIAI es necesario que cumpla con los estándares de documentación y organización del mismo, siendo esta la principal razón por la que se seleccionó la metodología RUP para regir el proceso de creación del mismo. También fue seleccionado por la necesidad de contar con la mayor documentación posible sobre la herramienta para que futuros desarrolladores puedan agregarle funcionalidades a la misma consultando inicialmente las especificaciones plasmadas en la documentación. RUP está basado en las mejores prácticas del campo de la ingeniería de software, permite la temprana mitigación de los riesgos y proporciona mecanismos para gestionar la complejidad de los proyectos.

1.7. Conclusiones.

Al finalizar este capítulo se puede concluir que a pesar de existir varias herramientas y soluciones a nivel nacional e internacional relacionadas con la investigación, ninguna es lo suficientemente versátil o adaptable para resolver el problema planteado. Para lograr el desarrollo de una aplicación portable e interoperable como la que se requiere, el uso de servicios web, Java como lenguaje de programación y SQLite como motor de base de datos se considera una combinación acertada para garantizar la portabilidad, sin sacrificar velocidad. También el hecho de utilizar HornetQ como servidor de mensajería junto al API JMS permite el desarrollo de una herramienta que cumple con los estándares vinculados a la manipulación de mensajes y su fiabilidad. Además la posibilidad de agregar *plugins* a la herramienta potencia su utilidad para el usuario final, que puede adaptarla a sus necesidades específicas.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1. Introducción.

En este capítulo se realiza la modelación de negocio, definición de requisitos funcionales y no funcionales, definición de los actores del sistema, los casos de uso del sistema y su descripción, como guía para la posterior creación del Servidor de notificaciones usando las tecnologías definidas en el capítulo anterior.

2.2. Modelo de negocio.

2.2.1. Proceso Reenviar notificación.

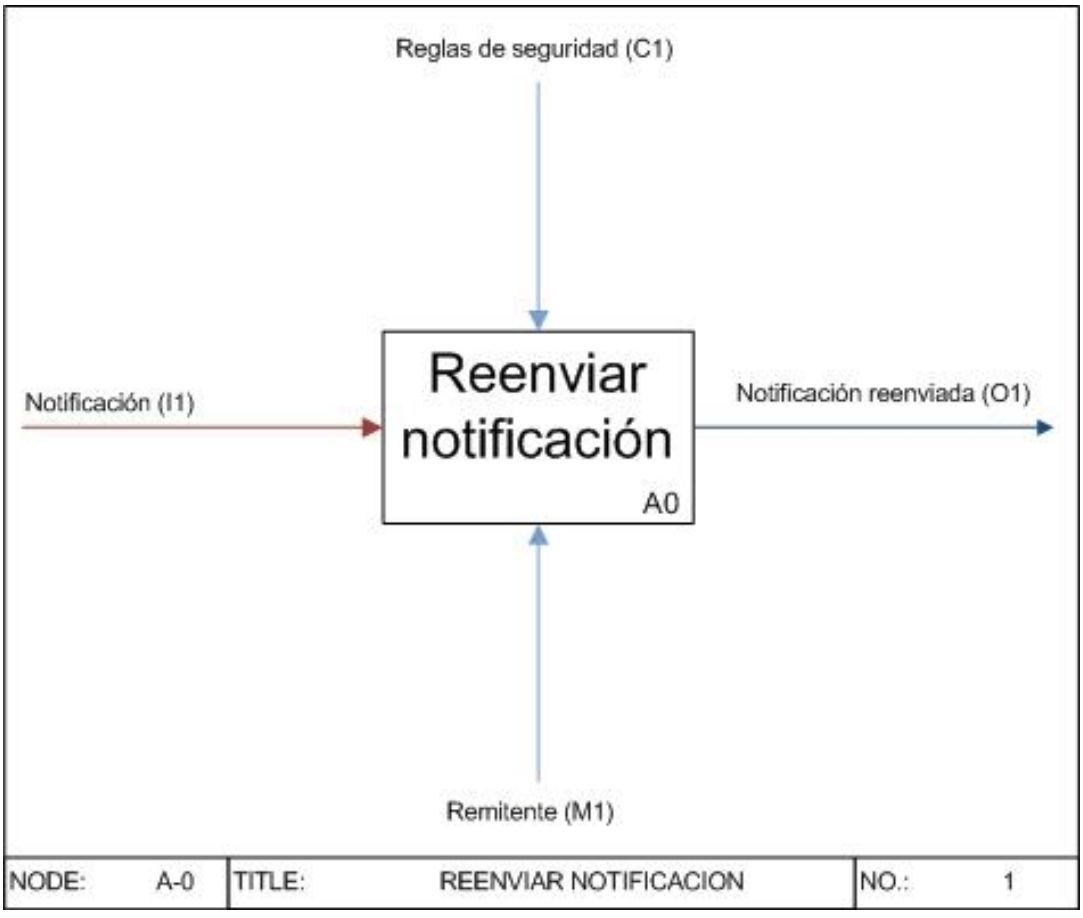

Figura 2. Diagrama Reenviar Notificación.

2.2.2. Entradas, controles, salidas y mecanismos.

Tabla 1: Ficha del proceso A0.

Ficha de Proceso		
Proceso	Reenviar Notificación.	
Notación:	A0	
Entradas:	I1	Notificación. Incluye • Identificador (establecido por el remitente). • Cuerpo (contenido de texto de la notificación). • Remitente (identificador del remitente). • Adjuntos (listado de ficheros adjuntos). • Atributos (conjunto de atributos extras propios del tipo de notificación específico).
Controles:	C1	Reglas de seguridad. Se refiere a las reglas definidas en la entidad para la manipulación de las notificaciones.
Mecanismos:	M1	Remitente. Aplicaciones clientes con permiso para enviar notificaciones.
Salidas:	O1	Notificación reenviada. Incluye • Identificador (generado por el Servidor de notificaciones). • Cuerpo (contenido de texto de la notificación). • Remitente (identificador del remitente). • Adjuntos (listado de ficheros adjuntos). • Atributos (conjunto de atributos extras propios del tipo de notificación específico).

2.2.3. Descripción del proceso.

Es el proceso principal que hace referencia al flujo de actividades que se llevan a cabo para el correcto reenvío de las notificaciones a los usuarios o aplicaciones de destino, tomando como punto de partida la información del destinatario contenida en una notificación de entrada. La aplicación o usuario cliente, cumpliendo con las Reglas de seguridad que determinan el procesamiento de las notificaciones se autentica con el servidor de notificaciones y realiza la solicitud de reenvío de su notificación. Una vez la notificación es aceptada por el servidor, este realiza una de las siguientes acciones:

- ✓ Espera que su destinatario solicite recibirla para luego entregarla.
- ✓ Intenta reenviarla tantas veces sea necesario hasta que sea recibida por el destinatario o se alcance la cantidad máxima de intentos de reenvío configurados, tras los cuales el mensaje es enviado a colas especiales para almacenamiento indefinido, donde solo bajo una petición de un cliente puede ser extraído. Como resultado del flujo se obtiene una notificación reenviada a su destinatario.

2.3. Requisitos funcionales.

RF 1 - Autenticar usuario.

Funcionalidad que permite verificar la validez de las credenciales (usuario, contraseña y permisos para realizar la acción solicitada) de una aplicación del sistema previamente registrada o del administrador.

RF 2 - Iniciar servicios de comunicación para usuarios.

Funcionalidad que permite iniciar los servicios de envío y recepción de notificaciones en el servidor y también los de administración del mismo.

2.1 - Iniciar servicios de notificación.

2.2 - Iniciar servicios de administración.

RF 3 - Cambiar contraseña de administrador.

Funcionalidad que permite cambiar la contraseña del administrador del servidor de notificaciones.

RF 4 - Gestionar Aplicación.

Funcionalidad que a partir del listado de aplicaciones registradas en el sistema (hace referencia al requisito **Listar Aplicaciones**), permite seleccionar alguna de ellas para ser modificada, eliminada; o adicionar una nueva.

4.1 - Adicionar Aplicación.

Funcionalidad que permite adicionar una nueva aplicación. Registrando:

4.1.1 - Nombre de la aplicación.

Campo único, obligatorio, de longitud variable de hasta 20 caracteres y tipo texto.

4.1.2 - Contraseña de la aplicación.

Campo obligatorio, de longitud variable de hasta 20 caracteres y tipo texto.

4.1.3 - Permisos de la aplicación.

Campo obligatorio de longitud 1 carácter y tipo texto, al que se asignan los valores ("s" (para envío),"r" (para recepción),"b" (para ambos).

4.1.4 - Descripción de la aplicación.

Campo opcional de longitud 255 caracteres y tipo texto.

4.2 - Mostrar Aplicación.

Funcionalidad que permite mostrar de una aplicación previamente registrada su nombre, contraseña, permisos y descripción.

4.3 - Modificar Aplicación.

Funcionalidad que permite modificar de una aplicación previamente registrada su contraseña, permisos y descripción.

4.3.1 - Contraseña de la aplicación.

Campo obligatorio, de longitud variable de hasta 20 caracteres y tipo texto.

4.3.2 - Permisos de la aplicación.

Campo obligatorio de longitud 1 carácter y tipo texto, al que se asignan los valores: "s" (para envío),"r" (para recepción),"b" (para ambos).

4.3.3 - Descripción de la aplicación.

Campo opcional de longitud 255 caracteres y tipo texto.

4.4 - Eliminar Aplicación.

Funcionalidad que permite eliminar una aplicación previamente registrada en el servidor de notificaciones.

RF 5 - Listar Aplicaciones.

Funcionalidad que permite mostrar un listado de las aplicaciones registradas en el servidor de notificaciones, mostrando de cada una de ellas su nombre, contraseña, permisos y descripción.

RF 6 - Gestionar servidor de salida de notificaciones a usuarios.

Funcionalidad que permite a partir del listado de servidores registrados en el sistema (hace referencia al requisito **Listar servidores de salida de notificaciones a usuarios**), seleccionar alguno de ellos para ser mostrado, modificado o eliminado; o adicionar uno nuevo.

6.1 - Adicionar servidor de salida.

Funcionalidad que permite adicionar la información de conexión a un servidor de salida usado por el servidor de notificaciones. Registrando:

6.1.1 - Identificador del servidor

Campo único, obligatorio, de longitud variable hasta 20 caracteres y tipo texto.

6.1.2 - Descripción del servidor.

Campo opcional, de longitud variable de hasta 255 caracteres y tipo texto.

6.1.3 - Listado de propiedades.

Colección de campos únicos, de longitud variable de hasta 255 caracteres y tipo texto.

6.2 - Modificar servidor de salida.

Funcionalidad que permite modificar de un servidor de salida su Descripción y Listado de propiedades.

6.3 - Eliminar servidor de salida.

Funcionalidad que permite eliminar un servidor de salida previamente registrado en el servidor de notificaciones.

RF 7 - Listar servidores de salida de notificaciones a usuarios.

Funcionalidad que permite mostrar un listado de los servidores de salida de notificaciones a usuarios registrados en el servidor de notificaciones, mostrando de cada uno de ellos su identificador y descripción.

RF 8 - Gestionar canal de comunicación.

Funcionalidad que permite a partir del listado de canales de comunicación registrados en el sistema (hace referencia al requisito **Listar canales de comunicación**) seleccionar uno de ellos para ser modificado o eliminado; o adicionar uno nuevo.

8.1 - Adicionar canal de comunicación.

Funcionalidad que permite adicionar un canal de comunicación al servidor de notificaciones.

Registrando en el caso de los canales de usuario:

8.1.1 - Nombre del canal.

Campo obligatorio, único, de longitud variable hasta 20 caracteres y tipo texto.

8.1.2.- Identificador de un Plugin.

Campo obligatorio de longitud variable hasta 20 caracteres y tipo texto.

8.1.3 - Identificador del servidor que utilizará el Plugin para manejar el mensaje.

Campo obligatorio de longitud variable hasta 20 caracteres y tipo texto.

8.1.4 - Identificador de la cola de mensajes donde se almacenarán los mensajes.

Campo obligatorio de longitud variable hasta 20 caracteres y tipo texto.

8.1.5 - Estado.

Campo obligatorio de longitud variable hasta 8 caracteres y tipo texto.

8.1.6 - Descripción del canal

Campo opcional de longitud variable hasta 255 caracteres y tipo texto.

Registrando en el caso de los canales de sistema:

8.1.7 - Identificador de la aplicación que recibirá mensajes por ese canal.

Campo obligatorio y único de longitud variable hasta 20 caracteres y tipo texto.

8.2.2 - Estado

Campo obligatorio de longitud variable hasta 8 caracteres y tipo texto.

8.2.3 - Descripción del canal.

Campo opcional de longitud variable hasta 255 caracteres y tipo texto.

Cuando se registra un canal de sistema automáticamente se crea en el servidor una cola de mensajes con identificador igual al de la aplicación que recibirá mensajes por ese canal.

8.2 - Modificar canal de comunicación.

Funcionalidad que permite a partir del listado de canales de comunicación registrados en el sistema (hace referencia al requisito **Listar canales de comunicación**) seleccionar uno para modificar.

En caso de los canales de usuario los campos:

8.2.1 - Identificador del Plugin asociado.

8.2.2 - Identificador del servidor asociado.

8.2.3 - Identificador de la cola de mensajes asociada.

8.2.4 - Estado.

8.2.5 - Descripción del canal.

8.3 - Eliminar canal de comunicación.

Funcionalidad que permite a partir del listado de canales de comunicación registrados en el sistema (hace referencia al requisito **Listar canales de comunicación**) seleccionar uno para ser eliminado.

En el caso de los canales de sistema se eliminan cuando se elimina la aplicación a la cual pertenecen o se le deniegan los permisos de recepción de mensajes, caso en el cual se elimina también la cola de mensajes asociada al canal.

RF 9 - Listar canales de comunicación.

Funcionalidad que permite mostrar un listado de los canales de comunicación del servidor de notificaciones.

RF 10 - Asignar canal a Aplicación.

Funcionalidad que permite a partir de los listados de aplicaciones y de canales (hace referencia a los requisitos **Listar aplicaciones** y **Listar canales de comunicación**) asignar un canal de comunicación a una aplicación del Servidor de notificaciones para habilitar el envío de notificaciones por ese canal por parte de esa aplicación.

RF 11 - Gestionar colas de mensajes.

Funcionalidad que permite a partir del listado de colas de mensajes registradas en el sistema (hace referencia al requisito **Listar colas de mensajes**) seleccionar una de ellas para ser modificada o eliminada; o adicionar una nueva.

11.1 - Adicionar cola de mensajes.

Funcionalidad que permite adicionar una cola de mensajes al servidor de notificaciones. Registrando

11.1.1 - Nombre de la cola.

Campo obligatorio, único, de longitud variable hasta 20 caracteres y tipo texto.

11.1.2 - Descripción de la cola.

Campo opcional, de longitud variable hasta 255 caracteres y tipo texto.

