Universidad de las Ciencias Informáticas

Facultad # 10

Título: **Implementación de un plugin para Alfresco que permita la firma digital de documentos.**

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor(es): Rafael Espinosa Añel
Yucely Ferrer Placeres

Tutor(es): Ing. Evelio Maikel Medina Manrique

Mayo, 2009

Sólo los débiles hacen copias de seguridad en cintas, los hombres de verdad suben sus cosas importantes a un ftp y permiten que el resto del mundo haga un mirror.

Linus Torvalds

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmo la presente a los 11 días del mes de mayo del año 2009.

Yucely Ferrer Placeres.
Autor.

Rafael Espinosa Añel
Autor.

Ing. Evelio Maikel Medina Manrique
Tutor.

AGRADECIMIENTOS

A la Universidad de las Ciencias Informáticas, por habernos formado como profesionales a la altura de estos tiempos. En especial a la Facultad 10 por mostrarnos el maravilloso mundo del Software Libre.

Yucely:

Le agradezco a mi mamá y a mi papá por haberme dado la vida y por hacer de mi una persona de bien, brindándome en cada momento el apoyo necesario acompañado de un inmenso amor y cariño.

A mi hermano Rafaelito por siempre hacerme feliz y confiar en mí.

A mis tíos y primos por depositar en mí toda su confianza y ayudarme en todo.

A Sisledys mi novio por su amor y comprensión en cada momento que he pasado a su lado.

A mis compañeros de grupo por facilitarme momentos de alegría y tristeza que jamás olvidaré durante los 5 años de la carrera.

A todos mis profesores, en especial a Susel y a Loania por brindarme su ayuda y amistad en todo momento.

A toda mi familia por estar a mi lado y a mis amistades por confiar en mí.

En fin gracias a todos los que de una forma u otra ayudaron en mi formación.

Rafael:

Le agradezco a mi familia por apoyarme en todo momento. A mi novia por estar siempre pendiente de mi trabajo.

A los muchos amigos que en las buenas y malas compartieron conmigo.

Al Grupo de Desarrollo del proyecto Gestión Documental y Archivística por su incondicional apoyo y constancia en la realización de este trabajo. En especial a Michel David Suarez.

A mis compañeros de cinco años de estudio y esfuerzo, por brindarme su amistad desinteresada, y compartir tantas cosas buenas y malas, que perdurarán en mi recuerdo.

A todos los profes que nunca me fallaron en estos 5 años.

A los que confiaron y creyeron en mí.

A todos gracias.

DEDICATORIA

Yucely:

Dedico este trabajo a toda mi familia, en especial a mis padres por ayudarme a hacer realidad mi sueño de convertirme en profesional y por todo lo que significan para mí. A Rafi por ser mi hermano y mi amigo. A mima por ser mi guía y a mis abuelos que ya no están por haberme querido tanto. A mis tíos y primos por ser mis segundos padres y hermanos. A mi novio por quererme tanto.

En fin a todos los que confiaron en mí.

Rafael:

Dedico este trabajo a mis abuelos porque me enseñaron que no hay recompensa sin sacrificio. A mis padres, por darme la vida y estar allí siempre para mí. A mi hermanito que tanto quiero. A mis tíos por todos los consejos y la ayuda que siempre me han brindado en especial a Xiomy por ser para mí como una madre. A mi novia. A la familia de mi novia. A mis amigos por la confianza depositada en mí.

RESUMEN

Este trabajo tiene como objetivo el desarrollo de un plugin para Alfresco que permita la firma digital de documentos. Para ello se parte de la definición conceptual de los elementos que permitan la mejor comprensión del término: firma digital y la necesidad de desarrollarla de una manera estratégica y coherente en las aplicaciones de gestión documental y archivística, pasando por un estudio de las características más usadas para la seguridad de la información, las cuales brindaron su aporte para la creación del plugin. El desarrollo de dicho plugin se realiza a partir del proceso que se lleva a cabo para lograr la autenticidad y veracidad de la información gestionada por Alfresco. A partir de estas consideraciones se muestran los resultados, logrando el objetivo propuesto.

PALABRAS CLAVES

Alfresco

Firma digital

Plugin

Autenticidad

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Con el uso de Internet la publicación de obras digitales ha tomado auge, la posibilidad de acceder libremente a la información trae consigo que las mismas sean utilizadas de manera irracional. El riesgo de ser modificadas y adaptadas de forma no autorizada es una preocupación por parte de autores y editores, los cuales temen que sus trabajos pierdan la integridad y autenticidad con que fueron creados.

Ahora bien, la solución para lograr la seguridad de la información, consiste en tomar parte de la criptografía y hacer uso de ella para crear lo que se conoce como una firma electrónica o digital con la finalidad de proteger toda la información, lograr además que las personas finalmente reciban la obra creada por el autor y tengan la certeza de disfrutarla tal y como lo dispuso su creador, dando cumplimiento con el fin último de la propiedad intelectual: aumentar y promover el conocimiento.

En cuanto al por qué se cree que la firma digital es la mejor solución para conservar la autenticidad de las obras o documentos, es preciso recordar que el autor puede darse a conocer como creador de la obra mediante su nombre, firma o cualquier forma denominativa que lo identifique como tal. La firma va a realizar una doble función: la de permitir la identificación del autor confiriendo a la obra su legitimidad y, la de implicar la voluntad del autor de apropiarse del contenido de la obra que firma; de igual manera, se debe recordar que la firma u otra marca personal utilizada de forma regular, constituyen un medio para manifestar el consentimiento del autor hacia la obra que firma.

Por las razones antes expuestas parece legítimo sostener que la firma digital, al cumplir con las finalidades de identificación y seguridad, establece una marca personal que permite identificar a una persona y asociarla con el contenido del documento que está firmando.

En fin, se han tratado algunas utilidades de la firma digital y el por qué es necesario crearla, este trabajo se encarga de la seguridad de documentos digitales gestionados por Alfresco como sistema de gestión documental, la realización de la firma digital está basada en la importancia y necesidad que esto provoca para los autores y para los realizadores de obras y demás documentos, para así mantener la integridad de la información, la autenticidad del origen del mensaje, el no repudio del origen y la auditabilidad.

En correspondencia con ello la **situación problémica** de la investigación es la siguiente:
Actualmente ECM (Enterprise Content Management) Alfresco, no cuenta con un soporte informático que garantice autenticidad y confidencialidad en los documentos digitales que se gestionan por parte de sus usuarios. Por lo que se hace difícil garantizar que la información sea enviada o elaborada por quien dice ser. Este trabajo surge para dar solución a la situación antes expuesta, por lo que la **pregunta de la investigación** queda formulada de la siguiente forma:

¿Cómo lograr autenticidad y confidencialidad en los documentos digitales gestionados por Alfresco?

La firma digital es un tema fuertemente tratado y evaluado por las diferentes personas e instituciones que necesitan utilizarla para mejorar su seguridad documental, además, de lograr una mejor identificación y confidencialidad en las obras desarrolladas. Es por ello que el **objeto de estudio de la investigación** lo constituye la firma digital como elemento de seguridad. Mientras que el **campo de acción** se enfoca a la seguridad de la información gestionada por Alfresco.

El **objetivo general** consiste en Implementar un plugin que cumpla con todas las necesidades y requerimientos para gestionar la firma digital de documentos por Alfresco.

Para lograr un mejor desarrollo de la investigación se han trazado los **objetivos específicos** siguientes:

- Realizar un estudio que posibilite un conocimiento más amplio sobre la firma digital en los documentos.
- Seleccionar las tecnologías que más se ajusten para el desarrollo del plugin.
- Desarrollar el proceso de ingeniería de software (Modelación del negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba).

Con vistas al alcance de los objetivos propuestos se hace necesario realizar las siguientes **tareas de investigación**:

- Estudio detallado con el fin de conocer cómo se aplica la firma digital a documentos.

- Realización de una búsqueda bibliográfica detallada de trabajos previos relacionados con la temática.
- Estudio de las tendencias actuales para el desarrollo del plugin.
- Análisis de los estándares más utilizados para la firma digital de documentos.
- Selección de la metodología más adecuada para desarrollar el plugin.
- Aplicación de la metodología más adecuada para desarrollar el plugin.
- Diseño del plugin siguiendo la metodología seleccionada.
- Implementación del plugin.
- Prueba.
- Implantación oficial del plugin.

La realización de la presente investigación será un punto importante para mejorar la seguridad de documentos digitales, y así permitir que los trabajos personales o institucionales no sean duplicados, ni cambiados sin previa autorización del autor o grupo de desarrollo.

Para desarrollar las tareas antes propuestas se utilizaron los siguientes métodos:

Métodos teóricos:

- Análisis - Síntesis

Este método ha servido para analizar y comprender la teoría y documentación relacionada con el trabajo de la firma digital y la tecnología Alfresco, permitiendo extraer los elementos más importantes sobre estos temas.

Métodos Empíricos

- Observación

Se ha utilizado este método sobre otros trabajos que están estrechamente relacionados con la tecnología Alfresco o tienen elementos en común con la investigación. También se ha observado la situación real que se está investigando, permitiendo acercarse al objetivo final (implementación del plugin).

- Entrevista

Se han entrevistado personas que pueden aportar elementos significativos a la investigación, así como a su resultado final y uso.

Todos los métodos anteriormente expuestos brindaron informaciones importantes sobre el tema de investigación, permitiendo la solución y desarrollo del plugin.

Estructuración del Contenido

Para una mejor comprensión del tema, la investigación se ha estructurado en cuatro capítulos.

Capítulo 1 Fundamentación teórica, se realiza un estudio sobre la firma digital, definición e importancia, se investiga acerca del estado del arte del tema a desarrollar, su necesidad en la gestión documental, y por último una disertación de las tecnologías, y metodologías usadas.

Capítulo 2 Características del sistema, se presenta la propuesta de solución que se quiere implementar, se muestra el entorno donde se desarrolla el plugin mediante el modelo de dominio, el cual se representa mediante una herramienta. Se realiza la captura de los requerimientos que el plugin debe cumplir, llevando a cabo la representación y descripción del caso de uso del sistema.

Capítulo 3 Diseño del sistema, se lleva a cabo el diseño del plugin utilizando la metodología RUP, se describe y muestra el diagrama de clases del diseño y su diagrama de interacción, además, se realiza el diagrama de despliegue el cual describe cómo y dónde el plugin será puesto en funcionamiento.

Capítulo 4 Implementación y prueba, se realiza todo lo relacionado con los flujos de trabajo de implementación y prueba, se desarrolla el diagrama de implementación para dar una visión de cómo las clases, artefactos y otros elementos de bajo nivel, se unen para formar componentes de alto nivel así como las conexiones entre ellos y se realizan los casos de pruebas para garantizar buena calidad al software.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se lleva a cabo una investigación de la firma digital en los Sistemas de Gestión Documental, su definición e importancia, ventajas, entre otros, así como un estudio de las herramientas, lenguajes y metodologías utilizadas en la actualidad en el desarrollo de software, teniendo en cuenta que las que se utilicen deben garantizar el cumplimiento de todas las funcionalidades del sistema.

Todo esto desde un enfoque básico, sin profundizar en lo que podría ser un tema de análisis más extendido. Y por último se realiza la justificación de la propuesta de metodología para desarrollar el plugin.

1.2 Firma digital. Definición e importancia

Definición

La firma digital es una solución tecnológica que permite garantizar la *autoría* e *integridad* de los documentos digitales y la posibilidad de *demostrar* estas propiedades *ante terceros*, permitiendo que éstos gocen de características que únicamente eran propias de los documentos en formato duro. (1)

Importancia

Al existir un desarrollo cada vez mayor de las comunicaciones y una tendencia global a la agilización de trámites y operaciones, la firma digital se convierte en un aspecto importante, pues constituye el mecanismo fundamental para proporcionar seguridad y confianza en las redes abiertas, surgiendo así ante la fuerte demanda del mundo informatizado.

Dado el inminente desarrollo tecnológico que existe hoy en día, se espera que para el futuro todas las gestiones se realicen a través de Internet. Tanto es así que la firma digital es utilizada para cualquier tipo de información (texto, audio, sonido o imágenes) y su aplicación es sumamente versátil, actualmente sus beneficios se notan en los negocios, en la comercialización, en el ámbito legal y en la salud.

Por otro lado, la implementación práctica de la firma brinda al usuario la posibilidad de realizar cualquier tipo de operación disminuyendo así los costos y las complicaciones a la hora de utilizar el papel. A esto se

le puede añadir que para ello cuenta con la organización y velocidad de un sistema confiable y operable desde el hogar y el trabajo.

La repercusión de este fenómeno es tan grande que la Comunidad Europea está trabajando para lograr un ente (que se maneje con los parámetros de la firma digital) en todo el mundo, para poder realizar la operación de bolsa de un país a otro.

1.2.1 Origen de la firma digital

La firma digital comenzó a desarrollarse en los Estados Unidos (EE.UU) a partir de la década del setenta, por la National Security Agency, el National Institute for Standars and Technology y en el ámbito privado la Public Key Criptography Standars, que es un consorcio entre Apple, Microsoft, Digital, Lotus, Sun y el Massachusetts Institute of Technology. Desde su surgimiento hasta la actualidad ha evolucionado de manera vertiginosa logrando la confidencialidad de la información.

1.2.2 Funcionamiento de la firma digital

El sistema utilizado por la firma digital es la criptografía asimétrica o de clave pública, que genera a través de algoritmos dos claves, una pública y una privada que son complementarias (una para cifrar y la otra para poder descifrar), esto significa que si se firma con una se necesita la otra para desencriptar la firma.
La firma es lo que se cifra en sí, no el mensaje porque este puede ser visto por más personas, no se puede confundir la criptografía con encriptar un mensaje, por tanto de lo antes dicho se infiere que el mensaje firmado no posee digitalmente confidencialidad.

La fortaleza de este método radica en el secreto de la clave pues el proceso de cifrado es público. Esta clave debe ser conocida por el transmisor y el receptor y trasmitida por un canal seguro, de esta forma es necesaria una perfecta administración de claves para evitar que sean interceptadas por extraños.

Este tipo de método comenzó a desarrollarse en el año 1977. Al ser cifrado un mensaje con una clave solo podrá ser descifrado por la clave complementaria, pero conocer la clave pública no infiere el conocimiento de la clave privada.

1.2.3 Métodos para realizar la firma digital

En la actualidad existen varios métodos para realizar la firma digital a un documento. A continuación se relacionan los más utilizados con una breve explicación:

- El método conocido como RSA, es el más utilizado actualmente. Para que sea seguro, la longitud de sus claves (una pública y una privada) debe de ser de 1024 bits, es decir, un número de un poco más de 300 dígitos.

  El RSA es un sistema criptográfico con clave pública tanto para encriptado como para autenticación, que fue inventado por Ron Rivest, Adi Shamir y Leonard Adleman en 1977.

  El encriptado y la autenticación se producen sin compartir ninguna clave secreta: cada persona utiliza sólo las claves públicas de otros usuarios y sus propias claves privadas. Cualquiera puede enviar un mensaje encriptado o verificar un documento firmado utilizando sólo claves públicas, pero solamente alguien que posee la clave privada correcta puede desencriptar o firmar el mensaje.

- Otro método reconocido para la firma digital es el llamado DSA (Digital Signature Algorithm), que es oficialmente aceptado para las transacciones oficiales en el gobierno de Estados Unidos. Este método usa también claves del mismo tamaño que el RSA y se ha podido demostrar que es casi equivalente en seguridad a este.