11.1.3 – Tiempo entre reenvíos.

Campo obligatorio, de longitud variable y tipo entero.

11.1.4 – Capacidad máxima.

Campo obligatorio, de longitud variable y tipo entero.

11.1.5 – Cantidad de intentos de envío.

Campo obligatorio, de longitud variable y tipo entero.

11.2 - Modificar cola de mensajes.

Funcionalidad que permite modificar de una cola de mensajes registrada con anterioridad su descripción, tiempo entre intentos de reenvío, capacidad máxima y cantidad de intentos de reenvío.

11.3 - Eliminar cola de mensajes.

Funcionalidad que permite eliminar una cola de mensajes del servidor de notificaciones.

RF 12 - Listar Colas de Mensajes.

Funcionalidad que permite mostrar un listado de las colas de mensajes existentes en el servidor de notificaciones, mostrando su nombre, descripción, tiempo entre intentos de reenvío, capacidad máxima, cantidad de intentos de reenvío.

RF 13 – Cambiar contraseña de administrador.

Funcionalidad que permite cambiar la contraseña del administrador del servidor de notificaciones.

RF 14 - Reenviar notificación.

Funcionalidad que permite reenviar una notificación recibida hasta su destinatario.

14.1 - Almacenar notificación en cola de mensajes.

Funcionalidad que permite almacenar una notificación en el servidor de notificaciones.

14.2 - Extraer notificación de cola de mensajes

Funcionalidad que permite extraer una notificación previamente almacenada en el servidor de notificaciones para ser entregada a su destinatario.

RF 15 - Iniciar entrega de notificación.

Funcionalidad que permite iniciar la entrega de una notificación almacenada en el servidor de notificaciones, luego de una petición realizada por el destinatario.

RF 16 - Visualizar ayuda del sistema.

Funcionalidad que permite visualizar la ayuda para el administrador del servidor de notificaciones.

2.4. Requisitos no funcionales.

2.4.1. RNF Seguridad.

- ✓ **RNF Confidencialidad.**
  Todas las acciones de los clientes sobre las notificaciones se realizarán siempre que la solicitud sea acompañada de credenciales (usuario y contraseña) de autenticación.
- ✓ **RNF Disponibilidad.**
  Las notificaciones estarán disponibles para ser consumidas por las aplicaciones definidas desde el momento en que arriben al servidor de notificaciones.

2.4.2. RNF Usabilidad.

- ✓ **RNF Facilidad de uso.**

  Se debe garantizar que la herramienta pueda ser utilizada de manera intuitiva mediante las API de conexión para los clientes.

2.4.3. RNF Fiabilidad.

Este requerimiento está relacionado con la capacidad del usuario para confiar en las respuestas del sistema, en un sentido técnico, es decir, que la funcionalidad del sistema no se vea afectada por factores técnicos ajenos al mismo como pueden ser: fallas de software o hardware, fallos en la conexión o interrupción del servicio eléctrico (22).

- ✓ **RNF Disponibilidad del sistema.**
  El sistema deberá estar disponible el 98% del tiempo.
- ✓ **RNF Tiempo medio de reparación.**
  Si el fallo involucra hardware debe tener un tiempo medio de reparación de 24 horas. En los casos que involucra software debe ser de 3 horas.

2.4.4. RNF Eficiencia.

- ✓ **RNF Capacidad.**

  El número de clientes concurrentes haciendo peticiones al sistema como máximo debe ser 100 aplicaciones con el objetivo de mantener el rendimiento equilibrado.
- ✓ **RNF Tiempo de respuesta de la base de datos de configuración.**

  Las consultas a la base de datos de configuración no deberán exceder de 1 segundo como tiempo de respuesta.

2.4.5. RNF Soporte.

- ✓ **RNF Portabilidad.**

  El código basado en J2EE es portable a cualquier plataforma de sistema operativo, sólo es necesario contar con la versión correspondiente del JDK para cada uno.
- ✓ **RNF Estándares de codificación.**

  Los estándares de programación a seguir se rigen por las pautas y especificaciones establecidas por los desarrolladores del SIAI para el lenguaje de programación Java. Se especifican en el documento "Estándar de codificación Java" que se encuentra en el expediente del proyecto.

2.4.6. RNF Interfaz.

- ✓ **Interfaces Hardware:**

Estaciones cliente:

- Conexión: 100 Mbps
- RAM: 1 GB
- Procesador: Intel(R) Core(TM) 2 Duo CPU E4500 @ 2.20GHz × 2

Servidor:

- Conexión: 100 Mbps
- Procesador: Intel(R) Core(TM) i3-2100 CPU @ 3.10 GHz
- RAM: 2 GB.

- Espacio en disco disponible: 5 GB.

✓ **Interfaces Software.**

Estaciones cliente:

- Sistemas operativos GNU/Linux, Windows XP/7.

Servidores

- Sistemas operativos basados en GNU/Linux, Windows XP/7/8.
- JDK 1.6.x. o superior.

✓ **Interfaces de Comunicación.**

- Conexión 100 Mbps.

2.5. Casos de uso del sistema.

2.5.1. Definición de actores.

Tabla 2: Actores del sistema.

Actor	Descripción
Administrador	Interactúa con el "Módulo de notificaciones", teniendo todos los permisos y privilegios sobre el sistema. Dentro de las funciones a realizar se destacan las tareas de configuración y gestión del Servidor de notificaciones.
Aplicación	Interactúa con el "Módulo de notificaciones" como aplicación externa que solicita el reenvío de notificaciones a otras aplicaciones o usuarios, convirtiéndose en cliente del Servidor de notificaciones.

2.5.2. Diagrama de casos de uso del sistema.

Los casos de uso mostrados con color gris son críticos.

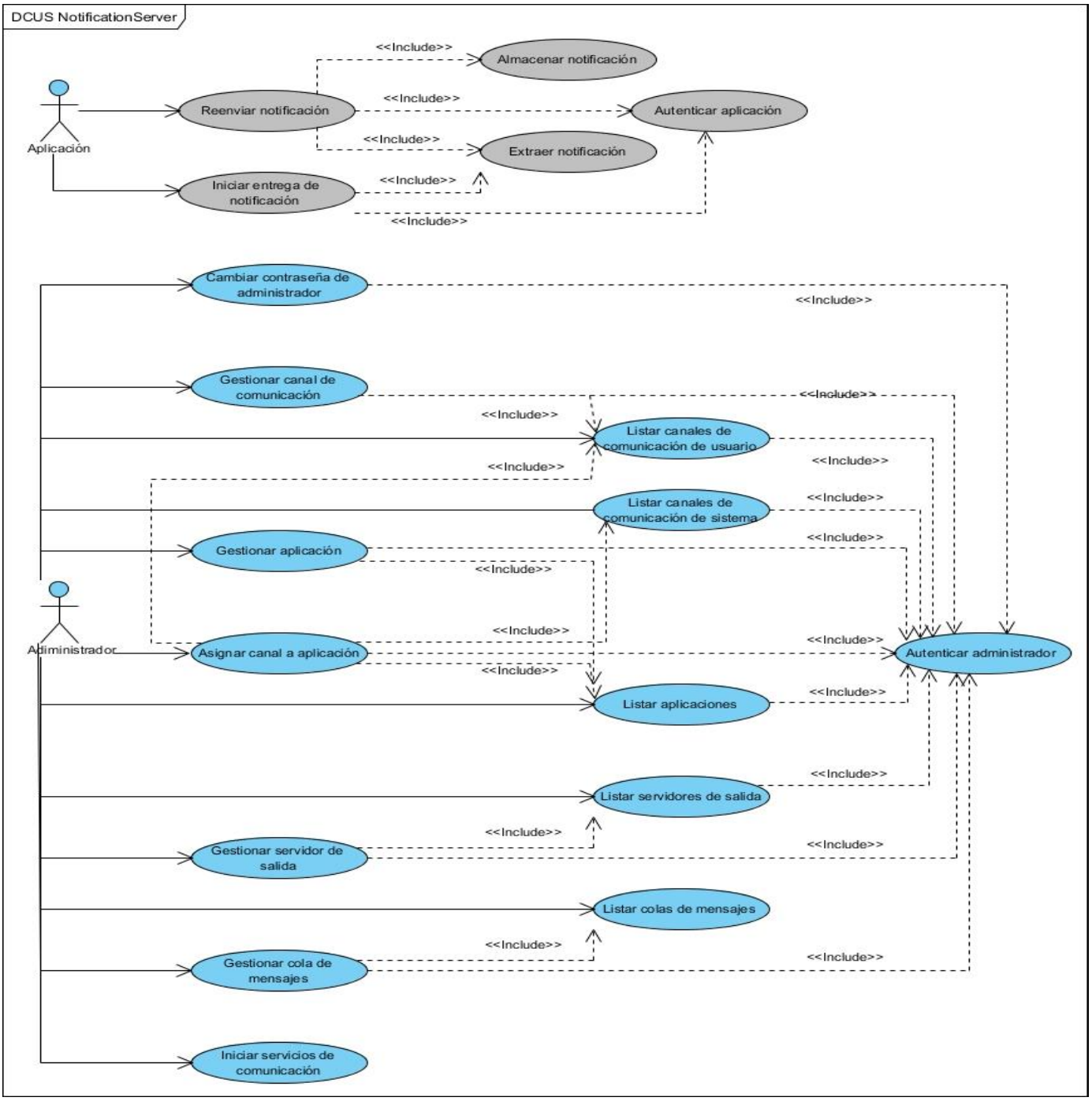

Figura 3. Casos de uso del sistema.

2.5.3. Descripción de los casos de uso.

Tabla 3. Descripción del caso de uso Reenviar Notificación.

Caso de Uso:	Reenviar notificación
Actores:	Aplicación (Inicia)
Resumen:	El caso de uso se inicia cuando una Aplicación necesita enviar una notificación a un usuario u otra aplicación.
Precondiciones:	Autenticarse en el Sistema.
Referencias	RF 14
Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Reenviar notificación"	
Acción del Actor	**Respuesta del Sistema**
1. Envía una notificación al Servidor de notificaciones especificando el siguiente dato: • Nombre del Canal de Comunicación.	2. Almacena la notificación en la cola asociada al canal especificado.
	3. Extrae la Notificación de la Cola y la envía a su destinatario (en el caso de las notificaciones a Usuarios).
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1 Retorna a la acción 2 de la sección **Reenviar notificación**.

Poscondiciones	Se reenvía la Notificación a su destinatario.

Las descripciones de los restantes CU críticos pueden encontrarse en el Anexo # 1 y los secundarios en el expediente de proyecto.

2.6. Conclusiones.

Luego de la realización de este capítulo se concluye que con la realización del modelo de negocio y los requisitos es posible encontrar un punto de entendimiento con el cliente que tributa a la comprensión de la aplicación a desarrollar y traza las primeras líneas para lograr un resultado satisfactorio. La definición de los casos de uso del sistema y su descripción permiten guiar el proceso de desarrollo de la solución de manera precisa y acorde con los requisitos identificados y las peticiones del cliente.

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

3.1. Introducción.

En este capítulo se describen los patrones utilizados, los diagramas de paquetes, de componentes, el modelo de datos y el diagrama de despliegue que permiten el desarrollo y organización del diseño e implementación del servidor de notificaciones ilustrando cada paso para brindar un mejor entendimiento del proceso. Además de las principales funcionalidades implementadas.

3.2. Arquitectura del sistema.

"La arquitectura de software es importante debido a que los sistemas de software crecen de forma tal que resulta muy complicado que sean diseñados, especificados y entendidos por un solo individuo. Uno de los aspectos que motivan el estudio en este campo es el factor humano, en términos de aspectos como inspecciones de diseño, comunicación a alto nivel entre los miembros del equipo de desarrollo, reutilización de componentes y comparación a alto nivel de diseños alternativos". (23)

3.2.1. Arquitectura del servidor de notificaciones.

Para el desarrollo del sistema se utilizó una arquitectura basada en *plugins* y se agruparon las funcionalidades en varios módulos. El módulo **Entrada de mensajes** contiene los servicios web que permiten la entrada de las notificaciones de los clientes al servidor. En el de **Seguridad** se encuentran los mecanismos para la encriptación de los mensajes y el control de acceso a las funcionalidades y recursos del servidor de notificaciones. La creación de los mensajes JMS y preparación de las notificaciones se realiza en el módulo de **Manipulación de mensajes**. Las operaciones de **almacenamiento y extracción de los mensajes** se realizan en el módulo del mismo nombre que es el encargado de comunicarse con el servidor de colas de mensajes HornetQ. En el módulo **Salida de mensajes** se encuentran las funcionalidades encargadas del control de los *plugins* de salida de los diferentes tipos de notificaciones y las comunicaciones con estos. Las operaciones de configuración del servidor de notificaciones se realizan a través del módulo de **Administración de configuraciones** y las de acceso a los datos almacenados en la base de datos a través del módulo de **Acceso a datos**, que tiene el control sobre la base de datos.

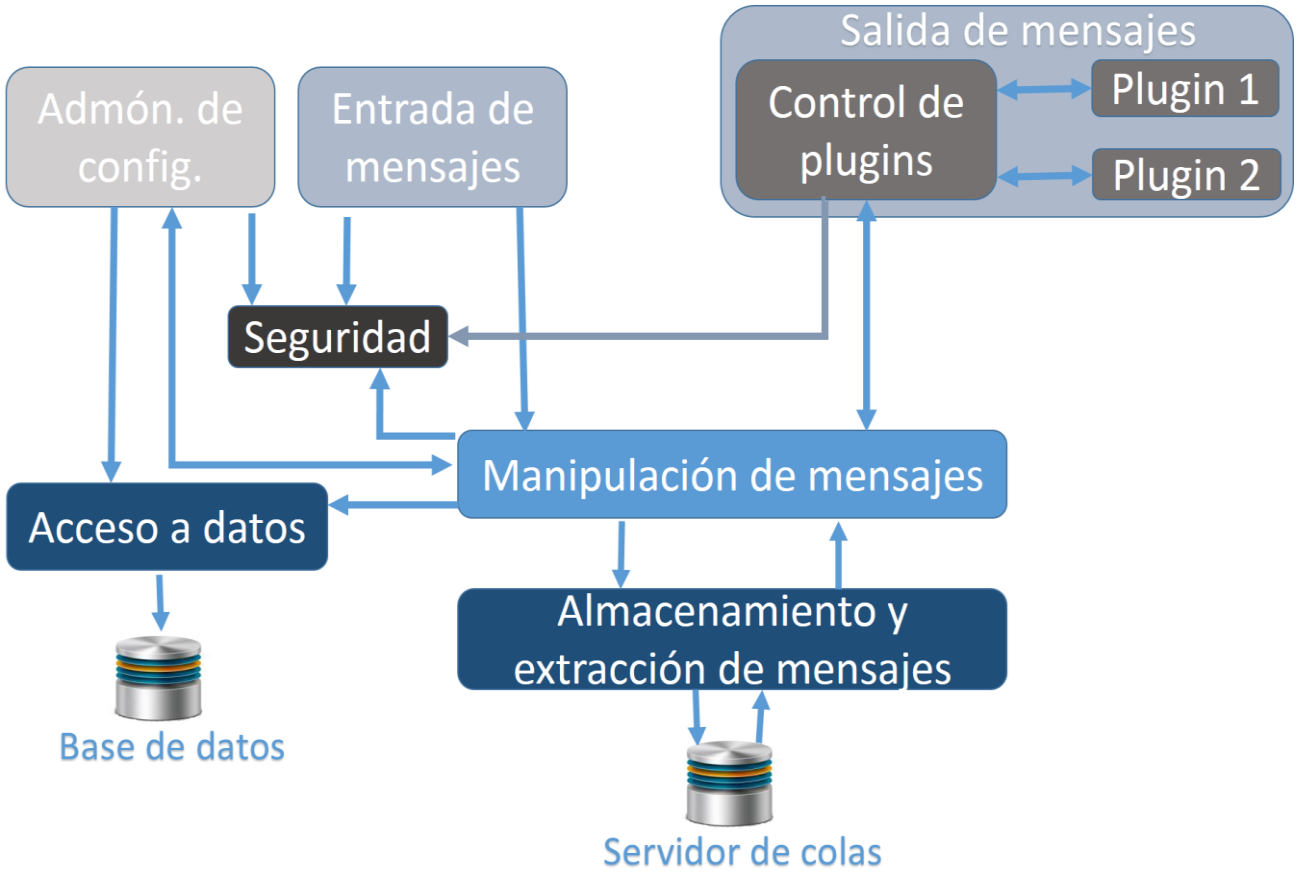

Figura 4. Arquitectura del Servidor de notificaciones.

3.3. Patrones.

Los patrones son los que definen la estructura de un sistema software, los cuales a su vez se componen de subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema. (24)

3.3.1. Patrón Singleton.

El uso de este patrón permite controlar la instanciación de determinadas clases que por sus características es necesario que solo exista una instancia pero que a la vez pueda ser usada desde varias clases dentro del sistema. Su uso fue especialmente útil en la clase controladora de Base de datos y la conexión al servidor de colas HornetQ donde era necesario mantener una sola conexión. Este patrón pertenece al grupo de los patrones creacionales.

3.4. Diagrama de paquetes.

La siguiente figura representa el diagrama de paquetes del Servidor de notificaciones.

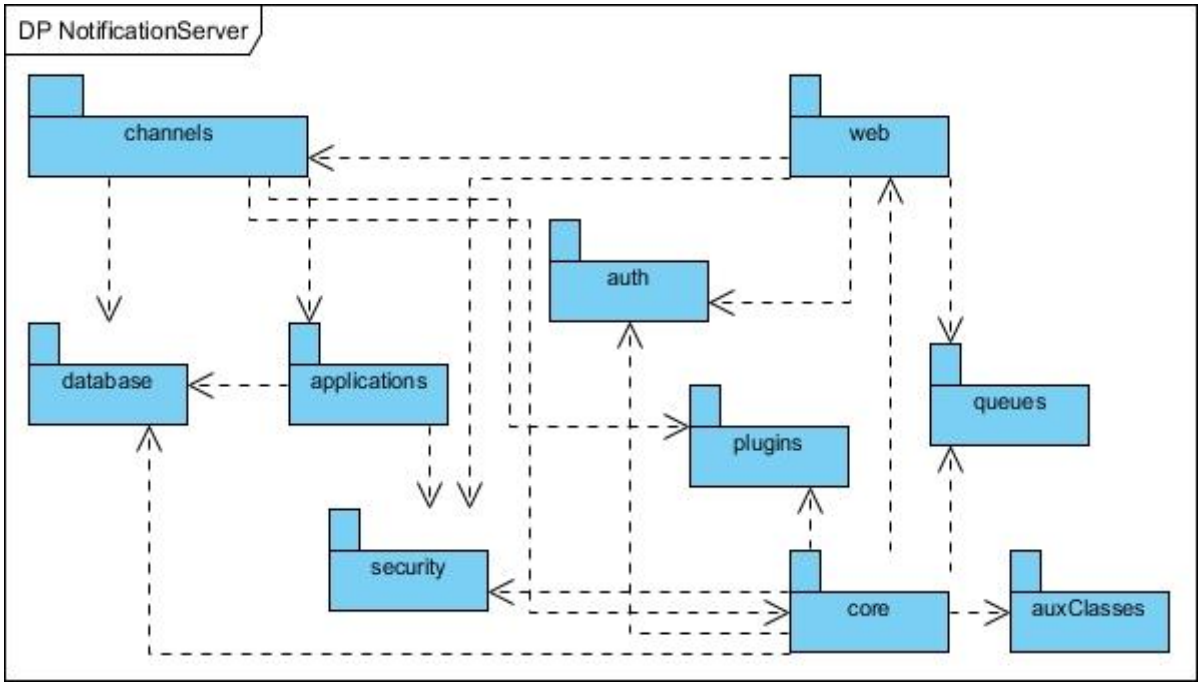

Figura 5. Diagrama de paquetes del servidor de notificaciones.

3.5. Diagrama de clases del diseño.

El Diagrama de Clases de Diseño describe gráficamente las especificaciones de las Clases de Software y las Interfaces en una aplicación. Y contiene las definiciones de las entidades de software en vez de conceptos del mundo real. Es el diagrama principal de diseño y análisis para un sistema. En él, la estructura de clases del sistema se especifica, con relaciones entre clases y estructuras de herencia. (25)

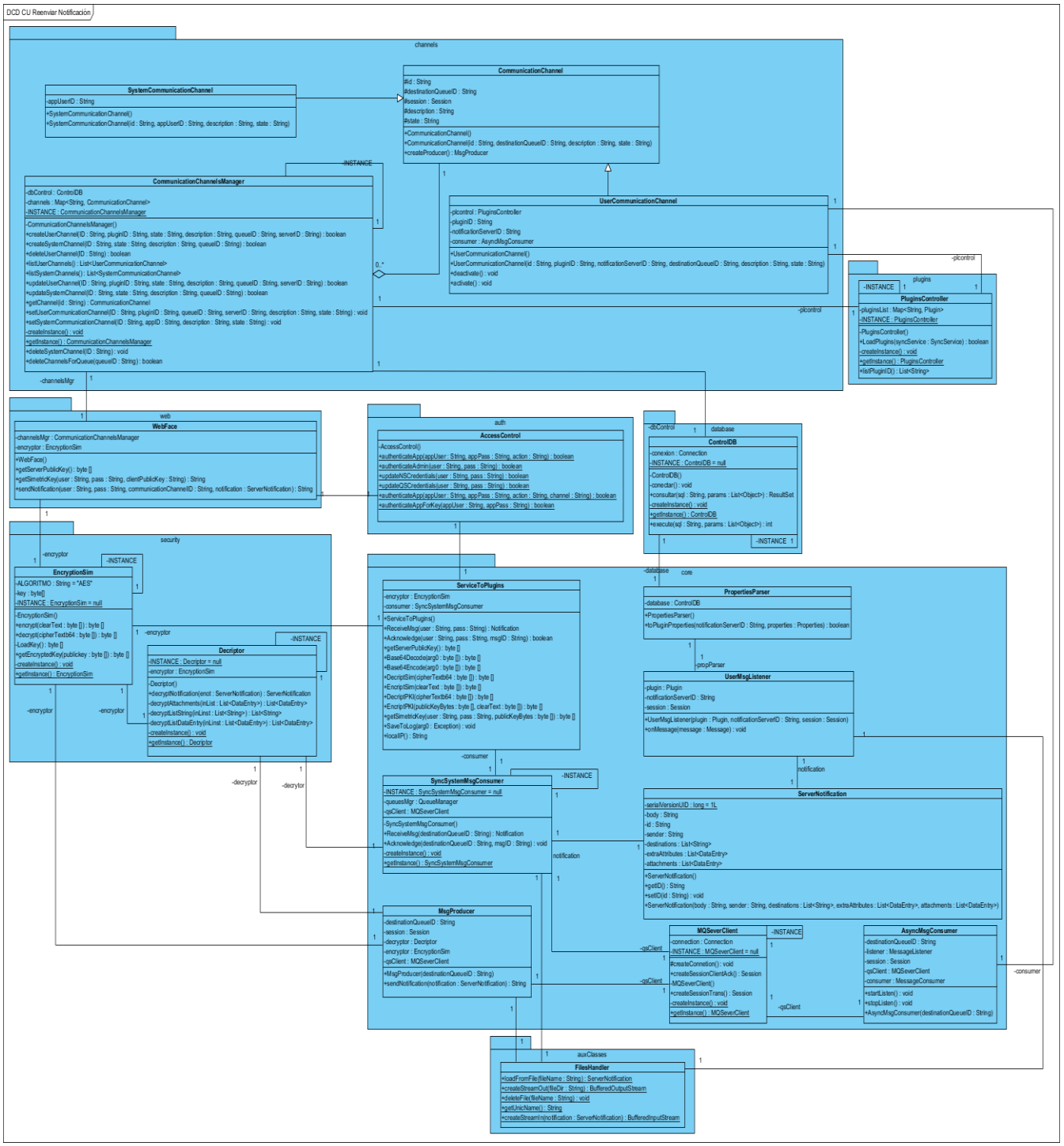

Figura 6. Diagrama de clases del diseño del Caso de uso Reenviar Notificación.

Los diagramas de clases del diseño de los demás Casos de uso críticos pueden encontrarse en el Anexo # 2. Los de los Casos de uso secundarios se encuentran en el expediente del proyecto.

3.6. Modelo físico de datos.

Este modelo representa las tablas donde se almacenarán los datos en la BD[26] de la aplicación.

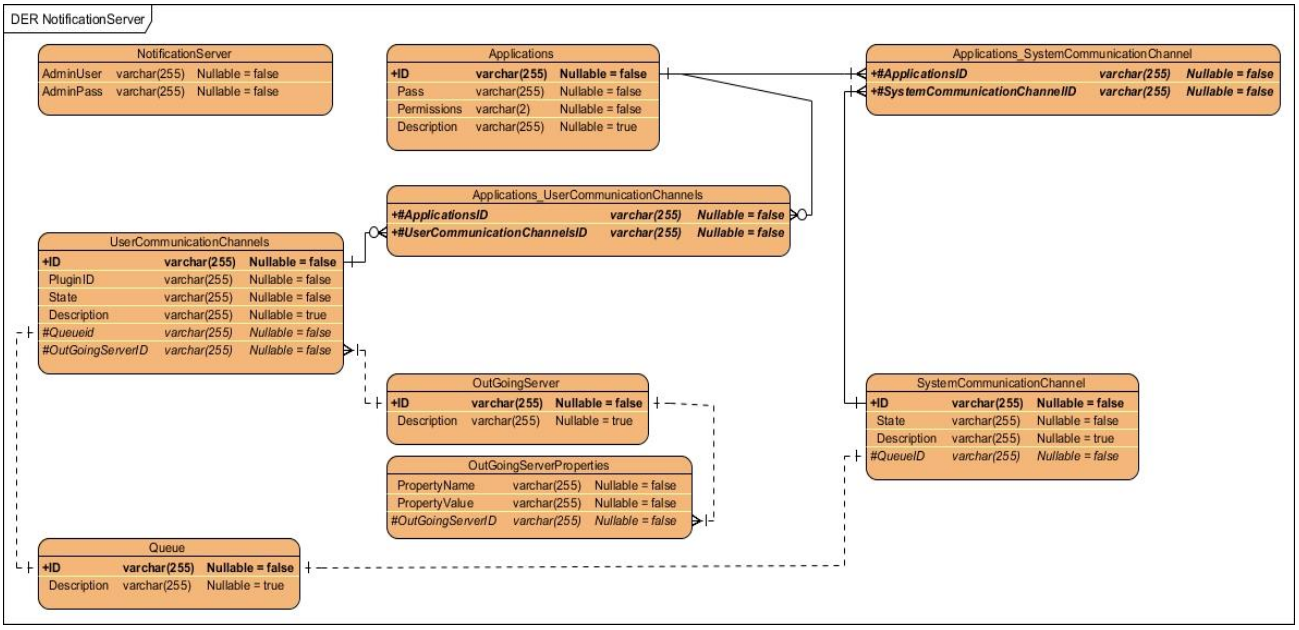

Figura 7. Modelo físico de datos del servidor de notificaciones.

La descripción de los atributos de los modelos de datos puede encontrarse en el Anexo # 3.

3.7. Diagrama de despliegue.

La siguiente figura muestra el diagrama de despliegue del servidor de notificaciones.

[26] BD: Base de datos.

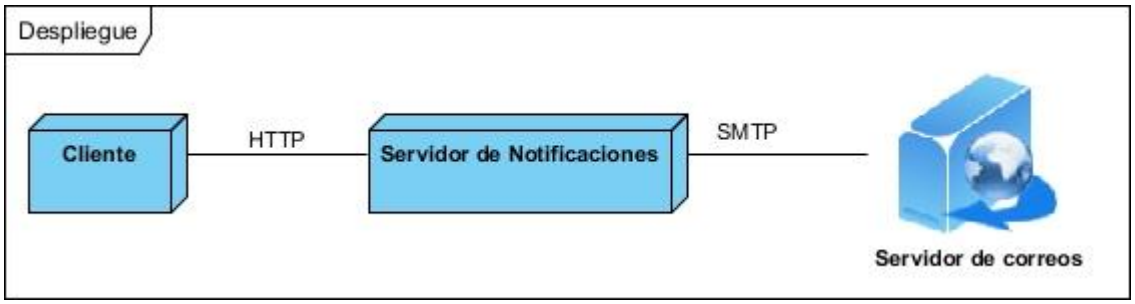

Figura 8. Diagrama de despliegue del Servidor de notificaciones.

3.7.1. Descripción de los nodos.

Tabla 4. Descripción de los nodos.

Nodo	Descripción
	Representa a un cliente que se comunica con el servidor de notificaciones para enviar o recibir una notificación.
	Servidor de notificaciones que recibirá las notificaciones de los clientes para su posterior reenvío. Este servidor contiene al servidor de colas de mensajes HornetQ embebido para el almacenamiento de los mensajes.
	Servidor de correos a través del cual el servidor de notificaciones reenviará las notificaciones de correo electrónico.