- Otro método es el que utiliza curvas elípticas, este método tiene la ventaja de reducir hasta en 45 dígitos las claves, manteniendo la misma seguridad. Por lo que es más propio para ser usado donde existen recursos reducidos como Smart Cards (Tarjetas Inteligentes). Actualmente este método se ha integrado como el reemplazo oficial del DSA para el gobierno de Estados Unidos.

- Entre los posibles ataques a los métodos anteriores está la posible construcción remota de una computadora cuántica, esta podría efectuar una cantidad tan grande de cálculos al mismo tiempo que podría romper los sistemas anteriores, incluso ya existen estos algoritmos que

romperían los sistemas. Pero a la vez, ya existe otro método de firma que aún con la computación cuántica no existe algoritmo que pueda romperlo. Este sistema está basado en lattices (retículas), se conoce como NTRU (Number Theory Research Unit) y es más eficiente que el RSA.

1.2.4 Funciones de la firma digital

La firma digital constituye una herramienta para conservar la legitimidad de las obras digitalizadas, su importancia radica en lograr preservar la autenticidad e integridad de las mismas. Con el uso de la firma digital se han derivado disímiles funciones que patentizan la necesidad de su utilización, dentro de las cuales encontramos las siguientes:

- Prueba la manifestación de la voluntad informáticamente expresada.
- Verifica que el contenido del documento no ha sido alterado.
- Atribuye el documento a su autor de manera fehaciente.
- Garantiza que el autor no puede negar haber firmado el documento o mensaje. (2)

1.3 Estado del Arte

El tema de la firma digital a nivel internacional es joven pero a la vez se está desarrollando en muchos países con la intención de mejorar el intercambio y tratamiento de la información tanto secreta como pública, para tener un conocimiento más detallado la investigación abarcó muchos países del mundo y varias organizaciones internacionales.

En Estados Unidos a finales de la década del setenta, el gobierno publicó el DES (Data Encryption Standard) para realizar sus comunicaciones de datos sensibles pero no clasificados. En EE.UU es donde más avanzada está la legislación sobre la firma electrónica. La ley de referencia de la firma digital para los legisladores de los Estados Unidos es el ABA (American Bar Association), Digital Signature Guidelines, del 1 de agosto de 1996. El valor probatorio de la firma ha sido ya aceptado en Utah, primer estado en legislar sobre la firma digital. La firma digital de Utah (Digital Signature Act Utah del 27 de febrero de 1995, modificada en 1996) se basa en un criptosistema asimétrico definido como un algoritmo que proporciona una pareja de claves seguras. Sus objetivos principales son, propiciar el comercio por medio de mensajes

electrónicos fiables, minimizar la falsificación de firmas digitales y el fraude en el comercio electrónico. El 9 de noviembre del 2001 el Congreso de los EE.UU aprobó una legislación para sustituir la firma convencional por la firma digital en determinados documentos.

En México los principales antecedentes legislativos se dan a partir de 1884 en el Código de Comercio donde se trata el telégrafo como medio de comunicación. En 1928 el Código Civil hace referencia al teléfono como otro medio de comunicación; en 1990 las Leyes Bancarias incorporan los medios telemáticos; en 1992 la Ley de Protección Federal al Consumidor protege a los consumidores de las ventas a distancia y tele marketing; en 1998 las Leyes Fiscales prevén las declaraciones y pagos en formato electrónico. El 29 abril del 2000 se aprobó en México un decreto mediante el cual reformó y adicionó disposiciones al Código Civil Federal, Código Federal de Procedimientos Civiles, Código de Comercio y a la Ley Federal de Protección al Consumidor para establecer el esquema jurídico y brindar mayor certeza a las operaciones vía electrónica o digital. Ante estos antecedentes la doctrina ha definido a la firma como el signo personal distintivo que, permite informar acerca de la identidad del autor de un documento. Algunos la consideran un sello y otros un distingo entre la firma electrónica y la firma electrónica avanzada, partiendo de la base de la garantía de integridad, atribución y accesibilidad.

Argentina viene implementando en el Sector Público iniciativas relativas a la digitalización de sus circuitos administrativos y a la utilización de la firma digital para dotar de seguridad a las comunicaciones internas. Para Argentina la firma digital constituye una herramienta tecnológica que permite garantizar la autoría e integridad de los documentos digitales, posibilitando que éstos gocen de características que únicamente eran propias de los documentos en papel.

En Colombia mediante la Ley 527 del 18 de agosto de 1999 se define y reglamenta el acceso y uso de los mensajes de datos, del comercio electrónico y de las firmas digitales, y se establecen las entidades de certificación.

En Chile el 10 de junio de 1999 el presidente Eduardo Frei presentó el Decreto 81 que regula el uso de la firma digital y los documentos electrónicos en la Administración del Estado. El 9 de agosto del 2000 inició, como complemento, un proyecto de ley que regula la firma electrónica, la prestación de servicios de

certificación de estas firmas y el procedimiento voluntario de acreditación de prestadores de servicio de certificación, para su uso en actos o contratos celebrados por medio de documentos electrónicos a través de medios electrónicos de comunicación.

Ecuador para marzo del 2001 presentó un Proyecto de Ley de comercio electrónico y firmas digitales, en términos muy similares a la chilena, utilizando el modelo mexicano.

En Panamá la Ley número 43 del 31 de julio del 2001 define y regula los documentos, las firmas electrónicas y las entidades de certificación en el comercio electrónico y el intercambio de documentos electrónicos.

Perú el 9 de agosto de 1999 presentó el Proyecto de Ley que regula la contratación electrónica, el cual fue dado a conocer en Washington DC, tiene por objeto regular la utilización de la firma electrónica, otorgándole plena validez y eficacia jurídica equiparada al uso de una firma manuscrita u otra análoga que conlleve manifestación de voluntad. El cual se fundamenta en la legislación colombiana, argentina, chilena y mexicana y fue aprobado por unanimidad por la Comisión de Reforma de Códigos.

En Venezuela el 26 de abril del 2000 se presentó un Proyecto de Ley Orgánica de Tecnologías de Información, que tiene por objeto promover y desarrollar el uso intensivo de las tecnologías de información en la sociedad, y regular el régimen jurídico de la función y el servicio público del sector de tecnologías de información, con el fin de establecer la Infraestructura Nacional de Tecnologías de Información, entendiendo por ésta, el conjunto de servicios y productos del sector de tecnologías de información. No considera el comercio electrónico, pero como en el caso de Chile establece lineamientos e instituciones para regular el uso de la firma digital y los documentos electrónicos en la Administración del Estado.

España por razones de seguridad y para ofrecer mayor confianza en los usuarios y jueces sobre la firma digital, decide hacer una reforma de ley cuyo objetivo fuera equiparar la firma manuscrita a cualquier otro medio de firma que cumpliera las mismas finalidades. La Circular del Banco de España 8/88 del 14 de junio creando el reglamento del Sistema Nacional de Compensación Electrónica, se convirtió en pionera y marcó un hito para la protección y seguridad necesaria en la identificación para el acceso a la información,

al indicar que la información se cifrará, para que las entidades introduzcan un dato de autentificación con la información de cada comunicación, por lo que se le confiere a este método el mismo valor que el que posee un escrito firmado por personas con poder.

Tras el Real Decreto del 17 de septiembre de 1998, en el que se reconoció el uso de la firma electrónica, su eficacia jurídica y la prestación al público de servicios de certificación. El 21 de octubre se aprobó la Ley sobre la Firma Digital. El 17 de noviembre, durante las jornadas de Comercio Electrónico de FECEMD (Federación de Comercio Electrónico y Marketing Directo), el Secretario General de Comunicaciones anunció que casi toda la Ley sobre Firma Electrónica es aplicable directamente, pero que aún resta un pequeño porcentaje (en el que se incluye la acreditación de las entidades certificadoras) que exige un breve reglamento que se está ultimando.

Aquellos que se oponen a la regulación de la firma electrónica alegan que es precipitado e imprudente tomar medidas tan pronto, porque si la iniciativa comunitaria va por otro camino puede quedar desfasada. Y realmente sería muy peligroso crear barreras internas para el comercio electrónico.

No parece que vayan a producirse demasiados conflictos. El 26 de octubre, tras una serie de difíciles negociaciones sobre los niveles de seguridad, el Parlamento Europeo aprobó una ley comunitaria que establece un marco común para la firma electrónica, y los ministros de la CMT (Comisión del Mercado de las Telecomunicaciones) son los encargados de velar por ello.

Actualmente en España se está trabajando en el Anteproyecto de Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico (18 de enero del 2002) que regula ciertos aspectos jurídicos de los servicios de la sociedad de la información y de la contratación por vía electrónica. Incorporando novedades importantes como la necesidad de constancia registral del Nombre de Dominio, ciertas obligaciones en relación con los contenidos en Internet, régimen de responsabilidad de los prestadores de servicios de información y certificación, impulsaron a la elaboración y aplicación de Códigos de Conducta, regulación de comunicaciones comerciales y contratación por vía electrónica, solución judicial y extrajudicial de conflictos, supervisión y control, infracciones y sanciones, así como reformas que permitan la informatización de los Registros Públicos. Parece garantizado que dentro de la Unión Europea la firma

será compatible, queda mucho camino por andar, la firma electrónica, tras su implantación, evolucionará constantemente, es de desear que no solo lo haga buscando mayor seguridad, sino también cierta convergencia.

Alemania por su parte aprobó el reglamento por el Consejo de Ministros el 31 de octubre de 1997, si bien funciona para el efectivo reconocimiento del valor jurídico de la documentación informática y de las firmas digitales, será necesario esperar a que sea operativo en virtud de la emanación de los posteriores e indispensables reglamentos técnicos de actuación.

Se define la firma digital como el resultado del proceso informático (validación) basado en un sistema de claves asimétricas o dobles, una pública y una privada, que permite al suscriptor transmitir la clave privada y al destinatario transmitir la clave pública, respectivamente, para verificar la procedencia y la integridad de un documento informático o de un conjunto de documentos informáticos. En el reglamento la firma digital está basada exclusivamente en el empleo de sistemas de cifrado llamados asimétricos.

Regula el Reglamento entre otras cosas: la validez del documento informático; el documento informático sin firma digital; el documento informático con firma digital; los certificadores; los certificados; autenticación de la firma digital; los actos públicos notariales; la validación temporal; la caducidad, revocación y suspensión de las claves; la firma digital falsa; la duplicidad, copia y extractos del documento; y la transmisión del documento. Está basada dicha normativa en soluciones extranjeras y supranacionales.

En la Comunidad Europea, la Comisión Europea ha financiado numerosos proyectos cuyo objetivo es la investigación de los aspectos técnicos, legales y económicos de la firma digital. Dicha Comisión hizo público en octubre de 1997 una Comunicación al Consejo, al Parlamento, al Comité Económico y Social y al Comité de las Regiones titulada “Iniciativa Europea de Comercio Electrónico”, con un subtitulo de “Hacia un Marco Europeo para la Firma Digital y el Cifrado”. En el segundo trimestre del 1998 se empezaron a encauzar las propuestas para nuevas medidas, una de las cuales podría ser la elaboración de una Directiva de firma digital.

Lo que pretende la Comisión Europea es encontrar un reconocimiento legal común en Europa de la firma digital, con el objetivo de armonizar las diferentes legislaciones, para que ésta tenga carta de naturaleza legal ante tribunales en materia penal, civil y mercantil, a efectos de prueba, apercibimiento y autenticidad.

Así se expidió a finales de 1998 un borrador de propuesta de directiva sobre firma electrónica y servicios relacionados. Pese a la seguridad ofrecida por la firma digital, el borrador de propuesta de directiva regula la firma electrónica en general, y no solo la firma digital en particular, en un intento de abarcar otras firmas electrónicas, basadas en técnicas distintas de la criptografía asimétrica. Esta neutralidad es seguramente conveniente, para dejar abiertas las puertas a desarrollos tecnológicos futuros. Pero, por otra parte, llevada a ese extremo, deja sin resolver, porque no son siquiera abordados, muchos de los problemas planteados actualmente por las firmas digitales, únicas firmas electrónicas seguras hoy en día. Para conseguir una coherencia europea se deberá, sin duda pasar por el establecimiento de una política europea de control armónica con otras potencias económicas como EE.UU, Canadá y Japón.

La OCDE (Organización para la Cooperación y Desarrollo Económico) recomendó la utilización de la criptografía (Guidelines for Criptography Policy) que fue aprobada el 27 de marzo de 1997. Esta recomendación no tiene fuerza vinculante y señala una serie de reglas que los gobiernos deberán tener en cuenta al adoptar la legislación acerca de la firma digital y terceros de confianza, con el fin de impedir la adopción de diferentes reglas nacionales que podrían dificultar el comercio electrónico y la sociedad de la información en general.

Organización Internacional de Normas ISO (International Organization for Standardization) la Norma ISO/IEC 7498-2 (Arquitectura de Seguridad de ISO) sobre la que descansan todos los desarrollos normativos posteriores, regula los servicios de seguridad sobre confidencialidad, integridad, autenticidad, control de accesos y no repudio. A través de su subcomité 27, SC 27, se trabaja en una norma de firma digital.

A nivel nacional

La firma digital en Cuba tiene un nivel bajo, pues esta tecnología aún no se ha desarrollado como en otros países del mundo, sin embargo, en el país el desarrollo de la informática es dirigido por todos los sectores

de la sociedad, incrementándose así el uso de Internet en las comunicaciones y el comercio. Este progreso informático ha motivado a realizar búsquedas de mecanismos para garantizar la seguridad en las redes y se han realizado varios procesos para priorizar su utilización. En tal sentido el MININT desarrolló una infraestructura de Clave Pública de la cual se han implementado ya varios procesos, como la validación y autenticación de certificados digitales.

En la UCI

La firma digital en Cuba y en la UCI tiene el mismo nivel de desarrollo. Con la salvedad de que en la universidad es donde más potencial intelectual y recursos existen para investigar, desarrollar y utilizar los beneficios de esta poderosa tecnología. En la universidad se llevan a cabo varios proyectos que están prestando mucha atención al desarrollo de la firma digital como forma segura y confidencial de transmitir información.

Software de firma digital

Además del desarrollo legislativo respecto a la firma digital también se han desarrollado varios software que permiten la realización de ésta, muchos son propietarios pero gracias a una investigación en la esfera del software libre se encontraron los productos relacionados a continuación:

Sinadura: firma digital de PDF en Linux.

Las empresas especialistas en Software Libre, Irontec y Zylk han publicado y liberado el código fuente de Sinadura, una aplicación Java que permite firmar digitalmente ficheros PDF, usando por ejemplo, certificados digitales de Izenpe, FNMT o el DNIE. Es la primera aplicación de escritorio para firma de PDF bajo licencia libre que se conoce y la interfaz de la aplicación está preparada para el soporte multi-idioma.