3.8. Diagrama de Componentes.

Los Diagramas de Componentes ilustran las piezas del software, controladores embebidos, etc. que conformarán un sistema. Un diagrama de Componentes tiene un nivel más alto de abstracción que un diagrama de clase usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Estos son bloques de construcción, ya que eventualmente un componente puede comprender una gran porción de un sistema. (26)

3.8.1. Diagrama de componentes del Caso de uso Reenviar notificación.

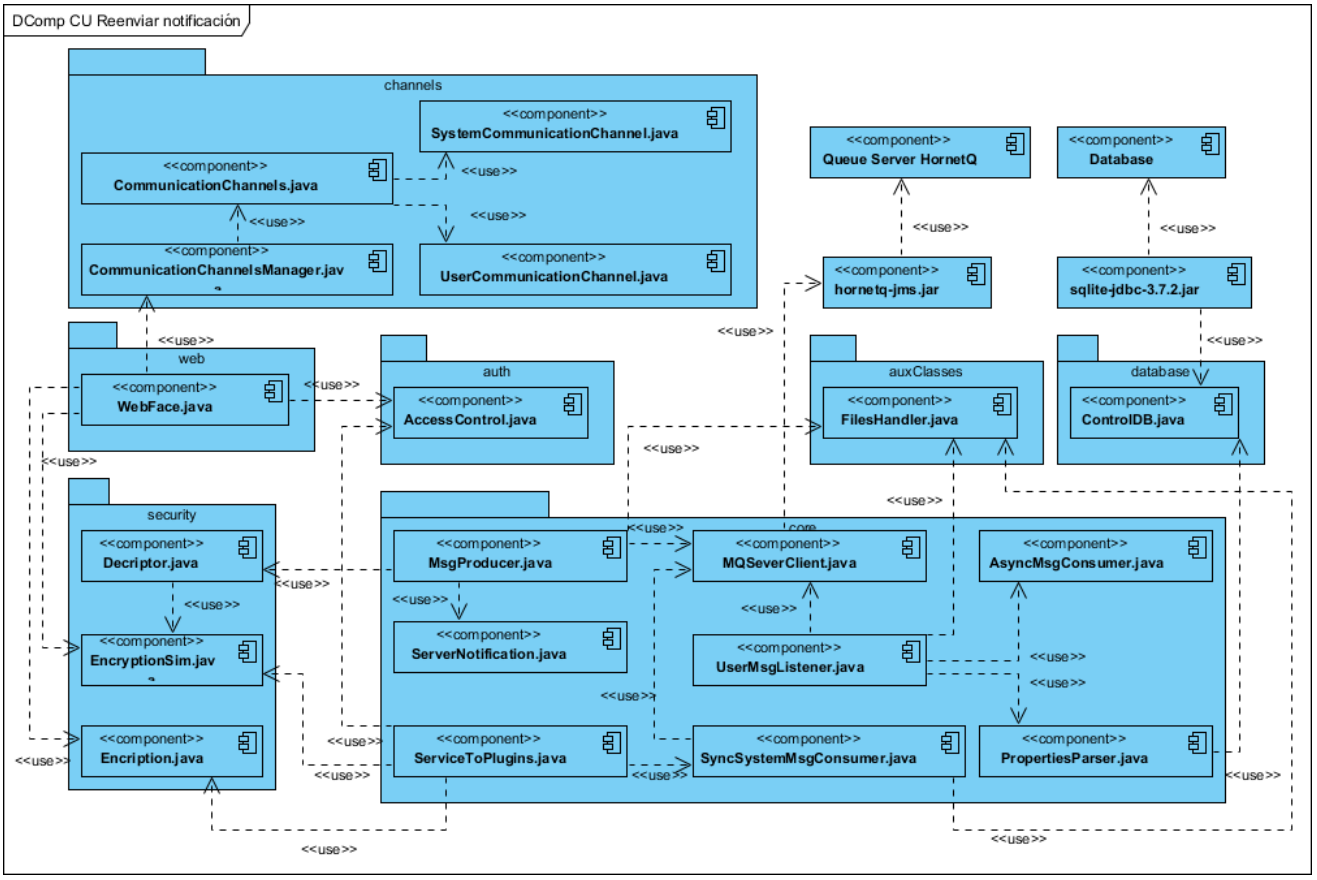

Figura 9. Diagrama de componentes del Caso de uso Reenviar notificación.

Los diagramas de componentes de los demás Casos de uso críticos se encuentran en el Anexo # 4. Los de los casos de uso secundarios se encuentran en el expediente de proyecto.

3.9. Principales funcionalidades y características del sistema.

3.9.1. Fiabilidad

El principal requerimiento planteado para la creación del Servidor de notificaciones fue garantizar que los mensajes enviados siempre llegaran a su destino. Para cumplir con este requerimiento la solución encontrada fue la inclusión del servidor de colas de mensajes HornetQ y el uso del API JMS para administrarlo y encapsular el reenvío de las notificaciones dentro de transacciones. La utilización de estas transacciones es de vital importancia, ya que ante un fallo en el reenvío de los mensajes, automáticamente se procederá a realizar otros intentos según la configuración establecida por el

administrador en el servidor de notificaciones. En las colas de HornetQ se almacenan los mensajes y se procede a su eliminación tras ocurrir uno de los siguientes casos:

1. Para los mensajes cuyo destinatario es una aplicación los mensajes solo se eliminan de las colas una vez que son consumidos por la aplicación de destino y esta confirma al servidor que recibió el mensaje. Esta confirmación se realiza enviando al servidor de notificaciones el identificador único asignado al mensaje que acaba de recibir. En caso de que la aplicación de destino no confirme que recibió el mensaje este no será eliminado de la cola y estará disponible para ser consumido nuevamente.
2. Cuando los mensajes son reenviados hacia un servidor de salida, como es el caso de un correo electrónico, una vez que estos arriban al servidor de notificaciones y son almacenados en sus respectivas colas se utiliza el mecanismo de escucha de mensajes previamente iniciado que se encarga de consumir cada mensaje de su cola y enviarlo al plugin encargado de entregarlo al servidor de destino. Si la acción de entrega se realiza correctamente, el mensaje es eliminado de su cola. En caso de que la acción de entrega falle, el mensaje no será eliminado de su cola, pero se aumentará en 1 los intentos de reenvío o se moverá la notificación a la cola de mensajes no enviados si se alcanzó el máximo número de intentos de reenvío configurados para esa cola. Cuando el intento de entrega falla, el servidor de colas HornetQ se encarga de iniciar el proceso de reenvío una vez más luego de transcurrido el tiempo entre intentos de entrega configurado para la cola en donde se encuentre el mensaje. El proceso de consumo, entrega y eliminación; o actualización de la cantidad de intentos se realiza dentro de una transacción JMS por lo que no es necesario implementar funciones que actualicen la cantidad de intentos de reenvío, ni la confirmación de consumo de mensajes o de espera del tiempo entre intentos de envío ya que estos son controlados por el servidor HornetQ. El proceso anterior se ilustra en la siguiente imagen.

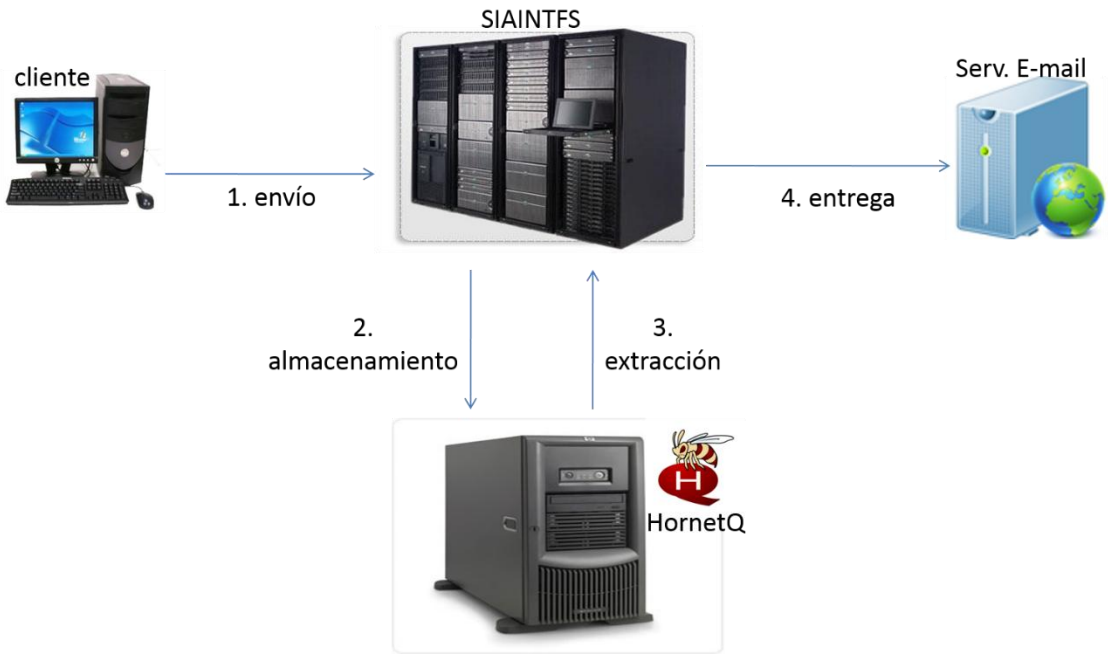

Figura 10. Proceso de recepción y entrega de notificaciones de correo**.**

3.9.2. Tipos de notificación soportados.

El servidor de notificaciones esta implementado de forma tal que es muy flexible en cuanto a las posibles formas de notificación que pueden ser agregadas; principalmente por los atributos que contienen las notificaciones que recibe de los clientes. A continuación se describen los campos que se pueden incluir.

- ✓ Identificador de la notificación.
- ✓ Cuerpo de la notificación (tipo texto).
- ✓ Identificador del emisor.
- ✓ Listado de destinatarios.
- ✓ Listado de atributos extras en la forma "llave", "valor".
- ✓ Listado de archivos adjuntos.

En el listado de atributos extras es posible incluir todos los campos necesarios para construir una notificación del tipo deseado, solo sería necesario implementar los plugins adecuados para crear y manejar la entrega de la notificación deseada a partir del tipo de notificación genérico de entrada.

3.9.3. Control de comunicación entre clientes

Para la comunicación de entre los clientes se implementó un sistema de canales de comunicación que permite definir en cual cola será almacenada una notificación llegada al servidor, a que servidor de destino será enviada la notificación una vez que se inicie su entrega y cuál será el plugin encargado de manejar dicha notificación.

Debido a la necesidad de restringir la comunicación entre determinados clientes o el uso de determinadas formas de notificación para algún cliente, durante el desarrollo del servidor de notificaciones se implementó un mecanismo mediante el cual un cliente solo podrá enviar notificaciones a través de aquellos canales que le hayan sido asignados por el administrador del servidor de notificaciones.

3.9.4. Seguridad de las notificaciones

Debido la necesidad de transmisión de las notificaciones a través de la red fue necesario utilizar un mecanismo que permitiera garantizar la seguridad de los datos transferidos. Esto se logró utilizando el algoritmo de clave simétrica AES para la encriptación del contenido de las notificaciones. El uso de un algoritmo de clave simétrica trajo consigo la necesidad de garantizar la transferencia segura de su clave secreta entre el servidor y los clientes que hagan uso del servicio; para resolver este problema se decidió encriptar la clave simétrica mediante el uso del algoritmo de clave pública RSA, lo que brinda a la solución la posibilidad de enviar la clave simétrica de manera segura a través de la red y evitar la necesidad de intercambiar dicha clave de manera manual entre cada uno de los clientes y el servidor cada vez que se decida cambiar la clave utilizada. Para garantizar que solo se envíe la clave secreta a los clientes registrados es necesario que los clientes se autentiquen mediante un nombre de usuario y contraseña cuando soliciten dicha clave. En la siguiente figura se ilustra el proceso de obtención de la clave secreta por parte de un cliente.

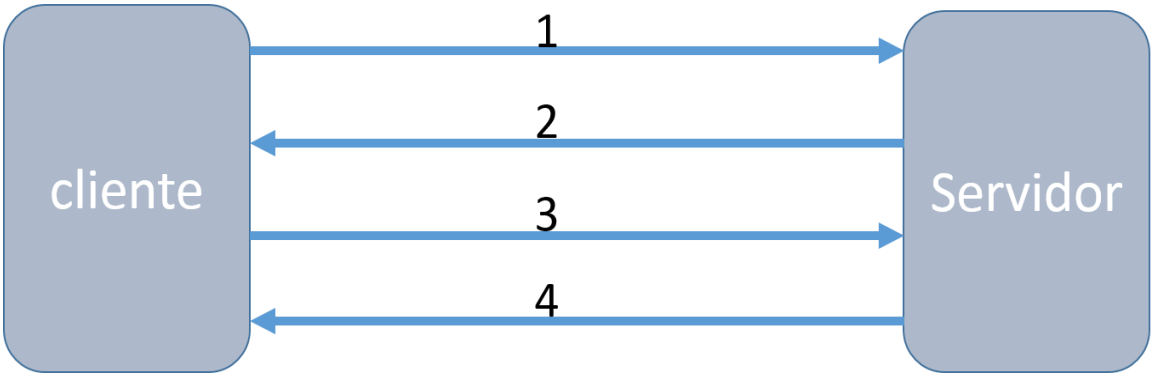

Figura 11. Proceso para obtener clave simétrica del servidor.

Descripción de las acciones:

1. El cliente solicita la clave pública del servidor.
2. El servidor envía su clave pública al cliente.
3. El cliente solicita la clave secreta al servidor, enviando su clave pública (la del cliente) además de su nombre de usuario y contraseña encriptados con la clave pública del servidor (obtenida en el paso anterior).
4. El servidor verifica las credenciales del cliente y envía la clave secreta encriptada con la clave pública del cliente (recibida en el paso anterior).

3.10. Conclusiones.

Al terminar este capítulo se puede concluir que con la definición de la arquitectura a utilizar, los patrones que rigen el desarrollo y el modelamiento de los datos persistentes se dan pasos de avance muy importantes en el desarrollo de una aplicación con calidad, que responda a los requerimientos definidos anteriormente y que pueda ser entendible para otros desarrolladores al contar con un desarrollo ordenado y estandarizado que cumpla con las buenas practicas provistas por los patrones utilizados. Además con la descripción del diagrama de despliegue y sus nodos fue posible definir políticas de seguridad para la red empresarial y organizar los componentes correctamente.

CAPÍTULO 4: PRUEBAS

4.1. Introducción.

En el presente capítulo se describe el proceso de pruebas realizadas al sistema y sus resultados. Específicamente pruebas de integración y de rendimiento ante determinados escenarios posibles durante la explotación de la herramienta, brindándose datos útiles para los clientes finales.

4.2. Pruebas del sistema.

"El único instrumento adecuado para determinar el status de la calidad de un producto software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos. En las pruebas se usan casos de prueba, especificados de forma estructurada mediante Técnicas de prueba". (27)

4.3. Pruebas de integración.

"Las Pruebas de Integración son aquellas que permiten probar en conjunto distintos subsistemas funcionales o componentes del proyecto para verificar que interactúan de manera correcta y que se ajustan a los requisitos especificados (sean estos funcionales o no). La aplicación de este tipo de pruebas en un proyecto proporciona una serie de ventajas, especialmente relacionadas con la prevención de la aparición de errores ocasionados por:

- ✓ *Un mal diseño de las interfaces de los componentes*
- ✓ *Un mal uso de los componentes"* (28)

Tabla 5. Pruebas de integración: Iteración # 1.

Módulo actual	Módulo integrado	Funcionalidad	Condiciones de ejecución	Escenarios de prueba	Resultado previsto	Resultado real
Servidor de notificaciones del SIAI (SIAINTFS)	Módulo de Carga de Datos Offline del SIAI (SIAIODL)	Iniciar entrega de notificación.	El SIAINTFS debe estar en ejecución y el SIAIODL debe realizar las solicitudes de recepción de mensajes.	Iniciar el reenvío de notificaciones al SIAIODL, habiendo mensajes disponibles para él en su cola de mensajes.	El SIAINTFS realiza las encuestas a la cola del SIAIODL y retorna una notificación para ser entregada.	Se realiza el reenvío satisfactoriamente.
Servidor de notificaciones del SIAI (SIAINTFS)	Módulo de Carga de Datos Offline del SIAI (SIAIODL)	Iniciar entrega de notificación.	El SIAINTFS debe estar en ejecución y el SIAIODL debe realizar las solicitudes de recepción de mensajes.	Iniciar el reenvío de notificaciones al SIAIODL sin que haya mensajes disponibles para él en su cola de mensajes.	El SIAINTFS realiza las encuestas a la cola del SIAIODL y retorna una notificación nula al no haber notificaciones para entregar.	Detectado un error de funcionamiento cuando la cola está vacía. Error que detiene el servicio de entrega de notificaciones desde todas las colas.

Servidor de notificaciones del SIAI (SIAINTFS)	Módulo de Carga de Datos Offline del SIAI (SIAIODL).	Recibir notificación para ser reenviada	El SIAINTFS debe estar en ejecución y el SIAIODL debe entregar al SIAINTFS la notificación para ser reenviada.	Recibir la notificación para ser reenviada	Recibir correctamente la notificación y almacenarla.	Se recibe la notificación de manera satisfactoria.

Tabla 6. Pruebas de integración: Iteración # 2

Módulo actual	Módulo integrado	Funcionalidad	Condiciones de ejecución	Escenarios de prueba	Resultado previsto	Resultado real
Servidor de notificaciones del SIAI (SIAINTFS)	Módulo de Carga de Datos Offline del SIAI (SIAIODL)	Iniciar entrega de notificación.	El SIAINTFS debe estar en ejecución y el SIAIODL debe realizar las solicitudes de recepción de mensajes.	Iniciar el reenvío de notificaciones al SIAIODL, habiendo mensajes disponibles para él en su cola de mensajes.	El SIAINTFS realiza las encuestas a la cola del SIAIODL y retorna una notificación para ser entregada.	Se realiza la entrega satisfactoriamente.

Servidor de Notificaciones del SIAI (SIAINTFS)	Módulo de Carga de Datos Offline del SIAI (SIAIODL)	Iniciar entrega de notificación.	El SIAINTFS debe estar en ejecución y el SIAIODL debe realizar las solicitudes de recepción de mensajes.	Iniciar el reenvío de notificaciones al SIAIODL sin que haya mensajes disponibles para él en su cola de mensajes.	El SIAINTFS realiza las encuestas a la cola del SIAIODL y retorna una notificación nula al no haber notificaciones para entregar.	El SIAINTFS realiza las encuestas a la cola del SIAIODL y retorna una notificación nula al no haber notificaciones para entregar.
Servidor de Notificaciones del SIAI (SIAINTFS)	Módulo de Carga de Datos Offline del SIAI (SIAIODL).	Recibir notificación para ser reenviada	El SIAINTFS debe estar en ejecución y el SIAIODL debe entregar al SIAINTFS la notificación para ser reenviada.	Recibir la notificación para ser reenviada	Recibir correctamente la notificación y almacenarla.	Se recibe la notificación de manera satisfactoria.

4.3.1. Resultados de las pruebas de integración.

En la primera iteración se detectó un error crítico que afectaba totalmente el funcionamiento del Servidor de notificaciones ya que detenía por tiempo indefinido la entrega de notificaciones a los clientes. No se detectaron errores en el servicio de recepción de las notificaciones para ser reenviadas.

En la segunda iteración se comprobó que el error encontrado en la primera iteración fue corregido y el servidor cumplió con sus funciones de manera correcta.

4.4. Pruebas de rendimiento.

Esta prueba se centró en la simulación de un entorno de trabajo donde el servidor debe recibir y entregar mensajes a aplicaciones. Las características del entorno se detallan a continuación:

Características de la máquina servidor.

- ✓ Procesador: Intel(R) Core(TM) i3-2100 CPU @ 3.10GHz (4 CPUs), ~3.1GHz
- ✓ Memoria RAM: 2 GB
- ✓ Sistema Operativo: Ubuntu 12.4 32 bits

Características de la máquina cliente.

- ✓ Procesador: Intel(R) Core(TM) 2 Duo CPU E4500 @ 2.20GHz × 2
- ✓ Memoria RAM: 1 GB
- ✓ Sistema Operativo: Ubuntu 12.10 32 bit

Red.

- ✓ Velocidad: 100 Mbps
- ✓ Tipo: LAN

Las pruebas fueron realizadas utilizando clientes implementados en lenguaje Python (uno para enviar y otro para recibir las notificaciones). Las solicitudes de reenvío o de entrega de notificación en cada una de las pruebas se realizaron de manera concurrente mediante el uso de hilos, por lo que el resultado es similar a un escenario con **n** clientes donde cada uno realice una solicitud. Fue necesario realizar las

pruebas de esta manera para calcular el tiempo total que demoró el servidor en atender todo el bloque de peticiones y enviar las respuestas correspondientes.

Por otra parte, las pruebas se centraron en el reenvío de mensajes entre aplicaciones debido a que para las notificaciones de correo electrónico, el envío del mensaje hasta el servidor de notificaciones se realiza de la misma manera que en el primer caso; además de que la velocidad de salida y concurrencia dependen del servidor de correos de destino. En tal sentido, de realizar pruebas de rendimiento para reenvío de notificaciones de correo, los resultados estarían marcados por las características de dicho servidor y no aportarían información relevante sobre el servidor de notificaciones. Los resultados obtenidos en las pruebas de envíos de mensajes hasta el servidor de notificaciones son válidos tanto para mensajes entre aplicaciones como mensajes de correo electrónico ya que los mensajes utilizados contienen toda la información necesaria para ser enviados a un destinatario de correos.

Aunque no se muestren datos estadísticos sobre el rendimiento del servidor durante el reenvío de notificaciones de correos, si se realizaron pruebas de dicha funcionalidad utilizando el servidor de correos de la UCI, simulando tres posibles escenarios:

- ✓ La conexión del servidor de notificaciones con el servidor de correo se podía establecer satisfactoriamente, pudiendo enviarse los mensajes de manera correcta.
- ✓ No existía conexión entre el servidor de notificaciones y el servidor de correo, en cuyo caso se realizaron los intentos de reenvío configurados y se descartaron los mensajes al alcanzarse el máximo número de ellos.
- ✓ No existía conexión entre el servidor de notificaciones y el servidor de correo, pero antes de alcanzarse el máximo número de intentos de reenvío configurados, se restableció la conexión, realizándose el reenvío de manera correcta.

El valor del tamaño de los mensajes mostrado en las pruebas se calculó a partir del tamaño de los archivos adjuntos contenidos en las notificaciones.

Prueba # 1.

Para una prueba de envío de mensajes al servidor como se detalla a continuación:

- ✓ Tamaño del mensaje: 182 KB

- ✓ Cantidad: 100 mensajes

Los resultados obtenidos fueron:

- ✓ Hora de inicio: 22:53:41.886
- ✓ Hora de fin: 22:53:52.498
- ✓ Tiempo total: 10 Segundos y 612 milisegundos
- ✓ Total de datos transferido: 18200 KB
- ✓ Datos transferidos por segundo: 1654,545454545455 KB/Seg (redondeando el tiempo a 11 Seg)

Para una prueba de consumo de mensajes del servidor como se detalla a continuación:

- ✓ Tamaño del mensaje: 182 KB
- ✓ Cantidad: 100 mensajes

Los resultados obtenidos fueron:

- ✓ Hora de inicio: 23:15:10.623
- ✓ Hora de fin: 23:15:18.318
- ✓ Tiempo total: 7 Segundos y 695 milisegundos
- ✓ Total de datos transferido: 18200 KB
- ✓ Datos transferidos por segundo: 2275 KB/Seg (redondeando el tiempo a 8 Seg)

Prueba # 2.

Para una prueba de envió de mensajes al servidor como se detalla a continuación:

- ✓ Tamaño del mensaje: 1 MB
- ✓ Cantidad: 50 mensajes

Los resultados obtenidos fueron:

- ✓ Hora de inicio: 17:21:55.137
- ✓ Hora de fin: 17:22:25.565

- ✓ Tiempo total: 30 Segundos y 428 milisegundos
- ✓ Total de datos transferido: 50 MB
- ✓ Datos transferidos por segundo: 1,666666666666667 MB/Seg (redondeando el tiempo a 30 Seg)

Para una prueba de consumo de mensajes del servidor como se detalla a continuación:

- ✓ Tamaño del mensaje: 1 MB
- ✓ Cantidad: 50 mensajes

Los resultados obtenidos fueron:

- ✓ Hora de inicio: 17:27:15.827
- ✓ Hora de fin: 17:27:31.009
- ✓ Tiempo total: 16 Segundos y 182 milisegundos
- ✓ Total de datos transferido: 50 MB
- ✓ Datos transferidos por segundo: 3,125 MB/Seg (redondeando el tiempo a 16 Seg)

4.5. Conclusiones.

Luego de finalizar el presente capítulo es posible llegar a la conclusión de que la realización de las pruebas constituye una muy buena guía para los usuarios finales, especialmente al relacionarse posibles escenarios y características en los cuales utilizar el servidor de notificaciones de manera segura y estable. Escenarios donde la carga de trabajo puede ser manejada por el servidor de manera eficiente justo como arrojaron los resultados de las pruebas. También es posible afirmar que el servidor cumple con las necesidades de integración con otras aplicaciones como es requerido por el cliente y su funcionamiento general es el esperado según las condiciones previstas durante las etapas del desarrollo del sistema.

CONCLUSIONES

De manera general, luego de la realización de este trabajo es posible llegar a varias conclusiones. Hasta el momento en que se desarrolla esta investigación, a pesar de existir varias herramientas que de alguna manera posibilitan el intercambio de mensajes entre aplicaciones, ninguna es lo suficientemente completa como para resolver el problema enfrentado ni adaptable como para permitir su utilización en otros escenarios, lo que incluye a la herramienta creada como resultado de este trabajo como la más útil en escenarios como el de SIAI y similares a este. La combinación de las tecnologías seleccionadas fue una decisión acertada ya que permitió la creación de una herramienta potente que cumple con los requerimientos planteados y resuelve la problemática atacada sin ser muy compleja de utilizar o desplegar por los clientes. La arquitectura basada en plugins y las notificaciones de entrada genéricas son sin dudas decisiones acertadas que aportan libertad de uso a los clientes sin necesidad de cambiar el funcionamiento del servidor ante la necesidad de agregar funcionalidades. La realización de las pruebas seleccionadas fue muy importante ya que permitió determinar problemas que atentaban contra el funcionamiento de la herramienta y que habrían sido graves si se hubieran detectado luego del despliegue en el entorno del beneficiario, así como comprobar el correcto funcionamiento de las funcionalidades implementadas.

RECOMENDACIONES

- ✓ Desplegar el servidor de notificaciones sobre sistemas operativos basados en Linux e instalar la librería “libaio“ en el servidor para aprovechar la característica de lectura/escritura asíncrona en el disco duro y mejorar el rendimiento de la aplicación.
- ✓ No utilizar el servidor de notificaciones en escenarios donde se requiera comunicación en tiempo real (ej. Transferencia de voz en una llamada telefónica).
- ✓ Desarrollar nuevas API de comunicación desde los clientes para los diferentes lenguajes de programación.
- ✓ Desarrollar *plugins* para incluir nuevas formas de notificación a los destinatarios de los mensajes.
- ✓ Incorporar un mecanismo para registrar las trazas de las operaciones que se realizan en el servidor.

REFERENCIAS BIBLIOGRÁFICAS

1. cisco.com. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012–2017. [Online] mayo 20, 2013. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html.
2. NetSupport .Inc. NetSupport Notify. [Online] [Cited: noviembre 25, 2012.] http://www.netsupportnotify.com/es/index.asp.
3. Google .Inc. C2DM. [Online] [Cited: noviembre 25, 2012.] https://developers.google.com/android/c2dm/.
4. Dell inc. dell.com. [Online] abril 14, 2013. http://i.dell.com/sites/content/shared-content/data-sheets/en/Documents/alertfind-data-sheet.pdf.
5. Pérez Barroso, Javier. Biblioteca UCI. *Sistema de Notificación de Eventos.* [Online] [Cited: noviembre 24, 2012.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_04573_11/1/TD_04573_11.pdf.
6. Lara Pérez, Yaneilis. Biblioteca UCI. *Implemetación y pruebas de un sistema de notificaciones de RHODA 2.1.* [Online] [Cited: noviembre 24, 2012.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_04652_11/1/TD_04652_11.pdf.
7. Garzón Ferrer, Lisandra Yenet. Biblioteca UCI. *Análisis y diseño del componente de notificación y mensajería para productos del área temática sistemas de apoyo a la salud.* [Online] [Cited: noviembre 24, 2012.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_1021_08/1/TD_1021_08.pdf.
8. MSDN Microsoft. [Online] [Cited: noviembre 23, 2012.] http://msdn.microsoft.com/es-es/library/19ww660c%28v=vs.80%29.aspx.
9. Oracle Corporation. [Online] [Cited: noviembre 23, 2012.] http://www.oracle.com/technetwork/java/changelog-136407.html.
10. Tech Terms. [Online] [Cited: noviembre 23, 2012.] http://www.techterms.com/definition/plugin.
11. Pousa, Adrián. [Online] mayo 10, 2013. http://postgrado.info.unlp.edu.ar/Carreras/Especializaciones/Redes_y_Seguridad/Trabajos_Finales/Pousa_Adrian.pdf.
12. Bitberry Software. Informacion sobre la encriptación. [Online] mayo 10, 2013. http://www.bitzipper.com/es/aes-encryption.html.
13. w3c.es. w3c.es. [Online] febrero 2, 2013. http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb.
14. eumed.net. eumed.net. [Online] enero 12, 2013. http://www.eumed.net/tesis-doctorales/2007/cavl/Beneficios%20de%20los%20servicios%20Web.htm.
15. Oracle Corporation. Oracle Corporation. [Online] [Cited: noviembre 23, 2012.] http://www.oracle.com/lad/technologies/java/overview/index.html.
16. Eclipse Foundation. Eclipse. [Online] [Cited: noviembre 23, 2012.] http://www.eclipse.org/.
17. JBoss Community. HornetQ. [Online] [Cited: noviembre 24, 2013.] http://www.jboss.org/hornetq.
18. Fox, Tim. *Benchmark Comparison of Messaging Throughput in Enterprise Messaging Systems using the Java Message Service AP.* mayo 10, 2013.
19. Visual Paradigm. *Visual Paradigm.* [Online] 2010. [Cited: 11 25, 2012.] http://www.visual-paradigm.com/aboutus/newsreleases/vpuml80.jsp.
20. Martin Maldonado , Daniel. aplicacionesempresariales.com. [Online] Julio 1, 2008. [Cited: Diciembre 9, 2012.] http://www.aplicacionesempresariales.com/sqlite-el-motor-de-base-de-datos-agil-y-robusto.html.
21. SQLite. SQLite. [Online] [Cited: noviembre 23, 2012.] http://www.sqlite.org/.
22. SIAI, Equipo de desarrollo del. *Especificación de requisitos de software v2.0 "Módulo Plataforma Web".* 2010.