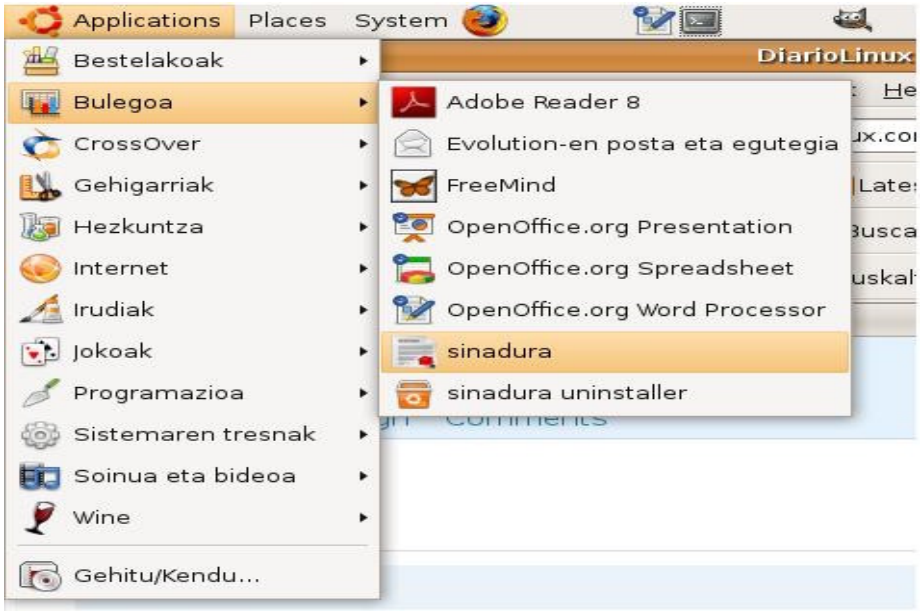


Figura 1: Software Sinadura

Viafirma se convierte en la primera plataforma software libre con soporte para el DNI electrónico. Es una plataforma de autenticación y firma digital implementada en Java. Gracias a Viafirma 1.2 ahora es mucho más fácil que otras aplicaciones hagan uso del DNI para autenticar a sus usuarios o solicitarles que autoricen o firmen documentos y transacciones.

Figura 2: Software Viafirma

EParapher

Para firma digital de archivos se puede utilizar EParapher, una herramienta que soporta casi todo tipo de archivos. Esta herramienta firma y convierte el archivo a tres posible estándares: PDF, PDF/A y XML.

Entre sus características destaca:

- Conversión y firma rápida de cualquier archivo de texto, imagen y archivos de Office u Open Office.
- Permite crear, suprimir, importar y exportar certificados y llaves privadas.
- Soporta los archivos de almacenamiento de llaves: PKCS#12, JKS, JCEKS y BKS.
- Permite utilizar certificados de Windows y Smart Cards.
- EParapher está disponible para las plataformas: Windows, MacOSX y Linux.

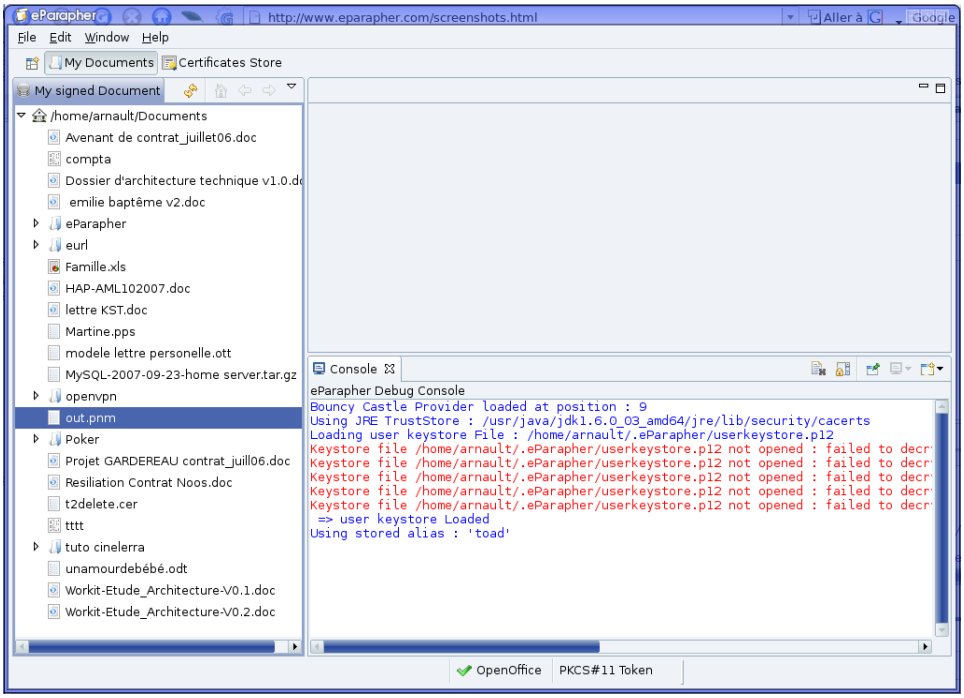


Figura 3: Software EParapher

1.4 Necesidad de la Firma Digital en la Gestión Documental y Archivística

La firma digital es necesaria en la gestión documental porque posibilita que la información gestionada cumpla con todos los elementos de seguridad, por ejemplo garantiza la **autenticidad del origen del mensaje**, aspecto que protege al receptor del documento, garantiza que el mensaje ha sido generado por el emisor identificado en el documento, no pudiendo alguna otra entidad suplantar a un usuario del sistema, otro elemento es la **integridad de la información**, esto significa que la información enviada no ha sido modificada. La integridad es una cualidad importante para otorgarle validez legal a una información. La firma digital detecta la integridad de la información que es firmada, independiente al medio de almacenamiento, además, la integridad de un documento es una forma de protección contra la modificación intencional o accidental de los datos, el **no repudio del origen** es una característica susceptible de verificación mediante un perito informático, a pesar que solo se aplica en algunos algoritmos, en caso de duda o petición de alguna de las partes, para confirmar la validez o no de la firma.

Las ventajas que ofrece la firma digital vienen dadas por su procedimiento de verificación, que es exacto e imposible de falsificar en la práctica. Puede ser realizada en diferentes puntos del mundo, de forma simultánea y sin necesidad de testigos. En un contexto electrónico, en el que no existe contacto directo entre las partes, brinda la posibilidad a los usuarios de presentar un documento digital que ofrezca las mismas funcionalidades y características de seguridad que los documentos en formato duro.

1.5 Metodologías de Desarrollo de Software

En los últimos tiempos la variedad de metodologías y procesos de desarrollo de software han aumentado de manera impresionante. Se podría decir que han surgidos dos corrientes. Las llamadas metodologías pesadas y las metodologías ágiles o ligeras. La diferencia fundamental entre ellas es que, mientras las metodologías pesadas buscan cumplir el objetivo común mediante el orden y la documentación, las ligeras tratan de mejorar la calidad del software a través de una comunicación directa e inmediata entre las personas que intervienen en el proceso.

Como base para escoger una metodología que se ajuste a las necesidades específicas del trabajo, el tipo de aplicación y la finalidad del mismo, se realizó un estudio de las principales características de dos de las metodologías más usadas en la actualidad para el desarrollo de software.

1.5.1 XP. Programación Extrema

La Programación Extrema define una manera de reunir a clientes y programadores en un equipo firmemente integrado con condiciones de trabajo que promueven la comunicación y solución de un problema. XP requiere de colaboración y disciplina. Debido a la interacción intensa, XP funciona mejor para equipos pequeños y medianos, con 15 miembros aproximadamente.

Etapas de la metodología: Planificación del proyecto, Diseño, Codificación, Pruebas.
XP define UserStories (Historias de Usuario) como base del software a desarrollar, estas historias las escribe el cliente y describen escenarios sobre el funcionamiento del programa, a partir de las historias de usuario y de la arquitectura perseguida se crea un plan de releases entre el equipo de desarrollo y el cliente.

Los análisis incluyen las estimaciones de tiempo y los recursos requeridos para terminar cada característica. La sincronización se basa en las iteraciones cortas, generalmente de una a tres semanas, durante las cuales se termina una porción pequeña pero funcional del proyecto. Cada iteración incluye el diseño, codificación, pruebas y lanzamiento, y cada lanzamiento se integra con los anteriores.

Debido a que el proyecto se construye en pasos pequeños pero funcionales, el lanzamiento final es realmente un acontecimiento. La aplicación es completa cuando tiene la funcionalidad suficiente para ser útil o apta para venderse a los clientes.

Puntos claves: Ligero, Cercano al desarrollo, Se basa en Historias de Usuario, Fuerte comunicación con el cliente, El código pertenece a todos, Programación por parejas, Pruebas como base de la funcionalidad, Pobre en cuanto a documentación.

1.5.2 Proceso Unificado del Software

RUP (Rational Unified Process) es una metodología basada en un pequeño grupo de principios claves: el equipo de un proyecto de software debe planificar el desarrollo; debe conocer hacia donde se dirige; debe documentar el proyecto de una manera perdurable y extensible.

Los verdaderos aspectos definitorios del Proceso Unificado, y que lo convierten en único, se resumen en tres características, dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental.
Un proyecto realizado con RUP se divide en cuatro fases:
Inicio, Elaboración, Construcción, Transición.

En cada fase se ejecutan una o varias iteraciones con los flujos de trabajo definidos por RUP. Modelado del negocio, Requerimientos, Análisis y diseño, Implementación, Pruebas, Despliegue, Ambiente, Gestión de proyecto, Gestión de configuración.

El proceso define una serie de roles que se distribuyen entre los miembros del equipo y que especifican las tareas de cada uno y sus resultados. Se basa en casos de uso y está muy orientado a la arquitectura

del sistema, documentándose lo mejor posible, basándose en UML (Unified Modeling Language) como lenguaje principal.

Puntos claves: Pesado, Dividido en cuatro fases, Las fases se dividen en iteraciones, Se definen flujos de trabajo, Los artefactos son el objetivo, Se basa en roles, UML, Muy organizativo, Mucha documentación.

1.6. Herramientas de desarrollo de software

En la actualidad las herramientas de desarrollo de software han aumentado de manera paulatina con el desarrollo de la informática. Existen dos corrientes principales, las herramientas privadas y las libres. La diferencia fundamental entre ellas es que, mientras los propietarios buscan el control total de sus herramientas y las ganancias que estas le pueden aportar, los libres (software libre) tratan de mejorar sus herramientas y adaptarlas a las necesidades de cada persona que interviene en el proceso, sin costos por su uso, esto no implica que la adquisición del producto sea gratis. Se realizó un estudio de las principales características de las herramientas que se utilizarán para el desarrollo del plugin.

1.6.1 Visual Paradigm

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado a objeto, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.

1.6.2 IDE NetBeans

El IDE NetBeans es un producto de código abierto, además, es un entorno de desarrollo integrado para los desarrolladores de software. Con su uso se obtienen todas las herramientas necesarias para crear profesionales de escritorio, de empresa, web y aplicaciones móviles con el lenguaje Java, C / C + +, e incluso lenguajes dinámicos como PHP, Java Script, Groovy, y Ruby. NetBeans IDE es fácil de instalar y utilizar, además, se ejecuta en muchas plataformas, incluyendo Windows, Linux, Mac OS X y Solaris.

El IDE NetBeans 6.5 ofrece varias características y mejoras, como el rico de PHP, Java Script y Ajax como son: funciones de edición, soporte mejorado para el uso de la web de Hibernate y el marco Java Persistence API, y una mayor GlassFish v3 y MySQL integración.

1.7 Lenguajes a utilizar

En la actualidad la suma de lenguajes para el desarrollo de software ha aumentado de manera sorprendente. Existen dos tipos de lenguajes de programación claramente diferenciados; los lenguajes de bajo nivel y los de alto nivel.

Los lenguajes de bajo nivel son totalmente dependientes de la máquina, es decir que el programa que se realiza con este tipo de lenguaje no se puede migrar o utilizar en otras máquinas. Al estar prácticamente diseñado a medida del hardware, aprovecha al máximo las características del mismo. Dentro de este grupo se encuentran por ejemplo el lenguaje de máquina y el ensamblador.

Los lenguajes de alto nivel son aquellos que se encuentran más cercanos al lenguaje natural que al lenguaje máquina. Están dirigidos a solucionar problemas mediante el uso de EDD's (Estructuras Dinámicas de Datos). Se trata de lenguajes independientes de la arquitectura del ordenador. Por lo que, en principio, un programa escrito en un lenguaje de alto nivel, se puede migrar de una máquina a otra sin ningún tipo de problema. Estos lenguajes permiten al programador olvidarse por completo del funcionamiento interno del tipo de máquina para la que está diseñando el programa. Tan solo necesitan un traductor que entienda el código fuente, como las características de la máquina. Suelen usar tipos de datos para la programación y existen lenguajes de propósito general (cualquier tipo de aplicación) y de propósito específico (como FORTRAN para trabajos científicos).

Por otra parte también existen los lenguajes de modelado, que como su nombre lo indica son utilizados para modelar, especificar y construir de forma gráfica cada una de las partes del software. Se realizó un estudio de las principales características de los lenguajes que se utilizarán para la solución del plugin.

1.7.1 Java. Lenguaje de programación a utilizar

Se seleccionó el lenguaje de programación Java que es orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje posee una sintaxis clara y bien definida similar a la del lenguaje de programación C y C++, pero tiene un modelo de objeto más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. Reúne capacidades funcionales para la creación del Servicio Web, también existe mucha información, documentación e infinidad de ejemplos disponibles en Internet, es de la filosofía de Software Libre, pues no hay que pagar por licencias o patentes para obtener las actualizaciones.

1.7.2 UML (Unified Modeling Language)

El Lenguaje de Modelación Unificado (UML) es un lenguaje gráfico para visualizar, especificar, construir y documentar cada una de las partes que comprende el desarrollo de software. Posee formas de modelar conceptos como por ejemplo las funciones del sistema, además de otras particularidades como la de escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusable. Usa procesos de otras metodologías, aprovechando la experiencia de sus creadores UML 03.

Es desde finales de la década del 90, un lenguaje de modelación orientado a objetos estándar, de acuerdo con el Object Management Group, siendo utilizado diariamente por grandes organizaciones como: Microsoft, Oracle, Rational, etc.

Entre sus características se destacan:

- Tecnología de orientación a objetos.
- Viabilidad en la corrección de errores.
- Desarrollo incremental e iterativo.
- Participación del cliente en todas las etapas del proyecto.

UML es un lenguaje consolidado, fácil de aprender, y permite una comunicación fluida entre los diversos actores acerca del modelo.

Las desventajas más notables de UML es que no ha sido diseñado para modelar procesos de negocio, por lo que no está orientado a lo que necesita el experto en el dominio del negocio, predispone un enfoque orientado a objeto lo que puede contradecir un enfoque orientado al negocio, suele estar más orientado a los arquitectos de sistemas y diseñadores de software y está pensado para un público eminentemente técnico.

1.8 Propuesta de Metodología para desarrollar el plugin

A partir del análisis comparativo entre estas dos metodologías y atendiendo al proyecto que se enfrenta, el hecho de que el plugin será desarrollado para un Sistema de Gestión Documental y Archivística basado en Alfresco, es necesario escoger una metodología que se ajuste a las características del trabajo y que no genere una carga innecesaria. Se propone la utilización de RUP, que actualmente es una de las metodologías más usada en el desarrollo del software, por su eficiencia y calidad obtenida a lo largo del ciclo de vida del software, traza una mejor y completa línea de trabajo, es un proceso de desarrollo de software que proporciona una guía en el orden de las actividades de un equipo, dirige las tareas individuales de los desarrolladores, especifica que productos deberán ser desarrollados y ofrece criterios para monitorear, medir los productos y actividades de un proyecto.

1.9 Conclusiones

La realización de un estudio sobre la Firma Digital en los Sistemas de Gestión Documental y Archivística, ayudó a la comprensión de las ventajas que proporcionaría contar con un plugin para la firma digital de documentos en Alfresco, que garantice la total seguridad de la información relativa a sus usuarios. Un análisis de las tendencias y tecnologías dieron la comprensión necesaria para emprender con el desarrollo de la propuesta de solución.

Como metodología para el desarrollo del plugin se propone RUP, una metodología de las llamadas pesada que se centra en obtener al final del ciclo de vida del proyecto un producto documentado e implementado con calidad, como herramienta para realizar los distintos artefactos ingenieriles se eligió el Visual Paradigm en su versión 3.0, totalmente compatible con la tradicional en el entorno universitario, Rational Rose. El lenguaje de modelado será el potente y versátil UML, lenguaje gráfico para visualizar,

especificar, construir y documentar cada una de las partes que comprende el desarrollo de software. Por último, para desarrollar el plugin se escogió el IDE NetBeans, que soporta todo tipo de aplicación Java.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En este capítulo se realiza una descripción del entorno donde surge la necesidad del plugin y la propuesta de solución. Además, se representan los principales conceptos y los tipos más importantes de objetos en el contexto del sistema, un glosario con los términos utilizados para facilitar su comprensión y la captación de los requerimientos del plugin.

2.2 Situación Problémica y Problema

La gestión documental se ha convertido en una necesidad imprescindible para las empresas, organismos e instituciones. Las herramientas que se utilizan hoy para el desarrollo de esta actividad tienen sus necesidades representadas en la economía (gastos en locales, infraestructuras para garantizar la conservación) y en la duración del tiempo de trabajo para realizar la búsqueda y organización de la documentación, la fundamental es que hasta ahora no todas garantizan el acceso seguro, la autenticidad e integridad de los documentos almacenados, aquí es donde los certificados digitales tienen mucho que decir y que aportar a este tipo de soluciones.

El proyecto Gestión Documental y Archivística, tiene como objetivo brindar la posibilidad de mantener disponible toda la información gestionada y atribuirle a ésta características de seguridad imprescindibles a lo largo del tiempo. Para su desarrollo no cuenta con una herramienta que brinde la posibilidad de firmar y validar documentos. Existe necesidad de una herramienta que posibilite el no repudio en la transformación del formato duro al electrónico y su posterior almacenamiento generando evidencias legales.

2.3 Propuesta de sistema

Se propone como solución la implementación de un servicio web, el cual pretende satisfacer la necesidad de la firma digital para el Sistema de Gestión Documental y Archivística.

Se trata de un producto exportable que se apoya en el servidor Ldap para la generación de certificados digitales, firma todo tipo de documento, avala la total autenticidad, auditabilidad y seguridad de los documentos digitales y garantiza que los documentos firmados tengan una validez de larga duración. Para ello se utilizan estándares que ofrecen una total validez a lo largo del tiempo.

En fin se propone un servicio web porque la integración de las aplicaciones y los servicios web se han convertido en los últimos años en términos ampliamente conocidos por las empresas. En ese escenario, y debido al auge que ha alcanzado la arquitectura orientada a servicios (SOA, Services Oriented Architecture), los estándares se han convertido en elementos indispensables para facilitar el desarrollo de funcionalidades web sin limitaciones. Este servicio web aporta gran independencia entre el cliente y el servicio. De esta forma, los cambios a lo largo del tiempo en uno no deben afectar al otro. Además, aporta interoperabilidad entre aplicaciones de software, independientemente de sus propiedades o de las plataformas sobre las que se instale.

2.4 Modelo de Dominio

El Modelo de Dominio o Modelo Conceptual es una representación visual de los conceptos u objetos del mundo real, significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema.

En el Modelo de Dominio no se muestra comportamiento. Las clases conceptuales pueden tener atributos pero no métodos. Para cualquier solución de casos de uso que se haya elegido, los conceptos e ideas propias del dominio del problema son las mismas; un mismo modelo de dominio contempla cualquiera de las soluciones analizadas. El modelo de dominio es global, es decir se realiza para todos los casos de uso y no para uno en particular.

Un modelo de dominio representa la estructura y dinámica de la organización, en el diagrama del dominio que se presenta se muestran los principales conceptos, los tipos más importantes de objetos en el contexto del sistema. Este modelo se representa a través de un diagrama UML.

Algunas veces en dominios de negocios pequeños, no es necesario realizar un modelo de objetos para el dominio, puede ser suficiente con un glosario de términos. Por la relativa simplicidad del entorno donde se desarrolla el plugin, no es necesario profundizar a través de un modelo de negocio, basta con el modelo de dominio para capturar los principales conceptos alrededor del problema que la aplicación resuelve.

2.4.1 Representación del modelo de dominio

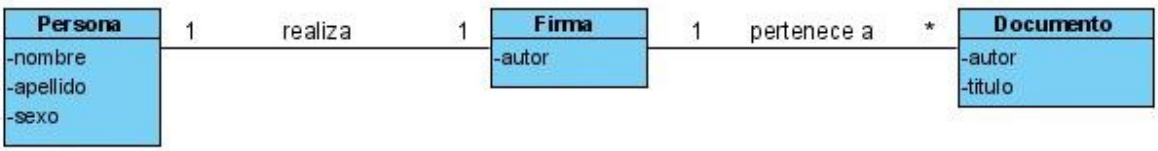


Figura 4: Modelo de dominio

2.4.2 Entidades y conceptos

Para desarrollar un trabajo correcto con el modelo de dominio es necesario hacer una investigación de los principales conceptos utilizados en el entorno de desarrollo. A continuación se presentan todos los conceptos importantes para la realización del plugin.

Persona: individuo que se encarga de realizar la firma a documentos.

Firma: nombre y apellido, o título, que se pone al pie de un escrito, para acreditar que procede de quien lo suscribe, para autorizar lo allí manifestado o para obligarse a lo declarado. (3)

Documento: objeto corporal producto de la actividad humana, que sirve de fuente de conocimiento y que demuestra o prueba algo. (4)

2.5 Modelo del sistema

A continuación se enumeran los requerimientos del plugin. Además, se realiza una representación del caso de uso y su análisis correspondiente.

2.5.1 Especificación de los requerimientos de software

Los requerimientos son una descripción de las necesidades o deseos de un producto. El propósito de esta sección es identificar y documentar lo que en realidad se necesita, en una forma clara y entendible. (5)

2.5.2 Requerimientos Funcionales

Los Requerimientos funcionales describen servicios o funciones que el sistema debe realizar. (6)

RF 1. Iniciar operación de firma digital.

RF 1.1 Registrar datos id (documento) y credenciales de acceso.

RF 1.2 Localizar dirección física de documento.

RF 2. Verificar que usuario tenga certificado.

RF 2.1 Confeccionar firma digital de documento.

RF 3. Registrar usuario que no tenga certificado.

RF 3.1 Verificar datos en Ldap.

RF 3.2 Generar certificado.

RF 3.2.1 Guardar certificado en almacén de claves.

RF 4. Verificar validez de firma.

RF 4.1 Actualizar documento en repositorio (Alfresco).

RF 4.2 Mostrar mensaje de éxito (true).

RF 5. Mostrar mensaje de error en operación de firma (false).

2.5.3 Requerimientos no funcionales

Los requisitos no funcionales (atributos de calidad) aseguran que se disponga de un sistema manejable y gestionable que ofrezca la funcionalidad requerida de manera fiable, ininterrumpida o con el tiempo mínimo de interrupción, incluso ante situaciones inusuales. (7)

Requerimientos de rendimiento

La aplicación debe ser eficiente, rápida y precisa.

Requerimientos de Soporte

El plugin debe ser de fácil instalación, adaptable a numerosas plataformas y de fácil mantenimiento.

Requerimientos de Portabilidad

El plugin debe ser fácil de adaptar a diferentes ambientes sin utilizar otros medios que los previstos.

Requerimientos de Seguridad

El plugin debe tener confidencialidad e integridad ante la información que maneja.

Requerimientos de confiabilidad

El plugin deberá tener un 100% de disponibilidad por lo que podrá ser usado las 24 horas del día.

El tiempo medio de reparación debe ser menor de 1 día.

Todas las salidas del plugin tienen que tener el 100% de veracidad y precisión.

Requerimientos de diseño

1. El lenguaje de programación que se usará es Java.
2. Para el análisis y el diseño del plugin se utilizará la metodología RUP, usando el lenguaje de modelación UML y como herramienta para llevarlo a cabo Visual Paradigm.
3. Para la implementación del plugin se utilizará el IDE NetBeans.

2.5.4 Actores del sistema

El actor del sistema es Aplicación Cliente. A continuación se muestra en la siguiente tabla su justificación.

Actor del sistema	Justificación
Aplicación Cliente	El actor utiliza el servicio web implementado para poder realizar la firma digital a los documentos que gestiona.

Tabla 1: Actor del sistema

2.5.5 Diagrama de casos de uso del sistema

Los casos de uso son fragmentos de funcionalidad del sistema. En ellos se describe la secuencia determinada de eventos que realiza un actor en interacción con la aplicación. Un diagrama de casos de uso es una representación gráfica de los actores y casos de uso del sistema, incluyendo sus interacciones. Todo sistema tiene como mínimo un diagrama, que es una representación gráfica del entorno del sistema (actores) y su funcionalidad principal (casos de uso). Un diagrama de casos de uso muestra, por tanto, los distintos requisitos funcionales que se esperan de una aplicación o sistema y cómo

se relaciona con su entorno (usuarios u otras aplicaciones). En el diagrama de caso de uso del sistema que se presenta el actor Aplicación Cliente se relaciona con el CU Firmar Documento.

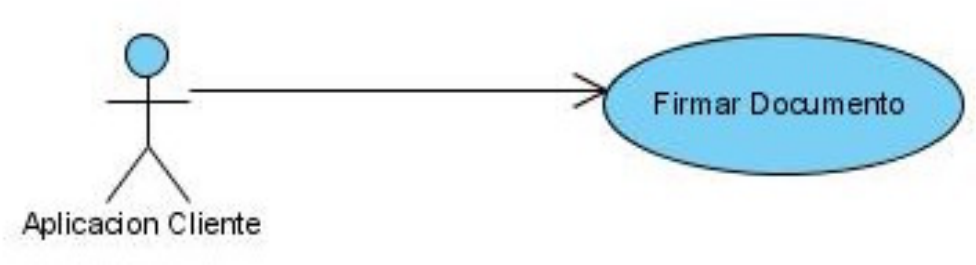


Figura 5: Diagrama de Caso de Uso del Sistema

2.5.6 Caso de uso del sistema (CUS)

CU	**Firmar Documento**	
Propósito	Proporcionar la firma digital de los documentos gestionados por la aplicación cliente.	
Actores	Aplicación Cliente	
Resumen:	El caso de uso se inicia cuando la aplicación cliente requiere del servicio para generar la firma digital a un documento, que está siendo utilizado por un usuario de dicha aplicación cliente. Se localiza el documento en su dirección física, se verifica que el usuario tenga certificado y se realiza la firma y en caso contrario se crea el certificado y de igual forma se realiza la firma, concluyendo el caso de uso.	
Referencias	RF1, RF1.1, RF1.2, RF2, RF2.1, RF3, RF3.1, RF3.2, RF3.2.1, RF4, RF4.1, RF4.2, RF5	
Flujo Normal		
Acción del actor.		**Respuesta del sistema**

1. La aplicación cliente invoca la función de realizar la firma a través del servicio web. 2. Entra los datos (id (documento) y credenciales de acceso).	3. Localiza la dirección física del documento. 4. Verifica si existe certificado usuario. 5. Realiza la firma digital del documento. 6. Actualiza el documento en el repositorio de Alfresco. 7. Finaliza el caso de uso.
Flujo Alterno	
Acciones del actor	**Respuesta del sistema**
Viene del paso 4 del flujo normal	1. El usuario no tiene certificado. 2. Se crea el certificado. 3. Se retorna al paso 5 del flujo normal.

Tabla 2: Descripción del Caso de Uso del Sistema: Firmar Documento

2.6 Conclusiones

En este capítulo se analiza la propuesta de solución basada en el modelo del dominio, se muestra un glosario de términos en el cual se explican aquellas palabras que puedan tener alguna ambigüedad en la comprensión del mismo. Se describe el caso de uso del sistema. Se definen los requisitos funcionales y no funcionales que debe cumplir el plugin, ganando claridad en la concepción del sistema a construir, y sienta las bases para el diseño e implementación del plugin.

CAPÍTULO 3: DISEÑO DEL SISTEMA

3.1 Introducción

En el presente capítulo se realiza una descripción de la organización del trabajo con el código, se expone el diagrama de clases del diseño de manera general, donde se agrupan todas las clases que intervienen en la realización del plugin. Se presenta el diagrama de interacción que muestra el comportamiento entre las clases y se realiza el diagrama de despliegue que describe la distribución física del sistema.

3.2 Estructuración del Modelo del Diseño

En la fase de diseño se modela el plugin de manera que soporte todos los requisitos, creándose una entrada apropiada para las actividades de la fase de implementación.

Lo principal de esta etapa es la elaboración de los diagramas de clases del diseño, donde se muestran las clases participantes. Las clases están incluidas en paquetes de acuerdo a sus funcionalidades. El plugin cuenta con 10 paquetes y 20 clases lo que posibilita una mejor organización y entendimiento del código.

El siguiente diagrama representa el paquete **com.gda.alfresco.manager** que está compuesto por la clase ContentHandler, que es la encargada de la administración del repositorio Alfresco.

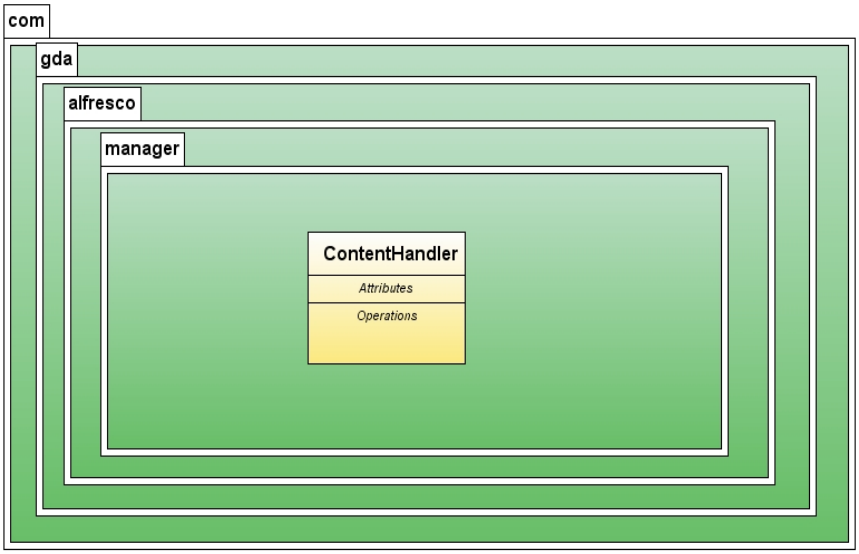


Figura 6: Paquete com.gda.alfresco.manager

El próximo paquete que se representa es **com.gda.alfresco.manager.models** que está formado por las clases Item y User que son las encargadas de establecer los modelos de datos utilizados para la administración.

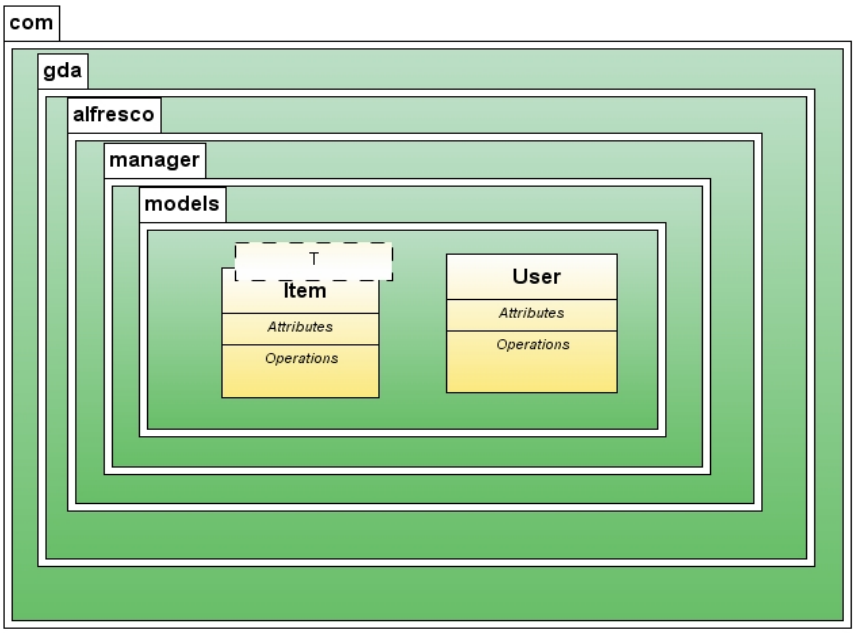


Figura 7: Paquete com.gda.alfresco.manager.models

A continuación se representa el paquete **com.gda.alfresco.manager.utils** que está compuesto por las clases Credentials y Constants, encargadas de crear las credenciales de acceso al repositorio y la definición de las variables constantes utilizadas en la definición de las propiedades de los contenidos en el repositorio.