23. CAMACHO, ERIKA, CARDESO, FABIO and NUÑEZ, GABRIEL. Páginas del Personal Académico de la USB. [Online] mayo 2, 2013. http://prof.usb.ve/lmendoza/Documentos/PS-6116/Guia%20Arquitectura%20v.2.pdf.
24. isg3.pbworks.com. [Online] mayo 2, 2013. http://isg3.pbworks.com/w/page/7624479/Patrones%20Arquitect%C3%B3nicos.
25. mmc.geofisica.unam.mx. mmc.geofisica.unam.mx. [Online] mayo 2, 2013. http://mmc.geofisica.unam.mx/LuCAS/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x219.html.
26. Sparx Systems Pty Ltd. sparxsystems.com.ar. [Online] mayo 3, 2013. http://www.sparxsystems.com.ar/resources/tutorial/uml2_componentdiagram.html.
27. pruebasdesoftware.com. pruebasdesoftware.com. [Online] mayo 3, 2013. http://pruebasdesoftware.com/laspruebasdesoftware.htm.
28. Scrum Manager. scrummanager.net. [Online] mayo 5, 2013. http://www.scrummanager.net/bok/index.php?title=Pruebas_de_integraci%C3%B3n.
29. definicion.de. definicion.de. *definicion.de.* [Online] 2008. [Cited: 12 07, 2012.] http://definicion.de/modulo/.
30. carlospes.com. carlospes.com. *carlospes.com.* [Online] 2006. [Cited: 12 07, 12.] http://www.carlospes.com/minidiccionario/modulo.php.
31. SIPEC. SIPEC. *SIPEC.* [Online] 2012. [Cited: 12 07, 2012.] http://sipec.sep.gob.mx/WebHelp/reportes/reporte.htm.
32. idef. *idef.* [Online] 2010. [Cited: 11 15, 2012.] http://www.idef.com/IDEF0.htm..
33. GNOME Project Listing. *GNOME Project Listing.* [Online] 2005-2009. [Cited: 11 15, 2012.] http://projects.gnome.org/dia/.
34. GSINNOVA. *GSINNOVA.* [Online] 2010. [Cited: 11 28, 2012.] http://www.rational.com.ar/herramientas/rup.html.
35. Osmosis Latina. *Osmosis Latina.* [Online] 12 31, 2007. [Cited: 11 28, 2012.] http://www.osmosislatina.com/lenguajes/uml/basico.htm.
36. GENBETADEV. *GENBETADEV.* [Online] Weblogs SL, 11 2004. [Cited: 11 25, 2012.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=eclipse-indigo.
37. JBoss Commuinity. *JBoss Commuinity.* [Online] 2008. [Cited: 11 25, 2012.] http://www.jboss.org/as7.
38. Seta, Leonardo De. Dos Ideas. *Dos Ideas.* [Online] 04 23, 2009. [Cited: 11 28, 2012.] http://www.dosideas.com/noticias/java/528-ejb-31-un-paso-importante-hacia-la-madurez.html.
39. Torres, Luis. Scribd. *Scribd.* [Online] 2011-2012. [Cited: 11 25, 2012.] http://es.scribd.com/doc/63846415/JPA.
40. Microsoft. *Microsoft.* [Online] 2012. [Cited: 11 25, 2012.] http://www.microsoft.com/en-us/download/details.aspx?id=1848.
41. msdn. *msdn.* [Online] 2005. [Cited: 11 25, 2012.] http://msdn.microsoft.com/es-es/library/ms225593%28v=vs.80%29.aspx.
42. Oglio, Pablo Dall. Agata. *Agata.* [Online] [Cited: 11 28, 2012.] http://agata.codigolivre.org.br/.
43. Eclipse. *Eclipse.* [Online] 2012. [Cited: 11 25, 2012.] http://www.eclipse.org/birt/phoenix/project/notable2.1.php.
44. Visconti, Marcello and Astudillo, Hernán. http://www.inf.utfsm.cl. [Online] mayo 2, 2013. http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.
45. Schauland, Derek. wiseGEEK clear answers for common questions. [Online] mayo 25, 2013. [Cited: junio 6, 2013.] http://www.wisegeek.com/what-is-a-messaging-server.htm#.
46. msdn.microsoft.com. msdn.microsoft.com. [Online] [Cited: junio 8, 2013.] http://msdn.microsoft.com/es-es/library/19ww660c(v=vs.80).aspx.

BIBLIOGRAFÍA

Bitberry Software. (10 de mayo de 2013). *Informacion sobre la encriptación*. Obtenido de http://www.bitzipper.com/es/aes-encryption.html

CAMACHO, E., CARDESO, F., & NUÑEZ, G. (2 de mayo de 2013). *Páginas del Personal Académico de la USB.* Obtenido de http://prof.usb.ve/lmendoza/Documentos/PS-6116/Guia%20Arquitectura%20v.2.pdf

carlospes.com. (2006). *carlospes.com*. Recuperado el 07 de 12 de 12, de carlospes.com: http://www.carlospes.com/minidiccionario/modulo.php

cisco.com. (20 de mayo de 2013). *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012–2017*. Obtenido de http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html

definicion.de. (2008). *definicion.de*. Recuperado el 07 de 12 de 2012, de definicion.de: http://definicion.de/modulo/

Dell inc. (14 de abril de 2013). *dell.com.* Obtenido de http://i.dell.com/sites/content/shared-content/data-sheets/en/Documents/alertfind-data-sheet.pdf

Eclipse. (2012). Recuperado el 25 de 11 de 2012, de Eclipse: http://www.eclipse.org/birt/phoenix/project/notable2.1.php

eumed.net. (12 de enero de 2013). *eumed.net*. Obtenido de http://www.eumed.net/tesis-doctorales/2007/cavl/Beneficios%20de%20los%20servicios%20Web.htm

Fox, T. (10 de mayo de 2013). Benchmark Comparison of Messaging Throughput in Enterprise Messaging Systems using the Java Message Service AP.

GENBETADEV. (11 de 2004). (Weblogs SL) Recuperado el 25 de 11 de 2012, de GENBETADEV: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=eclipse-indigo

GNOME Project Listing. (2005-2009). Recuperado el 15 de 11 de 2012, de GNOME Project Listing: http://projects.gnome.org/dia/

GSINNOVA. (2010). Recuperado el 28 de 11 de 2012, de GSINNOVA: http://www.rational.com.ar/herramientas/rup.html

idef. (2010). Recuperado el 15 de 11 de 2012, de idef: http://www.idef.com/IDEF0.htm.

isg3.pbworks.com. (2 de mayo de 2013). Obtenido de http://isg3.pbworks.com/w/page/7624479/Patrones%20Arquitect%C3%B3nicos

JBoss Commuinity. (2008). Recuperado el 25 de 11 de 2012, de JBoss Commuinity: http://www.jboss.org/as7

Martin Maldonado , D. (1 de Julio de 2008). *aplicacionesempresariales.com*. Recuperado el 9 de Diciembre de 2012, de http://www.aplicacionesempresariales.com/sqlite-el-motor-de-base-de-datos-agil-y-robusto.html

Microsoft. (2012). Recuperado el 25 de 11 de 2012, de Microsoft: http://www.microsoft.com/en-us/download/details.aspx?id=1848

msdn. (2005). Recuperado el 25 de 11 de 2012, de msdn: http://msdn.microsoft.com/es-es/library/ms225593%28v=vs.80%29.aspx

Oglio, P. D. (s.f.). *Agata*. Recuperado el 28 de 11 de 2012, de Agata: http://agata.codigolivre.org.br/

Osmosis Latina. (31 de 12 de 2007). Recuperado el 28 de 11 de 2012, de Osmosis Latina: http://www.osmosislatina.com/lenguajes/uml/basico.htm

Pousa, A. (10 de mayo de 2013). Obtenido de http://postgrado.info.unlp.edu.ar/Carreras/Especializaciones/Redes_y_Seguridad/Trabajos_Finales/Pousa_Adrian.pdf

pruebasdesoftware.com. (3 de mayo de 2013). *pruebasdesoftware.com*. Obtenido de http://pruebasdesoftware.com/laspruebasdesoftware.htm

Scrum Manager. (5 de mayo de 2013). *scrummanager.net*. Obtenido de http://www.scrummanager.net/bok/index.php?title=Pruebas_de_integraci%C3%B3n

Seta, L. D. (23 de 04 de 2009). *Dos Ideas*. Recuperado el 28 de 11 de 2012, de Dos Ideas: http://www.dosideas.com/noticias/java/528-ejb-31-un-paso-importante-hacia-la-madurez.html

SIPEC. (2012). *SIPEC*. Recuperado el 07 de 12 de 2012, de SIPEC: http://sipec.sep.gob.mx/WebHelp/reportes/reporte.htm

Sparx Systems Pty Ltd. (3 de mayo de 2013). *sparxsystems.com.ar*. Obtenido de http://www.sparxsystems.com.ar/resources/tutorial/uml2_componentdiagram.html

Torres, L. (2011-2012). *Scribd*. Recuperado el 25 de 11 de 2012, de Scribd: http://es.scribd.com/doc/63846415/JPA

Visconti, M., & Astudillo, H. (2 de mayo de 2013). *http://www.inf.utfsm.cl.* Obtenido de http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf

Visual Paradigm. (2010). Recuperado el 25 de 11 de 2012, de Visual Paradigm: http://www.visual-paradigm.com/aboutus/newsreleases/vpuml80.jsp

w3c.es. (2 de febrero de 2013). *w3c.es*. Obtenido de http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb

ANEXOS

Anexo 1.

Tabla 7. Descripción del caso de uso Almacenar notificación.

Caso de Uso:	Almacenar notificación
Actores:	Aplicación(Inicia)
Resumen:	El caso de uso se inicia cuando un cliente envía una notificación al servidor para ser reenviada.
Precondiciones:	
Referencias	RF 14.1
Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Almacenar notificación"	
Acción del Actor	**Respuesta del Sistema**
1. Invoca el servicio web encargado de reenviar una notificación especificando los siguientes datos. • Notificación de entrada. • Identificador del canal de comunicaciones que se utilizará	2. Desencripta la notificación de entrada.
	3. Envía la notificación a su cola correspondiente en el servidor de colas (especificada en el canal de comunicaciones a utilizar).

	4. Confirma al cliente que su notificación fue aceptada enviándole como respuesta el identificador de la misma.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1No envía la notificación a su cola correspondiente debido a un error.
	4.1 Envía al cliente el mensaje “ntfs-server_error-ans” indicando que ocurrió un error al procesar la notificación.
Poscondiciones	Se almacena una notificación en una cola.

Tabla 8. Descripción del caso de uso Extraer notificación.

Caso de Uso:	Extraer notificación
Actores:	Aplicación(Inicia)
Resumen:	El caso de uso se inicia cuando un cliente solicita recibir una notificación que le ha sido enviada previamente.
Precondiciones:	
Referencias	RF 14.2
Prioridad	Crítico
Flujo Normal de Eventos	
Sección “Extraer notificación”	

Acción del Actor	Respuesta del Sistema
1. Invoca el servicio web encargado de encuestar la cola del cliente y entregar una notificación.	2. Encuesta la cola del cliente y selecciona un mensaje.
	3. Bloquea el mensaje seleccionado para evitar que se realice una entrega duplicada.
	4. Extrae la notificación contenida en el mensaje seleccionado y la retorna.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	2.1 No selecciona ningún mensaje por no haber mensajes en la cola.
	4.1 Retorna un valor null por no haber mensajes en la cola.
Poscondiciones	Se extrae una notificación de una cola.

Tabla 9. Descripción del caso de uso Iniciar entrega de notificación.

Caso de Uso:	Iniciar entrega de notificación
Actores:	Aplicación(Inicia)
Resumen:	El caso de uso se inicia cuando una aplicación necesita recibir una notificación que le haya sido enviada previamente.
Precondiciones:	Autenticarse en el sistema

Referencias	RF 15
Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Iniciar entrega de notificación"	
Acción del Actor	**Respuesta del Sistema**
1. Envía una solicitud de recepción de notificación al Servidor de notificaciones mediante la invocación del servicio web correspondiente.	2. Determina el nombre de la cola a partir del nombre de la aplicación que realiza la petición.
	3. Extrae la Notificación de la cola y la envía como respuesta a la petición de la aplicación.
4. Confirma al Servidor de notificaciones la recepción de la notificación mediante la invocación del servicio web correspondiente.	5. Confirma al servidor de colas la recepción del mensaje.
Prototipo de Interfaz	
Poscondiciones	Se entrega la notificación a su destinatario.

Tabla 10. Descripción del caso de uso Autenticar aplicación.

Caso de Uso:	Autenticar aplicación
Actores:	Aplicación
Resumen:	El caso de uso se inicia ante cada solicitud de envío o recepción de mensajes al Servidor de notificaciones.
Precondiciones:	
Referencias	RF 1

<table>
<tr><td>Prioridad</td><td colspan="2">Crítico</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td colspan="3">Sección “Autenticar Aplicación”</td></tr>
<tr><td colspan="2">Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td colspan="2">1. Envía una petición para realizar alguna acción en el Servidor de notificaciones mediante la invocación del servicio web correspondiente.
Introduce los siguientes datos:
• Usuario.
• Contraseña.</td><td>2. Verifica la valides de los datos introducidos</td></tr>
<tr><td colspan="2"></td><td>3. Realiza la acción solicitada.</td></tr>
<tr><td>Poscondiciones</td><td colspan="2">Se realiza la acción Solicitada.</td></tr>
</table>

Anexo # 2.

Diagramas de clases de diseño de los Casos de uso críticos.