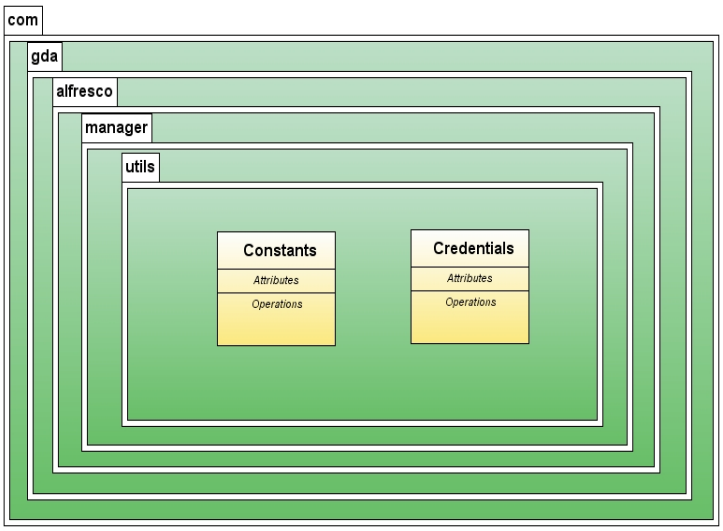


Figura 8: Paquete com.gda.alfresco.manager.utils

El siguiente diagrama representa el paquete **com.gda.alfresco.manager.utils.exceptions** que está formado por las clases InvalidKeyException, QueryFault, NodeRoot y NodeRootMissing, estas dos últimas tienen una relación generalización/especialización. Todas estas clases son empleadas para el manejo de las posibles excepciones generadas por las acciones realizadas sobre los contenidos del repositorio.

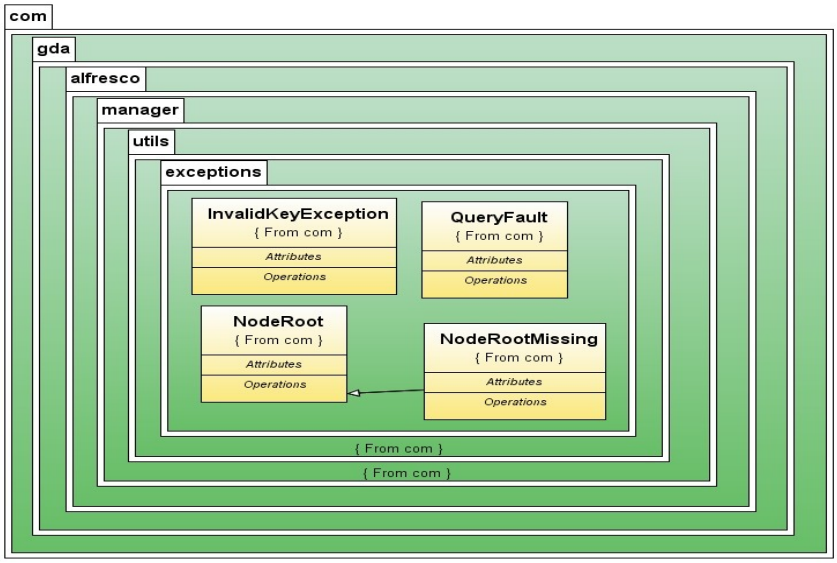


Figura 9: Paquete com.gda.alfresco.manager.utils.exceptions

El paquete **com.gda.conf** está formado por la clase LoadProperties encargada del manejo de las configuraciones del plugin.

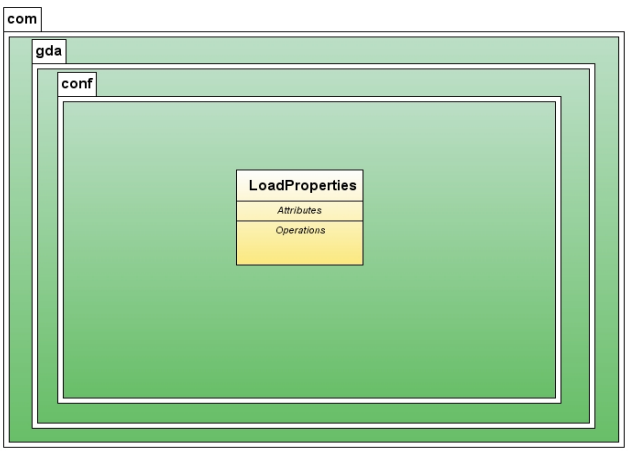


Figura 10: Paquete com.gda.conf

El siguiente paquete **com.gda.external.ldap** contiene las clases AutenticaJNDI y LdapException las cuales son las encargadas de la conexión, trabajo con el servidor Ldap para la generación de los certificados digitales y del trabajo con las posibles excepciones generadas.

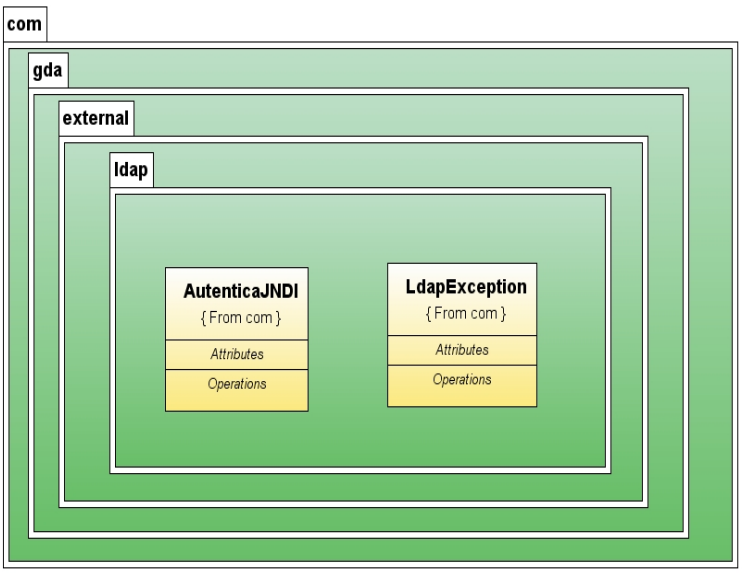


Figura 11: Paquete com.gda.external.ldap

La representación siguiente muestra el paquete **com.gda.external.ldap.wrappers** que contiene las clases LdapUser y LdapWrapperFactory que son encargadas de la generación de los modelos de datos necesarios para el trabajo con el servidor Ldap.

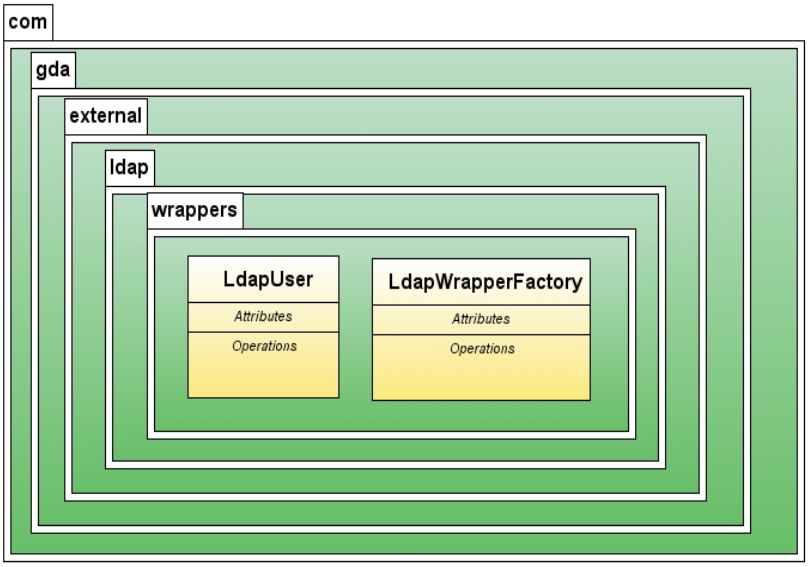


Figura 12: Paquete com.gda.external.ldap.wrappers

El paquete representado a continuación es **com.gda.security.certificate** que está formado por la clase MyCertificate, que se encarga de la generación y validación de los certificados.

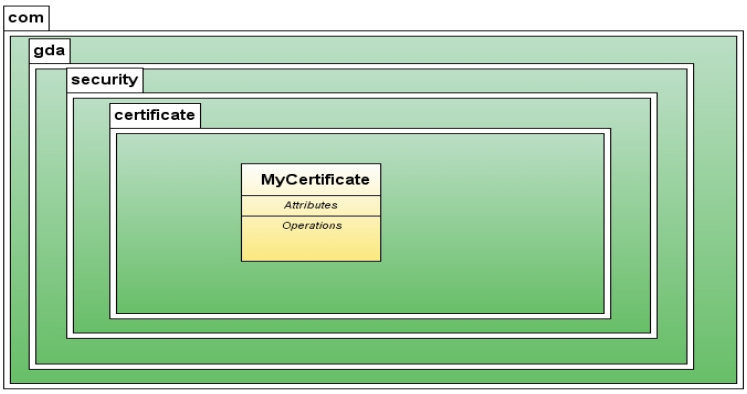


Figura 13: Paquete com.gda.security.certificate

El siguiente paquete **com.gda.security.sign** está compuesto por las clases SignFactory que tiene una relación de composición con la clase Sign que a la vez tiene relación de generalización/especialización con las clases PdfSign y XmlSign, todas tienen la tarea en conjunto de permitir el desarrollo de la firma digital.

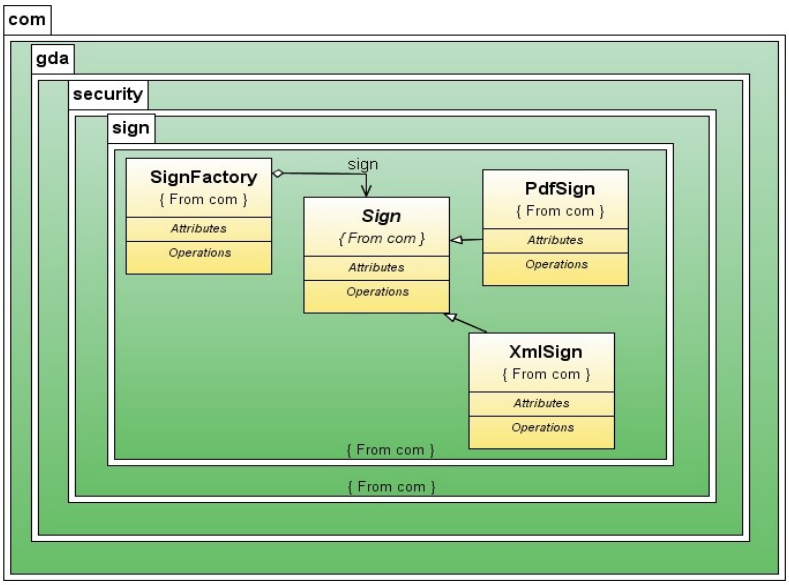


Figura 14: Paquete com.gda.security.sign

El último paquete utilizado es **com.gda.service** que está compuesto por la clase Signer que representa las funcionalidades que brindará el servicio web a implementar.

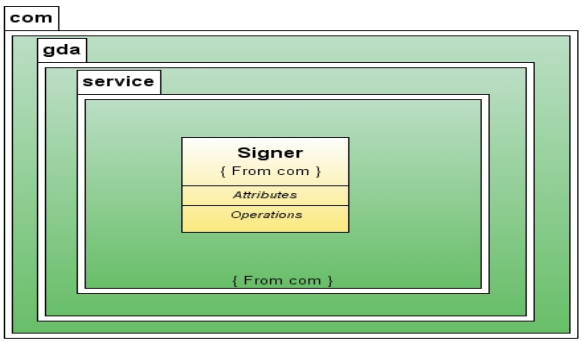


Figura 15: Paquete com.gda.service

3.3 Realización de los casos de uso del diseño

Diagrama de clases del diseño.

El diagrama general de clases del diseño está conformado por 20 clases, entre las cuales existen relaciones de composición, generalización y especialización.

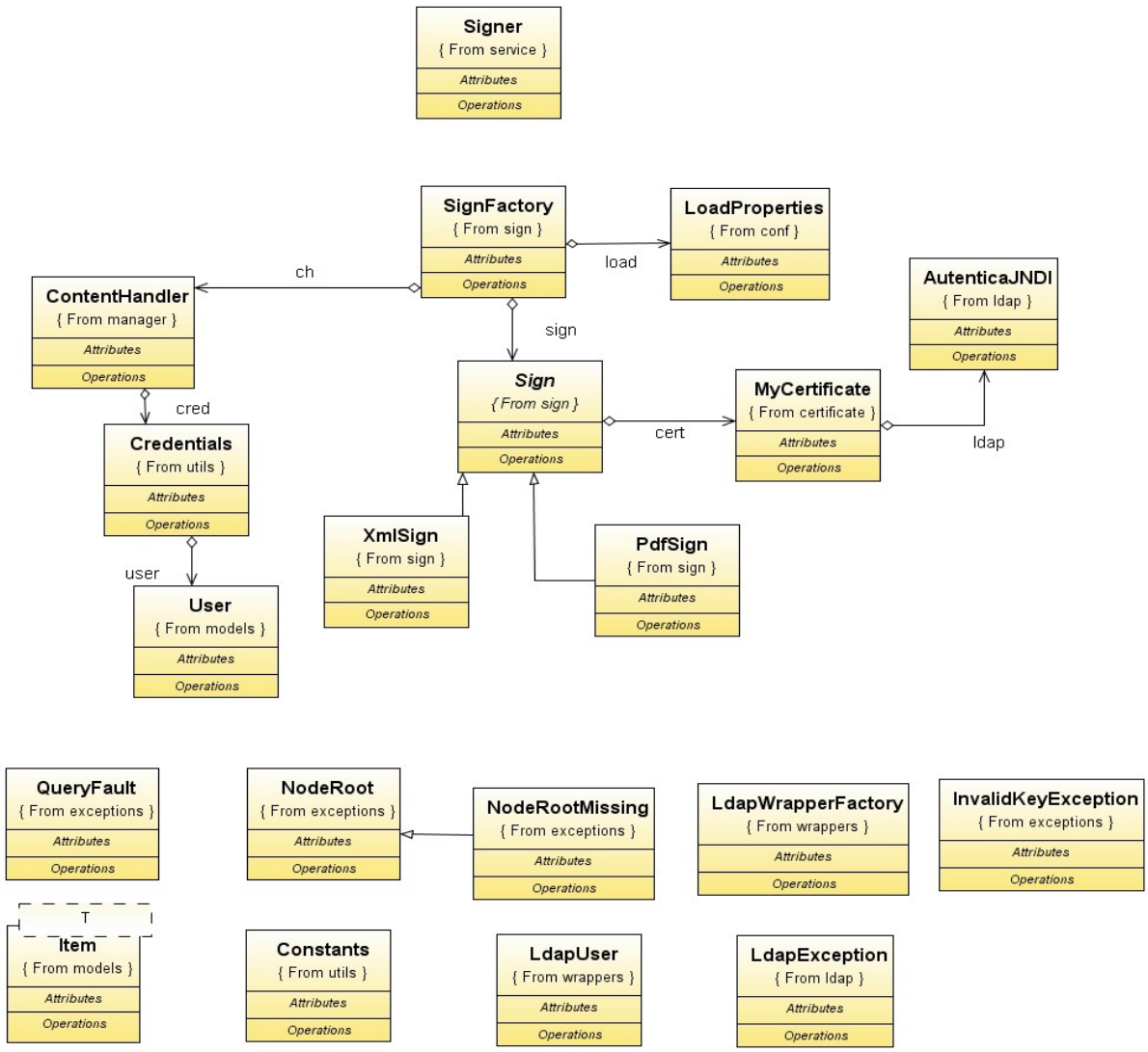


Figura 16: Diagrama general de clases del diseño

Diagramas de interacción

En esta etapa es fundamental la elaboración de los diagramas de interacción que muestran gráficamente cómo los objetos se comunican entre ellos con el objetivo de cumplir los requerimientos.

Esta interacción se puede expresar en diagramas de colaboración y de secuencia. Este último fue el desarrollado, por las características que brinda en la representación de la interacción entre los objetos.

Diagrama de secuencia

El diagrama de secuencia es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada método de la clase. Este contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementarlo y mensajes pasados entre los objetos.

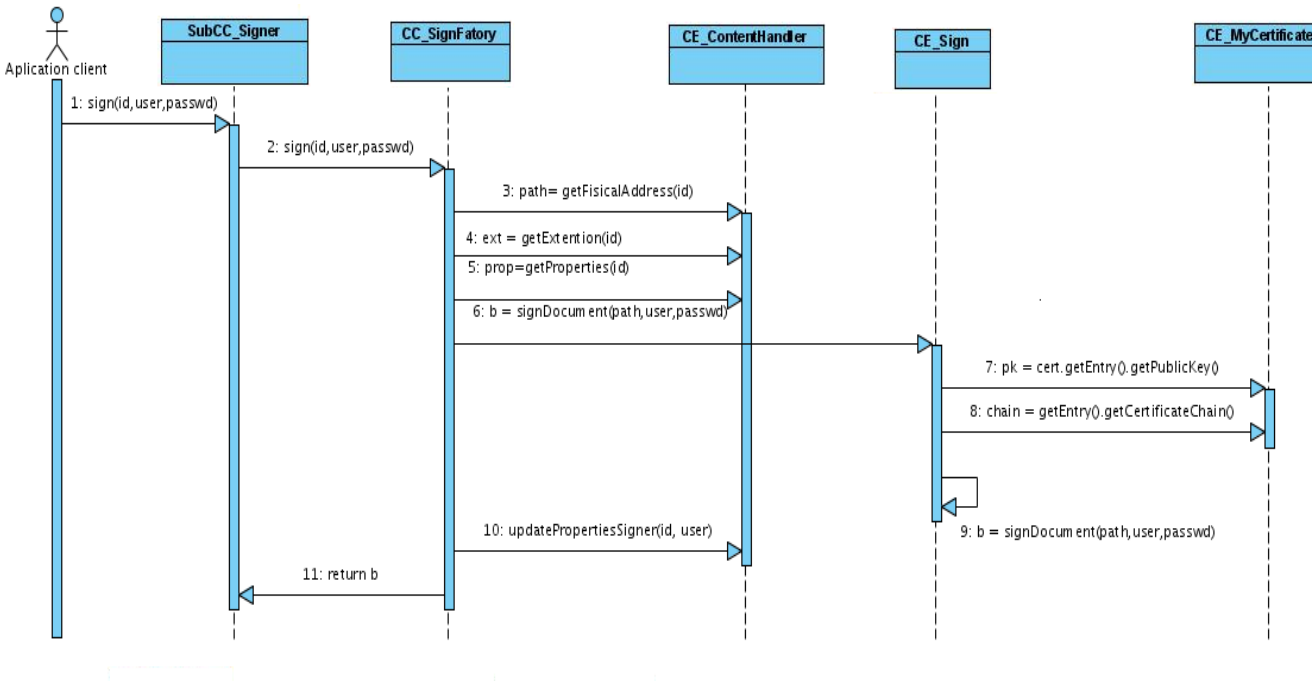


Figura 17: Diagrama de secuencia del Caso de Uso: Firmar Documento

3.4 Descripción de las clases del diseño

A continuación se realiza la descripción de las clases del diseño pertenecientes al desarrollo del plugin.

Nombre: Sign	
Tipo de clase: Entidad	
Atributo	Tipo
pk chain [0….*] user path	PrivateKey Certificate String String
Para cada responsabilidad:	
Nombre:	sign(String tempPath, String finalPath)
Descripción:	Realiza la firma del documento que está en la dirección tempPath y devuelve el documento firmado de la dirección finalPath.
Nombre:	verifySign(String finalPath)
Descripción:	Verifica la firma del documento que está en la dirección finalPath, devuelve true si la firma es válida. False en caso contrario.
Nombre:	getCert( )
Descripción:	Devuelve el certificado.
Nombre:	getPk( )
Descripción:	Devuelve la clave privada.
Nombre:	getChain( )
Descripción:	Devuelve un arreglo de certificados disponibles.
Nombre:	getUser( )
Descripción:	Devuelve el usuario.
Nombre:	getPath( )
Descripción:	Devuelve la dirección física de un documento.

Tabla 3: Descripción clase Sign

Nombre: SignFactory	
Tipo de clase: Controladora	
Atributo	Tipo
id	String
path	String
user	String
passwd	String
ext	String
temppath	String
finalpath	String
properties [0…*]	NamedValue
Para cada responsabilidad:	
Nombre:	getInstance( )
Descripción:	Devuelve una instancia de acuerdo al tipo de firma.
Nombre:	signDocument( )
Descripción:	Firma el documento deseado.
Nombre:	verifierDocumentSingned()
Descripción:	Verifica si la firma del documento es válida.
Nombre:	closeSession( )
Descripción:	Cierra sección.
Nombre:	getPathValues( )
Descripción:	Establece los valores de las direcciones físicas de acuerdo al tipo de firma.

Tabla 4: Descripción clase SignFactory

Nombre: LoadProperties	
Tipo de clase: Entidad	
Atributo	Tipo

properties	Properties
Para cada responsabilidad:	
Nombre:	parseProperties(Properties p, String v)
Descripción:	Devuelve la propiedad convertida a partir de la expresión v.
Nombre:	getValue(String key)
Descripción:	Devuelve el valor de la propiedad cuya clave sea key.

Tabla 5: Descripción clase LoadProperties

Nombre: Item	
Tipo de clase: Entidad	
Atributo	Tipo
label value id serialVersionUID	String T String long
Para cada responsabilidad:	
Nombre:	getId()
Descripción:	Devuelve el identificador (id) del ítem o nodo.
Nombre:	getLabel()
Descripción:	Devuelve el nombre (label) del ítem o nodo.
Nombre:	getValue()
Descripción:	Devuelve las propiedades del ítem o nodo.
Nombre:	setLabel(String label)
Descripción:	Cambia el nombre (label) del ítem o nodo.
Nombre:	toString()
Descripción:	Devuelve el nombre del ítem o nodo.

Tabla 6: Descripción clase Item

Nombre: ContentHandler	
Tipo de clase: Entidad	
Atributo	Tipo
repositoryService	RepositoryServiceSoapBindingStub
Para cada responsabilidad:	
Nombre:	isFolder(Item item)
Descripción:	Devuelve true en el caso que sea un directorio el ítem. False en caso contrario.
Nombre:	haveChails(Item item)
Descripción:	Devuelve true en el caso que el ítem tenga al menos un contenido dentro. False en caso contrario.
Nombre:	haveParent(Item item)
Descripción:	Devuelve true en el caso que el ítem no sea el ítem raíz. False en caso contrario.
Nombre:	haveSign(Item item)
Descripción:	Devuelve true si el contenido se puede firmar. False en caso contrario.
Nombre:	toltem(Node node)
Descripción:	Convierte un nodo del repositorio en un ítem.
Nombre:	getFisicalAdress(String id)
Descripción:	Devuelve la dirección física de un contenido.
Nombre:	getDownloadAdress(String id)
Descripción:	Devuelve la dirección de descarga del contenido.
Nombre:	getMimeType(Item item)
Descripción:	Devuelve el tipo de fichero. “Folder” en caso contrario.
Nombre:	getSize(Item item)
Descripción:	Devuelve el tamaño del fichero. “Folder” en caso contrario.
Nombre:	getContent(Item item)
Descripción:	Devuelve el tipo del fichero. “Folder” en caso contrario.
Nombre:	getEnconding(Item item)

Descripción:	Devuelve la codificación del fichero. “Folder” en caso contrario.
Nombre:	getLocale(Item item)
Descripción:	Devuelve el idioma del fichero. “Folder” en caso contrario.
Nombre:	getChildsItems(String id)
Descripción:	Devuelve una lista de todos los ítem hijos del ítem cuyo id es especificado. Null en caso contrario.
Nombre:	getIdNodeRoot()
Descripción:	Devuelve el id del nodo root o raíz.
Nombre:	getStreamfor(Item item)
Descripción:	Codifica la información del ítem especificado.
Nombre:	getParentsItems(String id)
Descripción:	Devuelve el padre del ítem especificado. Null en caso contrario.
Nombre:	allProperties(String id)
Descripción:	Devuelve un arreglo que contiene todas las propiedades del ítem. Null en caso contrario.
Nombre:	getProperties(String id, String name)
Descripción:	Devuelve el valor de la propiedad cuyo nombre sea name e identificador id. Null en caso contrario.
Nombre:	getProperties(NamedValue properties[0…*], String name)
Descripción:	Devuelve el valor de la propiedad especificado por el name.
Nombre:	updateProperties(String id, String newName)
Descripción:	Actualiza la propiedad nombre (name) del ítem cuyo identificador sea id.
Nombre:	updatePropertiesSigner(String id, String user)
Descripción:	Actualiza las propiedades de la firma para el ítem cuyo identificador sea id.
Nombre:	getItem(String id)
Descripción:	Devuelve el ítem cuyo identificador coincida con id. Null en caso contrario.
Nombre:	deleteItem(String id)
Descripción:	Elimina el ítem cuyo identificador coincida con id. Muestra excepción en caso que no exista dicho ítem.

Nombre:	crearPropertieContent(String fisicalAdress, String mimetipe, String size, String encoding, String locale)
Descripción:	Crea la propiedad content para un ítem.
Nombre:	getExtension(String id)
Descripción:	Devuelve la extensión del fichero cuyo identificador sea id.
Nombre:	getName(String id)
Descripción:	Devuelve el nombre del fichero cuyo identificador sea id.
Nombre:	finalize( )
Descripción:	Cierra la conexión con el servidor.
Nombre:	getCred( )
Descripción:	Devuelve las credenciales de acceso al repositorio.
Nombre:	moveItem(String idNode, String idDestino)
Descripción:	Mueve el ítem cuyo identificador sea idNode para el destino especificado en idDestino.
Nombre:	copyItem(String idNode, String idDestino)
Descripción:	Copia el ítem especificado para el id destino.
Nombre:	updateItemName(String id, NamedValue properties[0…*], String name)
Descripción:	Actualiza el nombre del ítem especificado.

Tabla 7: Descripción clase ContentHandler

Nombre: User	
Tipo de clase: Entidad	
Atributo	Tipo
name passwd	String String
Para cada responsabilidad:	
Nombre:	getName( )
Descripción:	Devuelve el nombre del usuario.

Nombre:	getPasswd( )
Descripción:	Devuelve la contraseña del usuario.

Tabla 8: Descripción clase User

Nombre: Credentials	
Tipo de clase: Entidad	
Atributo	Tipo
providerURL user	String User
Para cada responsabilidad:	
Nombre:	getProviderURL( )
Descripción:	Devuelve la dirección URL del repositorio.
Nombre:	getValidURL( )
Descripción:	Devuelve la dirección URL validada.
Nombre:	openSession ( )
Descripción:	Abre sesión en el repositorio.
Nombre:	closeSeccion( )
Descripción:	Cierra sesión en el repositorio.
Nombre:	getStore(int type)
Descripción:	Devuelve el espacio de trabajo en el repositorio.

Tabla 9: Descripción clase Credentials

Nombre: XmlSign	
Tipo de clase: Entidad	
Atributo	Tipo
prop[0…*]	NamedValue
Para cada responsabilidad:	
Nombre:	verify(String filePath)

Descripción:	Verifica que la firma del fichero ubicado en filePath sea válida. Devuelve false en caso contrario.
Nombre:	outputDocToFile(Document doc, String path)
Descripción:	Escribe un fichero XML listo para firmar en la dirección path.
Nombre:	loadDocumentFromFile(File file)
Descripción:	Carga un fichero XML ubicado en la dirección file.
Nombre:	createNode(Document document, String tag, String value)
Descripción:	Crea el nodo o tag del XML.
Nombre:	createDocument( )
Descripción:	Crea un documento XML en memoria.
Nombre:	documentSigner(String finalPath)
Descripción:	Crea un documento XML en memoria con las propiedades del documento firmado.
Nombre:	changeNameProperties( )
Descripción:	Cambia el nombre de los tag del documento XML.
Nombre:	documentSigned(String filePath)
Descripción:	Devuelve un array con los nombres y apellidos de los usuarios que han firmado el documento.
Nombre:	createTemDocumentSigned(String finalPath)
Descripción:	Crea un archivo XML temporal a partir de un documento firmado.
Nombre:	verifyDocumentSigned(String documentSigned, String tempPath)
Descripción:	Verifica la firma del documento XML temporal firmado.
Nombre:	sign(String tempPath, String finalPath)
Descripción:	Firma un documento XML especificado en la dirección tempPath y devuelve el documento firmado en la dirección finalPath.
Nombre:	verifySign(String finalPath)
Descripción:	Verifica que la firma del documento XML sea válida. Devuelve false en caso contrario.

Tabla 10: Descripción clase XmlSign

Nombre: PdfSign	
Tipo de clase: Entidad	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	updateDocumenSigned(String tempPath, String finalPath)
Descripción:	Actualiza el documento firmado en el repositorio.
Nombre:	excuteSign(String pdfFileName, String finalPath)
Descripción:	Ejecuta la firma del pdf ubicado en la dirección pdfFileName y devuelve el documento firmado en la dirección finalPath.
Nombre:	copyPagePDF(String path, String pathf)
Descripción:	Copia el pdf de la dirección path para la dirección pathf.
Nombre:	createBarCode(String path, String pathf, String sign)
Descripción:	Crea el código de barra correspondiente al valor de la variable sign, en el pdf especificado en la dirección path y lo copia en la dirección pathf.
Nombre:	haveSign(String path)
Descripción:	Devuelve true en el caso que el pdf contenga firma. False en caso contrario.
Nombre:	sign(String tempPath, String3 finalPath)
Descripción:	Firma un documento XML especificado en la dirección tempPath y devuelve el documento firmado en la dirección finalPath.
Nombre:	verifySign(String finalPath)
Descripción:	Verifica que la firma del documento XML sea válida. Devuelve false en caso contrario.