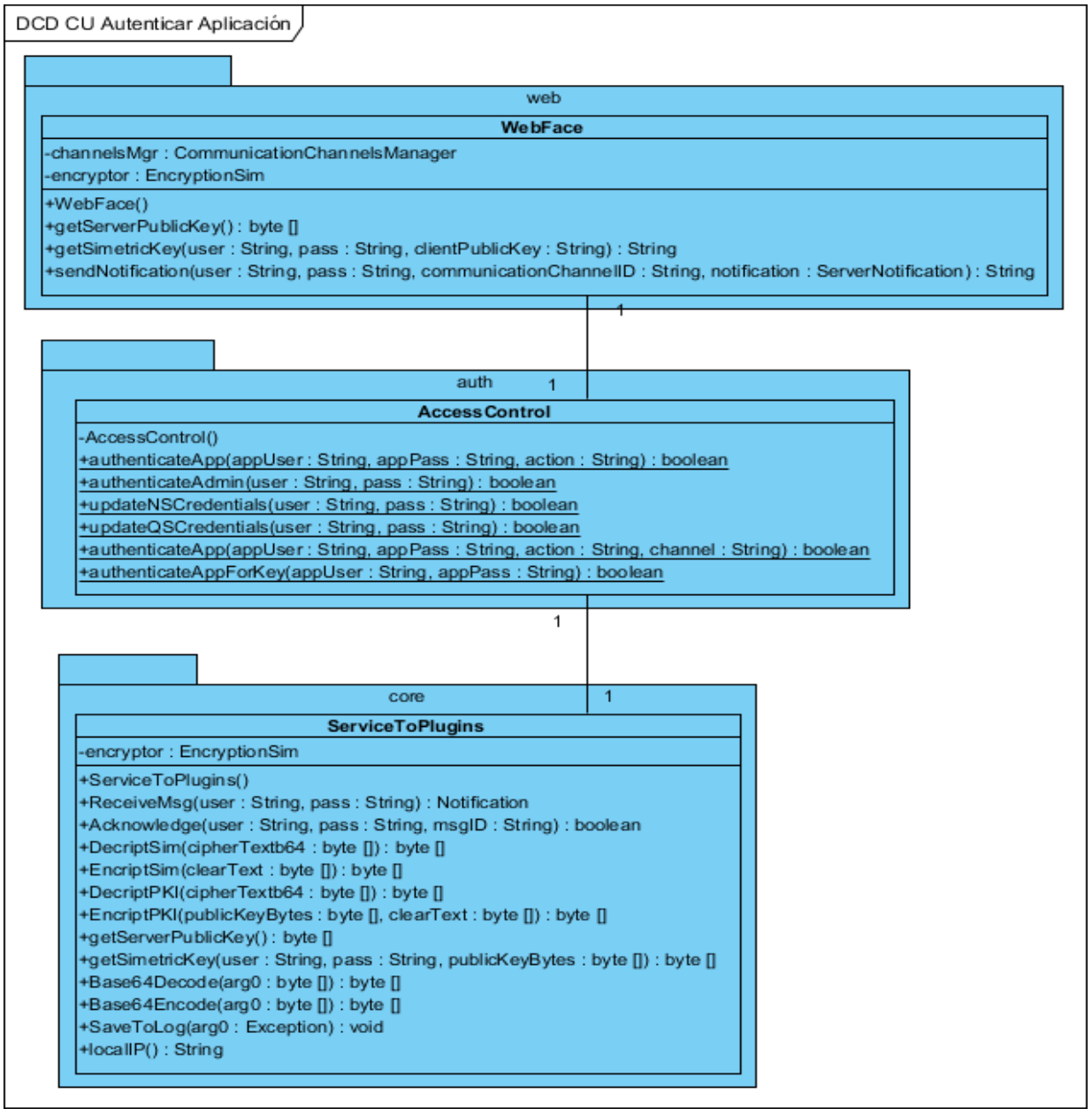

Figura 12. Diagrama de clases del diseño del Caso de uso **Autenticar aplicación**.

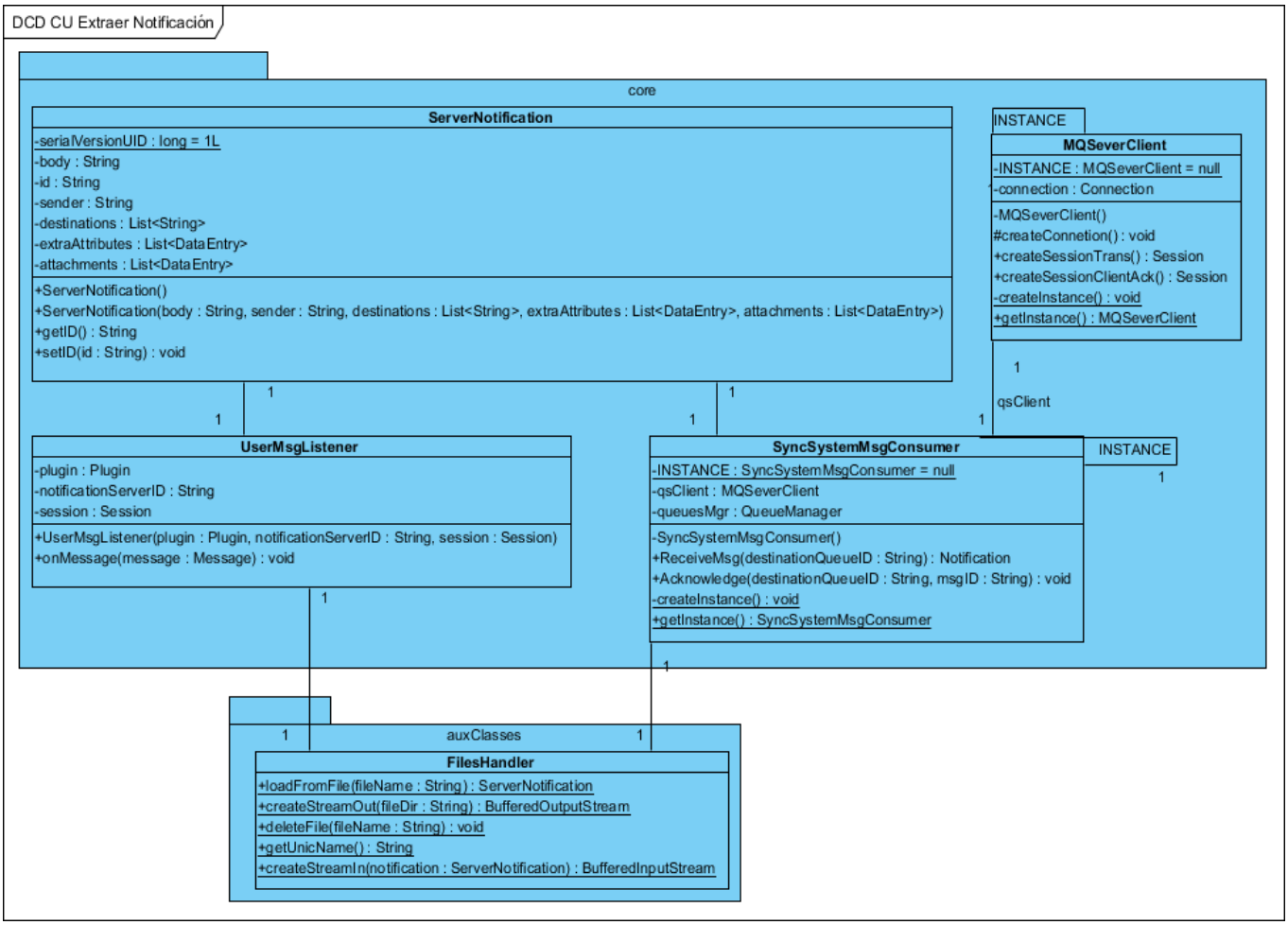

Figura 13. Diagrama de clases del diseño del Caso de uso **Extraer notificación**.

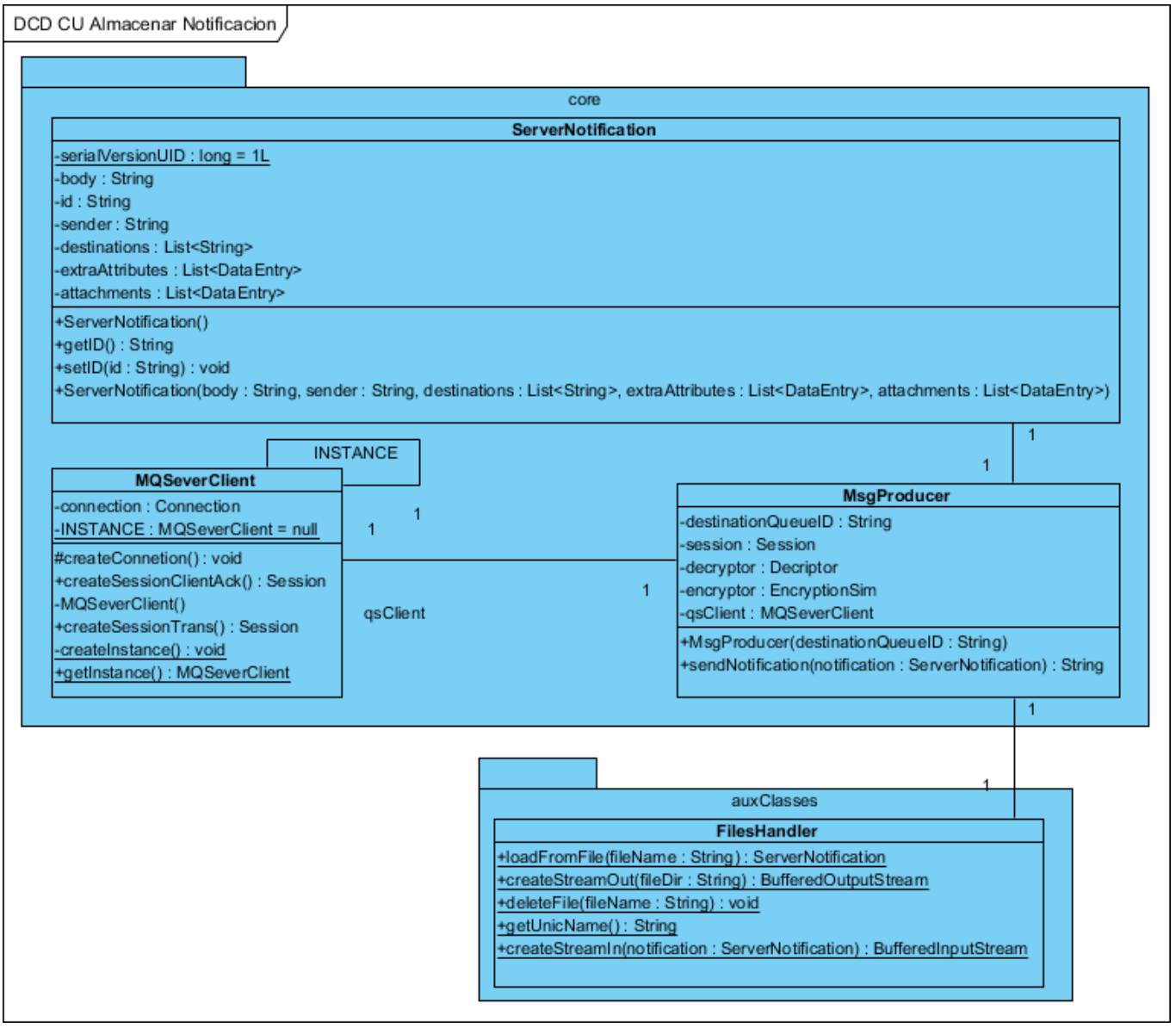

Figura 14. Diagrama de clases del diseño del Caso de uso Almacenar notificación.

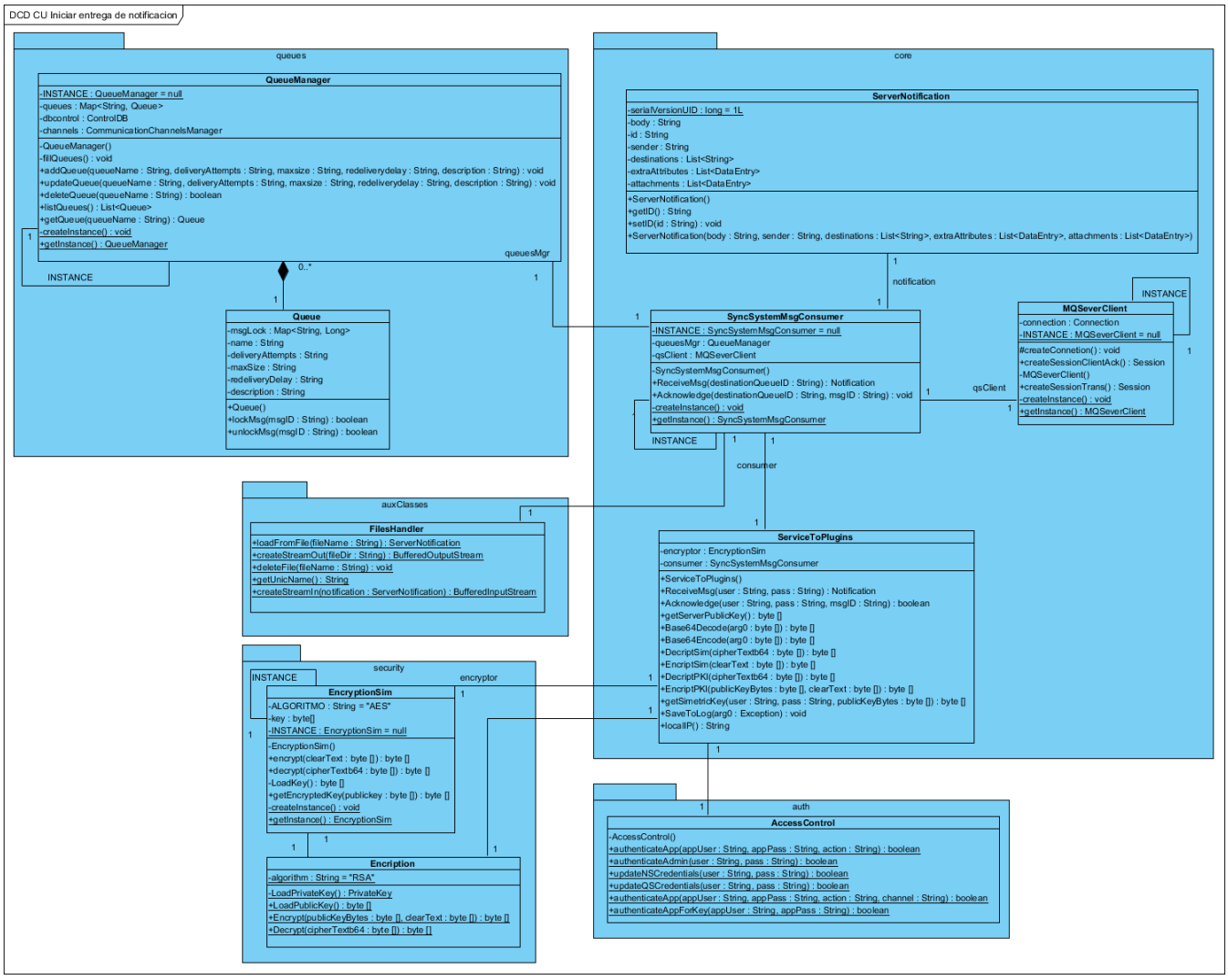

Figura 15.Diagrama de clases del diseño del Caso de uso **Iniciar entrega de notificación**.