Tabla 11: Descripción clase PdfSign

Nombre: My Certificate

Tipo de clase: Entidad	
Atributo	Tipo
user validity entry	String Int PrivateKeyEntry
Para cada responsabilidad:	
Nombre:	get Ldap ( )
Descripción:	Devuelve un objeto de conexión al Servidor Ldap.
Nombre:	crearIssuerDN(String userIssuerDN, String pasS, String providerUrl Ldap , String version Ldap , String basen Ldap)
Descripción:	Devuelve el emisor del certificado cuando el emisor es un usuario de Ldap.
Nombre:	crearSubjectDN(String user, String pasS, String providerUrl Ldap , String version Ldap , String basen Ldap)
Descripción:	Devuelve los datos del usuario al que se le desea crear el certificado.
Nombre:	createX509V3Certificate(String keyType, int keyBits, int months, String issuerDN, String subjectDN, String domain, String signAlgoritm)
Descripción:	Genera el certificado con los datos especificados.
Nombre:	getEntry()
Descripción:	Devuelve la entidad que contiene todos los datos del certificado.
Nombre:	saveCertificateToKeystore(String user, String dir)
Descripción:	Guarda el certificado del usuario en la dirección especificada. (dir es la dirección del directorio donde se almacenarán los certificados).
Nombre:	getKeystore(String user, String dir)
Descripción:	Devuelve el almacén de claves del usuario especificado en la dirección dir.
Nombre:	expiredCertificate(String user, String dir)
Descripción:	Devuelve true si el certificado ha caducado. False en caso contrario.
Nombre:	set Ldap (AutenticaJNDI Ldap)

Descripción:	Cambia el objeto conexión.
Nombre:	getUser( )
Descripción:	Devuelve el usuario.
Nombre:	getValidity( )
Descripción:	Devuelve la cantidad de meses válidos del certificado.

Tabla 12: Descripción clase My Certificate

Nombre: AutenticaJNDI	
Tipo de clase: Entidad	
Atributo	Tipo
Ldap Seguro USER PASSWOR providerUrl Ldap version Ldap basen Ldap entrada dn	Boolean String String String String String SearchResult String
Para cada responsabilidad:	
Nombre:	conectar Ldap Server( )
Descripción:	Establece la conexión con el servidor Ldap.
Nombre:	buscarPorLogin(String login)
Descripción:	Devuelve un objeto Ldap User con la entrada buscada o Null en caso de que no se encuentre. Ldap Exception en caso de error en la búsqueda.
Nombre:	buscarPorNombreLogin(String nombre)
Descripción:	Devuelve un objeto List que contiene los Ldap User que coinciden con la búsqueda (o una lista vacía). Ldap Exception en caso de error en la búsqueda.

Nombre:	buscar(String login)
Descripción:	Devuelve un objeto SearchResult con la entrada buscada o Null en caso de que no se encuentre. Ldap Exception en caso de error en la búsqueda.
Nombre:	login(String login, String clave)
Descripción:	Devuelve un boolean indicando si se autenticó al usuario o no. Ldap Exception cuando se produce un fallo en la autenticación deberá buscar al usuario, y una vez se tenga el Ldap User, llamar a la función public boolean login(Ldap User u, String clave).
Nombre:	getEntrada Ldap ( )
Descripción:	Devuelve el objeto SearchResult correspondiente a la entrada en el Ldap. Deberá buscar al usuario, y una vez se tenga el Ldap User, llamar a la función public boolean login (Ldap User u, String clave).
Nombre:	login(Ldap User usuario, String clave)
Descripción:	Devuelve un boolean indicando si se autenticó el usuario o no. Ldap Exception cuando se produce un fallo en la autenticación.

Tabla 13: Descripción clase AutenticaJNDI

Nombre: Constants	
Tipo de clase: Entidad	
Atributo	Tipo
NAMESPACE_SYSTEM_MODEL	String
NAMESPACE_CONTENT_MODEL	String
NODE_UUID	String
NODE_DB_ID	String
NODE_STORE_NAME	String
NODE_CREATOR	String
NODE_NAME	String
NODE_CONTENT	String
NODE_MODIFIED	String

NODE_CREATED	String
NODE_STORE_PROTOCOL	String
NODE_MODIFIER	String
NODE_PATH	String
ASSOC_CONTAINS	String
TYPE_FOLDER	String
TYPE_CONTENT	String
NODE_TITLE	String
NODE_ASSOCIATION_TYPE	String
NODE_ASSOCIATION_NAME	String
ROOT_ADRESS	String
QUERY_LANG_LUCENE	String
Para cada responsabilidad:	
Nombre:	createQNameString(String namespace, String name)
Descripción:	Devuelve el resultado de concatenar dos cadenas.
Nombre:	reemplazarPath(String path)
Descripción:	Devuelve el resultado de reemplazar la dirección existente por path.

Tabla 14: Descripción clase Constants

Nombre: Ldap WrapperFactory	
Tipo de clase: Entidad	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	getWrapper(SearchResult sr)
Descripción:	Dependiendo del tipo de resultado, el wrapper creará el objeto adecuado. La superclase siempre será Ldap User. Se puede comprobar el tipo de la instancia con 'instanceof'. Devuelve un usuario de Ldap o Ldap Exception

	cuando se produce un error.

Tabla 15: Descripción clase Ldap WrapperFactory

Nombre: Ldap User	
Tipo de clase: Entidad	
Atributo	Tipo
UC3MRELACION sr	Properties SearchResult
Para cada responsabilidad:	
Nombre:	get(String attr)
Descripción:	Este método extrae el valor de la propiedad que se le pasa como parámetro. Se puede llamar directamente, pero está pensado para que los demás métodos lo llamen a él, pues cada uno tiene la correspondencia del nombre del campo al valor que queremos: esto es, para obtener el nombre y los apellidos, la propiedad en Ldap es 'cn'. Para facilitar el trabajo, se proporciona el método getNombreApellidos().
Nombre:	getSearchResult( )
Descripción:	Este método devuelve el objeto SearchResult que se está envolviendo para permitir hacer operaciones adicionales que no están recogidas en el resto del método, devuelve el resultado de la búsqueda.
Nombre:	getAll(String attr)
Descripción:	Este método extrae los valores de la propiedad que se le pasa como parámetro. Devuelve los valores de la propiedad o Null.
Nombre:	getRelacionesComoTexto( )
Descripción:	Devuelve un array con las descripciones de todos los tipos que tiene.
Nombre:	getRelaciones( )
Descripción:	Si se quieren las descripciones de los tipos, se debe usar el método getRelacionesComoTexto(). Este método devuelve un array con estos

	valores.
Nombre:	getLogin( )
Descripción:	Devuelve el login del usuario o el nombre de usuario como tal.
Nombre:	getNombreApellidos( )
Descripción:	Devuelve el nombre y apellidos del usuario.
Nombre:	getOrganizationUnit( )
Descripción:	Devuelve la unidad organizacional a que pertenece dicho usuario.
Nombre:	getOrganization( )
Descripción:	Devuelve la organización a que pertenece dicho usuario.
Nombre:	getApellidosDespuesNombre( )
Descripción:	Devuelve los apellidos y después el nombre del usuario.
Nombre:	getEmail( )
Descripción:	Devuelve el email del usuario.
Nombre:	getNombre Ldap ( )
Descripción:	Método para obtener el nombre completo del usuario en Ldap. Vale para, junto con la base de búsqueda, formar la cadena completa de la entrada en Ldap.
Nombre:	getDistinguiShedName( )
Descripción:	Devuelve los datos del usuario concatenados y separados por coma.
Nombre:	equals(Object obj)
Descripción:	Compara si dos objetos son iguales. Devuelve true si son iguales, false en caso contrario.
Nombre:	hashCode( )
Descripción:	Método sobrescrito para la identidad de las instancias.
Nombre:	compareTo(Object o)
Descripción:	Método para ordenar con Collections.sort (...) listas de Ldap Users.

Tabla 16: Descripción clase Ldap User

Nombre: Ldap Exception	
Tipo de clase: Entidad	
Atributo	Tipo
serialVersionUID	long
Para cada responsabilidad:	
Nombre:	getCausa( )
Descripción:	Devuelve la causa del error.

Tabla 17: Descripción clase Ldap Exception

Nombre: NodeRoot	
Tipo de clase: Entidad	
Atributo	Tipo
Descripción:	Devuelve las excepciones de tipo Exception de NodeRoot.

Tabla 18: Descripción clase NodeRoot

Nombre: NodeRootMissing	
Tipo de clase: Entidad	
Atributo	Tipo
Descripción:	Recoge las excepciones de tipo NodeRootMissing.

Tabla 19: Descripción clase NodeRootMissing

Nombre: QueryFault	
Tipo de clase: Entidad	
Atributo	Tipo
serialVersionUID	long

Descripción:	Recoge las excepciones de tipo QueryFault.

Tabla 20: Descripción clase QueryFault

Nombre: Signer	
Tipo de clase: Controladora	
Atributo	Tipo
Para cada responsabilidad:	
Nombre:	sign(String id, String user, String password)
Descripción:	Devuelve true en caso que la firma sea exitosa. False en caso contrario.
Nombre:	verifier(String id, String user, String password)
Descripción:	Devuelve true en caso que la firma sea válida. False en caso contrario.

Tabla 21: Descripción clase Signer

Nombre: InvalidKeyException	
Tipo de clase: Entidad	
Atributo	Tipo
serialVersionUID	long
Descripción:	Devuelve las excepciones de tipo InvalidKeyException.

Tabla 22: Descripción clase InvalidKeyException

3.5 Patrones de asignación de responsabilidades utilizados en el diseño

Para la realización del diseño de la aplicación informática a implementar, se utilizaron los patrones GRASP, estos son Patrones Generales de Software para Asignación de Responsabilidades, es el acrónimo de General Responsability Assignment Software Patterns. Se considera que más que patrones son una serie de "Buenas Prácticas" de aplicación recomendable en el diseño de software. Entre los patrones GRASP se encuentran los siguientes:

Experto: el GRASP de Experto en información es el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo, lo cual permite que se conserve el encapsulamiento, soportando un bajo acoplamiento y una alta cohesión.

Creador: ayuda a identificar quién debe ser el responsable de la creación de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que: tiene la información necesaria para realizar la creación del objeto, usa directamente las instancias creadas del objeto, o almacena o maneja varias instancias de la clase. Este patrón brinda soporte de bajo acoplamiento, lo cual supone menos dependencias entre clases y posibilidades.

El patrón Experto y Creador se ponen de manifiesto en la clase SignFactory porque es la encargada de la creación de las instancias de las diferentes clases utilizadas en la realización de la firma.

Alta cohesión y bajo acoplamiento: se pueden separar, aunque están íntimamente ligados, de hecho si se aumenta mucho la cohesión del sistema software, se tiene un alto acoplamiento entre las clases, y por el contrario si reduce mucho el acoplamiento, se verá mermada la cohesión.

Alta cohesión: este patrón propone asignar la responsabilidad de manera que la complejidad se mantenga dentro de límites manejables asumiendo solamente las responsabilidades que deben manejar, evadiendo un trabajo excesivo. Su utilización mejora la claridad y facilidad con que se entiende el diseño, simplifica el mantenimiento y las mejoras de funcionalidad, generan bajo acoplamiento, soporta mayor capacidad de reutilización.

Este patrón se pone de manifiesto en la mayoría de las clases porque cada una es capaz de realizar sus responsabilidades sin la utilización de las demás.

Bajo acoplamiento: es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

Durante todo el trabajo con las clases del diseño se manifestó este patrón, pues solo existe relación entre siete clases lo que significa que existe poca dependencia y cada clase realiza sus funciones sin necesitar de otra.

Controlador: este patrón funciona como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos y la que los envía a las distintas clases según el método llamado.

Este patrón se pone de manifiesto cuando la clase Signer recibe los datos enviados desde una aplicación cliente y los envía a la clase SignFactory la cual es la encargada del manejo de la información.

3.7 Tratamiento de errores

En el diseño se tuvo en cuenta el tratamiento de errores a través de mensajes de fácil comprensión y lo más descriptivos posible, manteniendo al usuario informando de posibles riesgos o errores cometidos, mostrando además mensajes condicionales.

El tratamiento de errores se aprecia en la clase SignFactory que es la encargada de controlar todas las gestiones para realizar la firma. En el constructor se tratan los posibles errores a la hora de inicializar los valores de las diferentes variables e instancias, asegurando así, que cada una tome el valor que le corresponde. En caso contrario se captura el error ocurrido.

3.8 Seguridad

Para la seguridad del sistema el empleo de la autenticación es importante. Para ello se utilizan niveles de acceso para los diferentes usuarios del sistema, cada uno tendrá acceso a firmar los documentos que le pertenecen.

La clase encargada de la seguridad es Credentials, se apoya en la clase User para crear la instancia del usuario que establecerá la conexión con el repositorio.

3.9 Diagrama de despliegue

El modelo de despliegue describe la distribución física del sistema, muestra como están distribuidos los componentes de software entre los distintos nodos de cómputo. Permite comprender la correspondencia entre la arquitectura software y la arquitectura hardware. Las estaciones de trabajo, dispositivos y procesadores son reflejados como nodos.

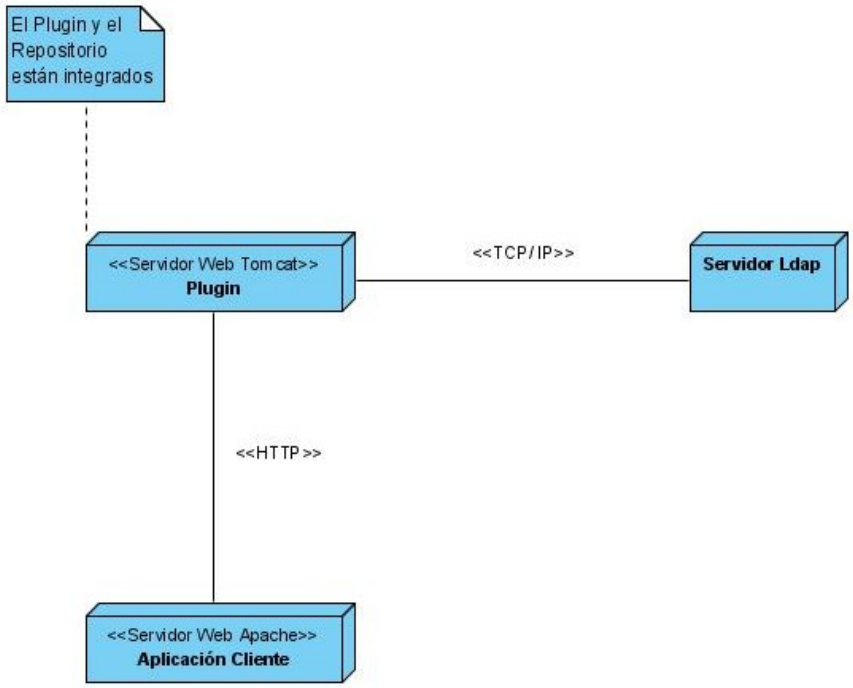


Figura 18: Diagrama de despliegue

En el diagrama de despliegue se representan varios nodos que constituyen los ordenadores presentes en el entorno de trabajo, el Plugin y el Repositorio están ubicados en el mismo ordenador pues este garantiza características de seguridad imprescindibles a la información que maneja el repositorio. La Aplicación

Cliente se comunica mediante el protocolo de comunicación HTTP con el Plugin y este se conecta mediante el protocolo de comunicación TCP/IP con el Servidor Ldap para trabajar con los datos de los usuarios en la generación y validación de los certificados digitales.

3.10 Conclusiones

En este capítulo se muestra el diagrama de clases del diseño y el diagrama de secuencia. Se describen las clases utilizadas y los patrones de diseño utilizados, por último el diagrama de despliegue que muestra como están distribuidos los dispositivos de software entre los diferentes nodos.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

4.1 Introducción

En este capítulo se expone todo lo relacionado con los flujos de trabajo Implementación y Prueba, cerrando con su realización el desarrollo del plugin, aquí se presenta el diagrama de componentes de forma general y además se realizan pruebas de caja blanca para asegurar la calidad del producto.