Anexo # 3

Descripción de las tablas del modelo físico de datos.

Tabla NotificationServer

Propósito: Esta tabla está destinada a guardar la información del Servidor de Notificaciones en las columnas como se explica a continuación.

- AdminUser: nombre de usuario para administrar el Servidor de Notificaciones, por defecto “administrador”.
- AdminPass: contraseña para administrar el Servidor de Notificaciones, por defecto “administrador”.

Tabla Applications

Propósito: Esta tabla contiene la información de las aplicaciones que utilizarán el Servidor de Notificaciones, en las columnas como se explica a continuación.

- ID: identificador de la aplicación con el cual se conectara al Servidor de Notificaciones.
- Pass: contraseña con la cual la aplicación se autenticara en el Servidor de Notificaciones.
- Permissions: permisos de envío (s), recepción (r), o ambos (b) de mensajes de la aplicación en el servidor.
- Description: descripción de la aplicación en cuestión.

Tabla UserCommunicationChannels

Propósito: Esta tabla contiene la información de los canales de comunicación de usuarios, en las columnas como se explica a continuación.

- ID: identificador del canal de comunicación.
- PluginID: identificador del Plugin asociado.
- State: estado del canal de comunicación, que puede ser “active” o “inactive”.
- Description: descripción del canal de comunicación.
- QueueID: identificador de la cola de mensajes asociada.
- OutgoingServerID: identificador del perfil de servidor de salida que utilizara se utilizara para reenviar el mensaje.

Tabla OutGoingServer

Propósito: Esta tabla contiene la información de los servidores de salida configurados en el Servidor de Notificaciones, en las columnas como se explica a continuación.

- ID: identificador del servidor.

- Description: descripción del servidor.

Tabla OutGoingServerproperties

Propósito: Esta tabla contiene la información de las propiedades de conexión a los servidores de salida, en las columnas como se explica a continuación.

- PropertyName: nombre de la propiedad de conexión.
- PropertyValue: valor de la propiedad de conexión.
- OutGoingServerID: identificador del servidor de salida al cual pertenece la propiedad.

Tabla Queue

Propósito: Esta es la tabla que contiene la información de las colas de mensajes existentes en el Servidor de Colas de mensajes, en las columnas como se explica a continuación.

- ID: identificador de la cola de mensajes.
- Description: descripción de la cola de mensajes.

Tabla SystemCommunicationChannel

Propósito: Esta tabla contiene la información de los canales de comunicación de sistema, en las columnas como se explica a continuación.

- ID: identificador del canal.
- State: estado del canal, "active" o “inactive”.
- Description: descripción del canal.
- QueueID: identificador de la cola asociada.

Anexo # 4

Diagramas de componentes de los Casos de uso críticos.

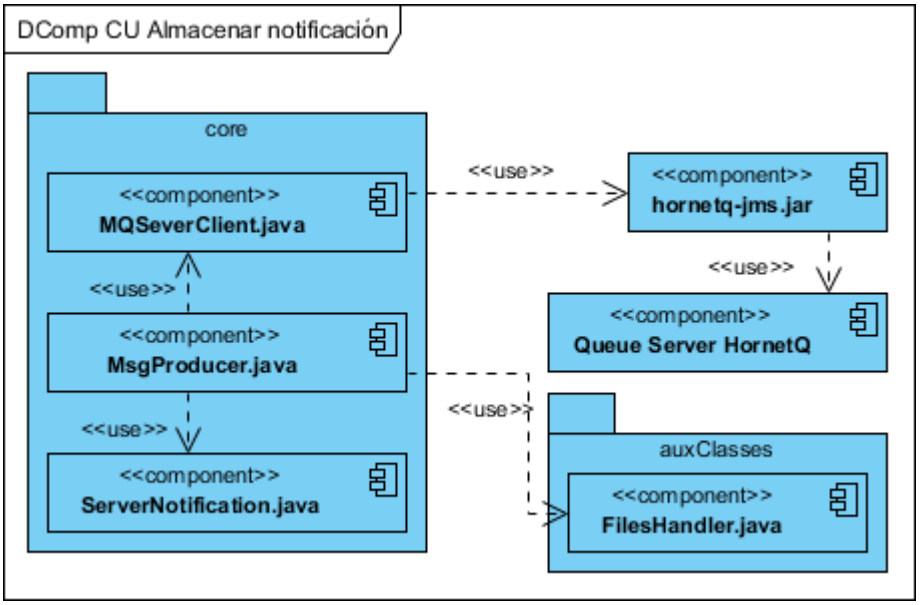

Figura 16. Diagrama de componentes del Caso de uso **Almacenar notificación**.

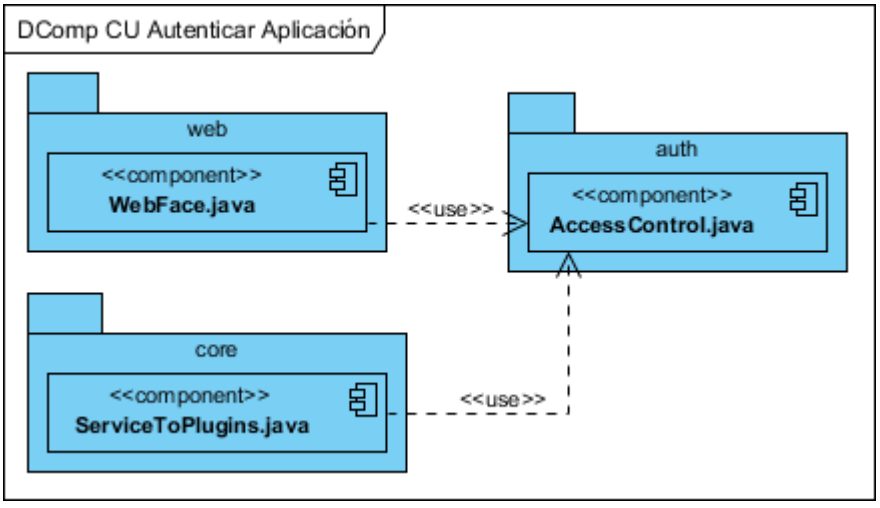

Figura 17. Diagrama componentes del Caso de uso **Autenticar aplicación**.

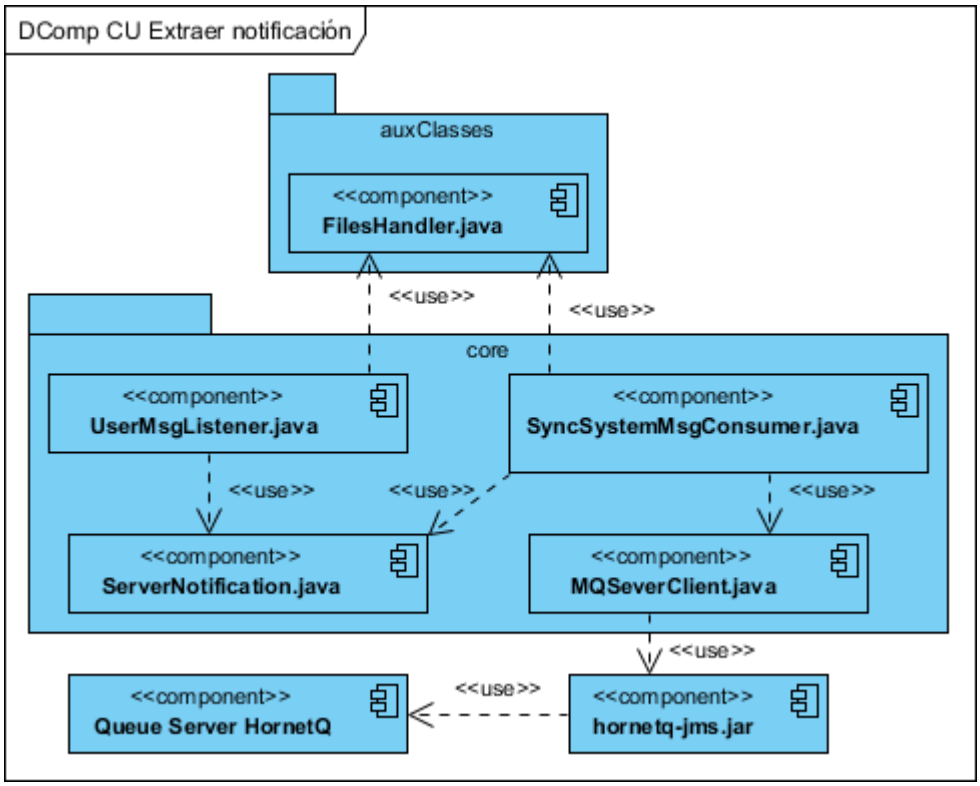

Figura 18. Diagrama componentes del Caso de uso **Extraer notificación**.

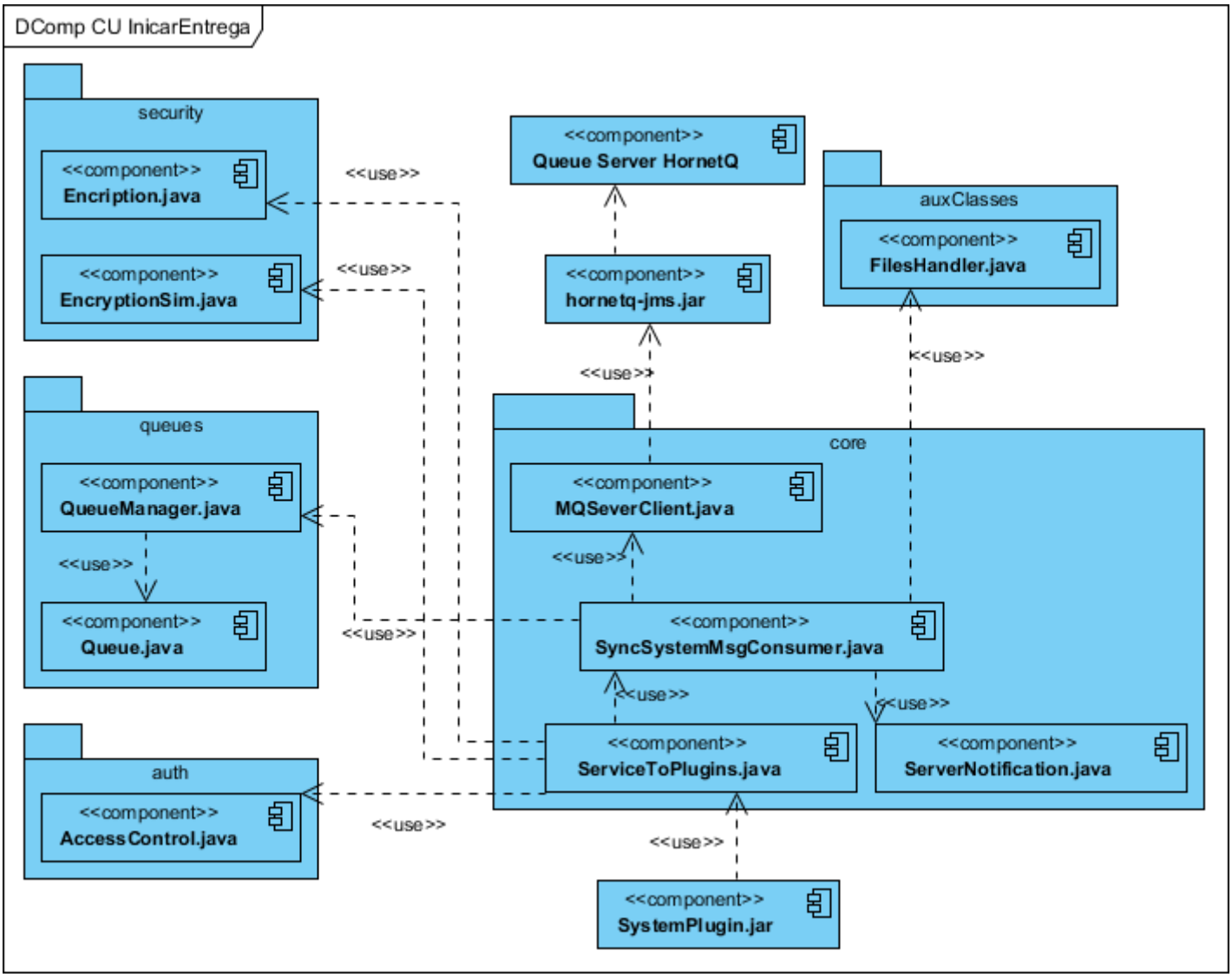

Figura 19. Diagrama componentes del Caso de uso **Iniciar entrega de notificación**.

Anexo # 5 Aval del cliente.

La Habana, 27 de mayo del 2013
"Año 55 de la Revolución"

Aval Proyecto "Módulo de Notificaciones del SIAI"

Por este medio se hace conocer la importancia del proyecto llevado a cabo por el estudiante Denis Henry Cruz Figueredo para la Dirección de Operaciones de Seguridad (DOPS) de ETECSA.

La tarea de notificar a los usuarios o administradores de eventos previamente establecidos es de vital importancia en aplicaciones que cumplen el rigor de mantenerse activas 24x7. Es una de las maneras en que puede un admnistrador ser proactivo en su trabajo, o enterarse un especialista de fraude de un escenario que previó como potencial para la ocurrencia de un fraude. Por ende, la implementación de este módulo aporta robustez y fiabilidad a la plataforma SIAI.

En este trabajo, el estudiante hace gala de oficio a la hora de diseñar la arquitectura, logrando desacoplar el enrutamiento de los mensajes de notificación de las vías para ser enviados dichos mensajes. Por lo tanto, aunque ahora solo se probó el correo electrónico, deja abierta la posibilidad de aumentar estas vías (*SMS*, *MMS*, *Blackberry*, etc) mediante el uso de *plug-ins*, concepto nunca mejor utilizado como en este trabajo.

Las pruebas de carga realizadas al módulo fueron satisfactorias y tuvo en cuenta todos los escenarios probables para hacerlo lo más tolerante a fallos posible. Estamos ante un buenísimo trabajo de arquitectura de software, muy bien implementado además. De destacar, los *plug-ins*, una idea que permitirá extender y escalar el módulo con poco esfuerzo de programación y mucha velocidad.

Sin más,

MSc. Jorge Luis Olmedo Flores
Especialista B en Telemática (EP)
Grupo de Desarrollo
Dpto. Soporte e Infraestructura
DOPS