4.2 Estructuración del Modelo de Implementación

El diagrama de componentes define cómo las clases, artefactos y otros elementos de bajo nivel, se unen para formar componentes de alto nivel y las conexiones entre ellos. Los componentes son artefactos de software compilados que trabajan acoplados para brindar el comportamiento requerido dentro de las restricciones definidas en el proceso de captura de requisitos.

El diagrama de componentes se realiza de manera general, además, está compuesto por dos paquetes: lib y extentions, el primero está formado por todas las librerías que utiliza la clase ContentHandler y el segundo contiene los componentes donde están implementadas las clases manejadoras de los posibles tipos de excepciones en el trabajo con el repositorio Alfresco.

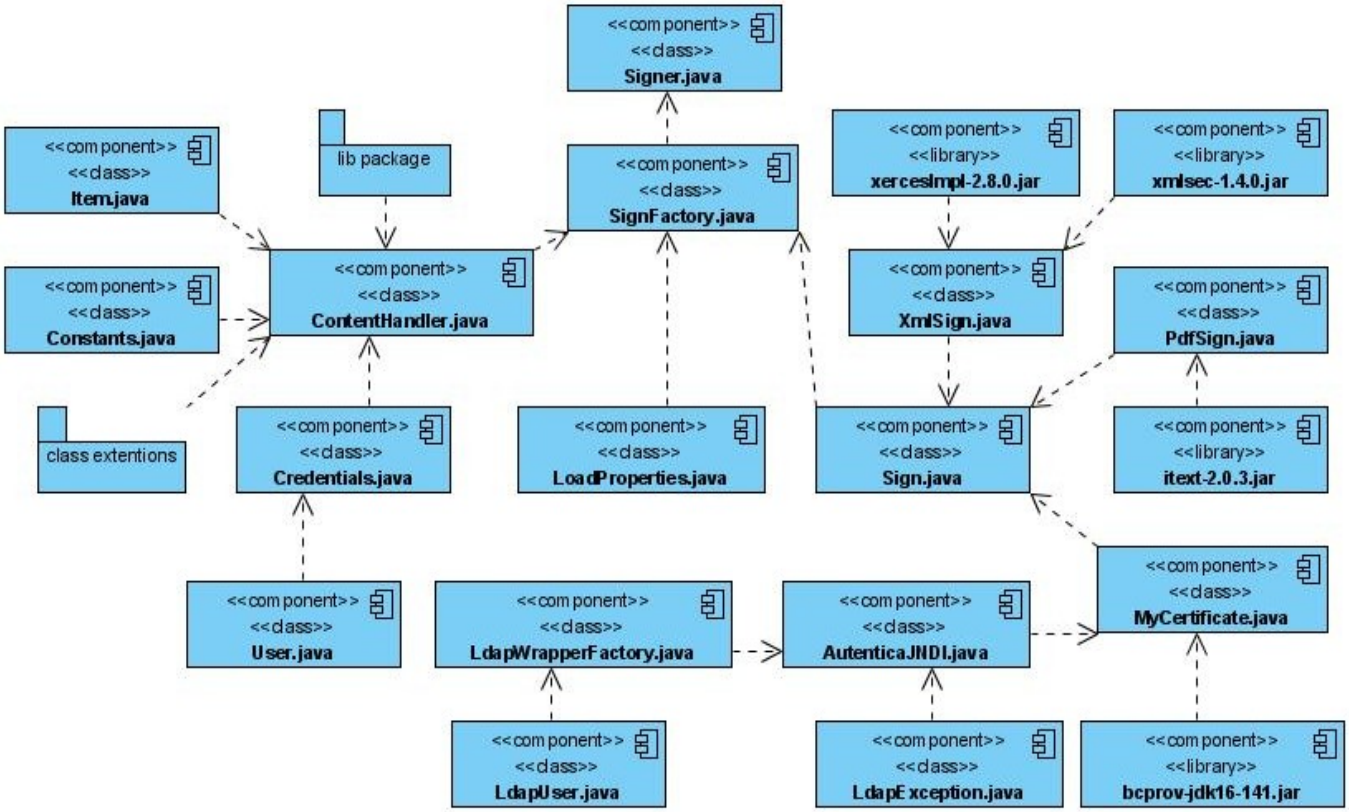


Figura 19: Diagrama de componentes

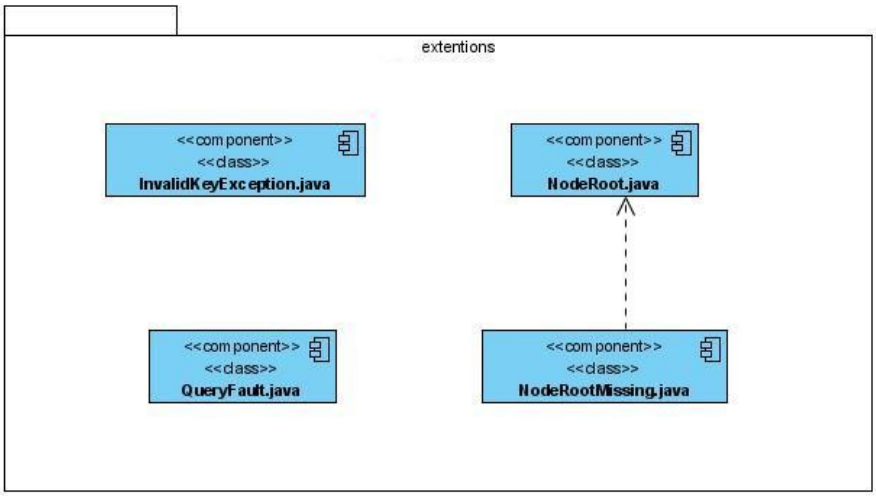


Figura 20: Representación del Paquete extentions

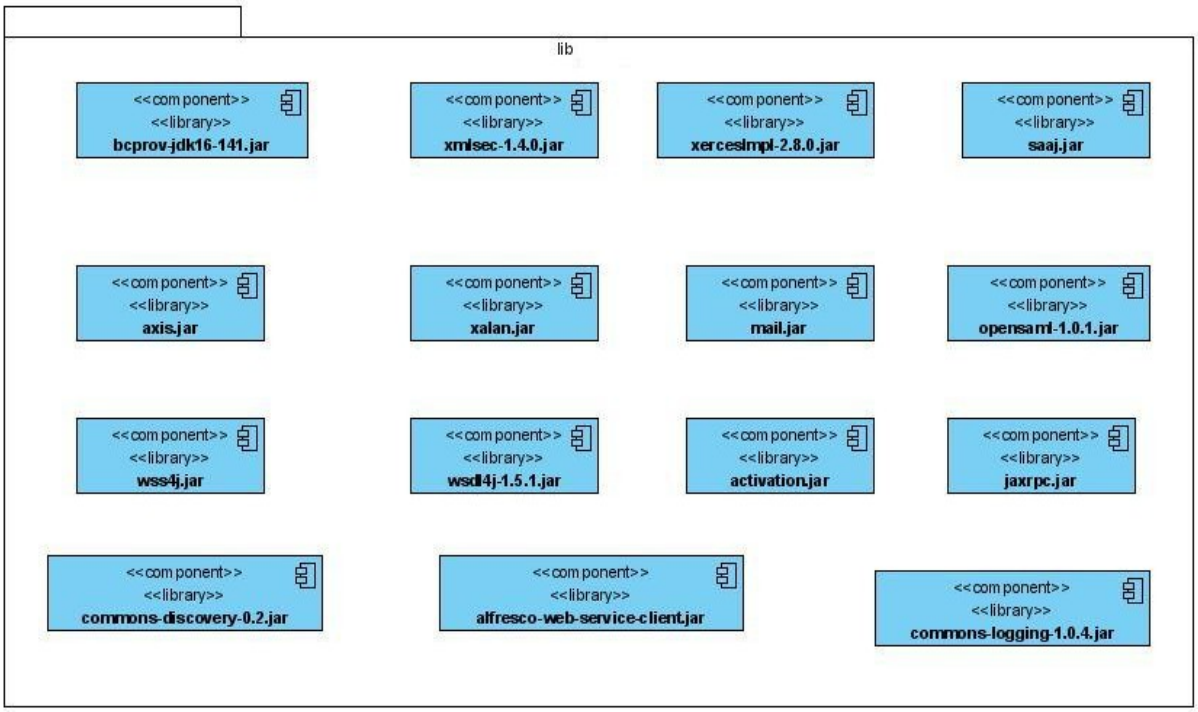


Figura 21: Representación del Paquete lib

4.3 Pruebas de Caja Blanca

Las pruebas de caja blanca están dirigidas a las funciones internas del software. Entre las técnicas usadas se encuentran; la cobertura de caminos (pruebas que hagan que se recorran todos los posibles caminos de ejecución), pruebas sobre las expresiones lógico-aritméticas, pruebas de camino de datos (definición-uso de variables) y la comprobación de los bucles (se verifican los bucles para 0,1 y n iteraciones, y luego para las iteraciones máximas, máximas menos uno y más uno).

4.3.1 Objetivos

El objetivo de realizar este tipo de prueba al plugin es que se garantice que se ejerciten por lo menos una vez todos los caminos independientes de cada método, todos los bucles en sus límites operacionales así como las estructuras internas de datos para asegurar su validez.

4.3.2 Alcance

El proceso de pruebas de caja blanca se va a concentrar principalmente en validar que cada método funcione apropiadamente.

4.3.3 Descripción

La prueba de Caja Blanca es considerada como uno de los tipos de pruebas más importantes que se le aplican a los programas informáticos, logrando como resultado la disminución de los errores existentes en los sistemas y por ende una mayor calidad y confiabilidad.

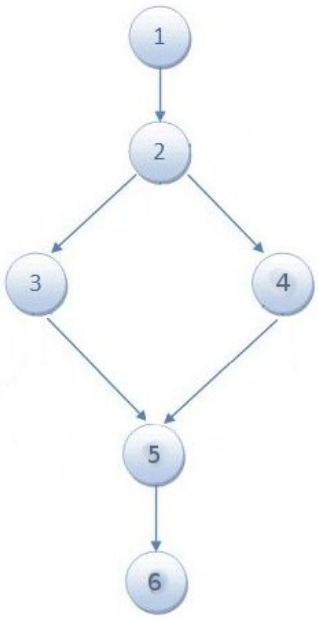
4.3.4 Métrica de la complejidad ciclomática

Se realizó el cálculo de la complejidad ciclomática a todos los bloques de código dentro del plugin, lo cual ayudó a reflejar una medida de la complejidad del código escrito, teniendo en cuenta el número de destinos posibles. Además permitió conocer el esfuerzo a realizar en cada una de las pruebas, el cual es exactamente el valor de la complejidad ciclomática en cada elemento. A partir de esta medida, se diseñaron pruebas que forzaron el recorrido de estos caminos, lo cual garantiza que se ejecute al menos una vez cada sentencia del programa y que cada condición se ejecute en sus variantes verdaderas y falsas.

4.3.4.1 Ejemplo 1: Método que realiza la firma a un documento

```
public boolean sign(String tempPath, String finalPath) {
 boolean resul = false;
 try {
 if (!new File(finalPath + new File(path).getName() + ".xml").isFile()) {
 outputDocToFile(createDocument(), finalPath + new File(path).getName() + ".xml");
 tempPath = finalPath + new File(path).getName() + ".xml";
 resul = true;
 } else {
 outputDocToFile(documentSigner(finalPath + new File(path).getName() + ".xml"), tempPath);
 new File(finalPath + new File(path).getName() + ".xml").delete();
 new File(tempPath).renameTo(new File(finalPath + new File(path).getName() + ".xml"));
 tempPath = finalPath + new File(path).getName() + ".xml";
 resul = true;
 }
 Document doc = (Document) loadDocumentFromFile(new File(tempPath));
 Constants.setSignatureSpecNSprefix("");
 File signatureFile = new File(tempPath);
 String baseURI = signatureFile.getAbsolutePath();
 XMLSignature xmlSignature =
 new XMLSignature(doc, baseURI, XMLSignature.ALGO_ID_SIGNATURE_RSA_SHA1);
 doc.getDocumentElement().appendChild(xmlSignature.getElement());
 Transforms transforms = new Transforms(doc);
 transforms.addTransform(Transforms.TRANSFORM_ENVELOPED_SIGNATURE);
 xmlSignature.addDocument("", transforms, Constants.ALGO_ID_DIGEST_SHA1);
 X509Certificate certs = (X509Certificate) this.cert.getEntry().getCertificate();
 xmlSignature.addKeyInfo(certs);
 xmlSignature.addKeyInfo(certs.getPublicKey());
 xmlSignature.sign(this.pk);
```

```
outputDocToFile(doc, signatureFile.getAbsolutePath());
} catch (NodeRootMissing ex) {
return false;
} catch (Exception ex) {
return false;
}
return resul;
}
```



Complejidad ciclomática *V(G)*

V(G) = N. de aristas - N. de nodos + 2

V(G) = 6 - 6 + 2

V(G) = 2

Caminos independientes:

1-2-3-5-6

1-2-4-5-6

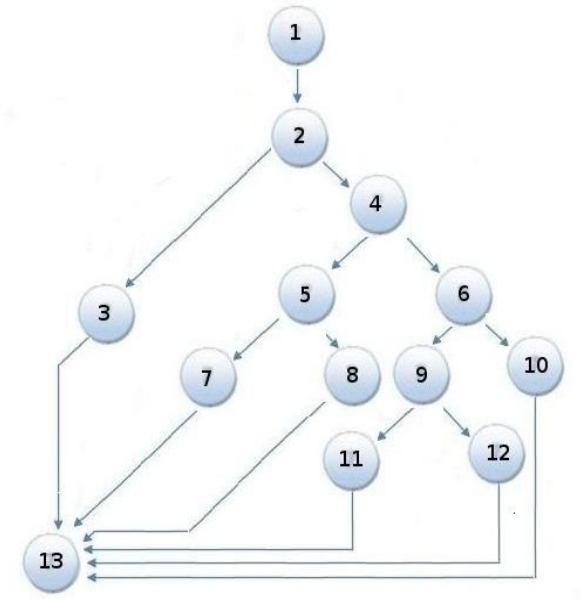
Figura 22: Grafo de flujo para el método que realiza la firma a un documento

4.3.4.2 Ejemplo 2: Método que verifica la firma de un documento

```
private boolean verify(String filePath) {
 boolean res = false;
 try {
```

```
System.out.println(filePath);
String signatureFileName = filePath;
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
dbf.setNamespaceAware(true);
dbf.setAttribute("http://xml.org/sax/features/namespaces", true);
File f = new File(signatureFileName);
DocumentBuilder db = dbf.newDocumentBuilder();
Document doc = db.parse(new java.io.FileInputStream(f));
Element sigElement = (Element)
doc.getElementsByTagName("Signature").item(doc.getElementsByTagName("Signature").getLength() - 1);
XMLSignature signature = new XMLSignature(sigElement, f.getAbsolutePath());
KeyInfo keyInfo = signature.getKeyInfo();
if (keyInfo != null) {
X509Certificate certs = keyInfo.getX509Certificate();
if (certs != null) {
if (signature.checkSignatureValue(certs)) {
res = true;
} else {
res = false;
}
} else {
PublicKey pkey = keyInfo.getPublicKey();
if (pkey != null) {
if (signature.checkSignatureValue(pkey)) {
 res = true;
} else {
 System.out.println("Inválido según la clave pública");
}
} else {
res = false;
```

```
}
}
} else {
res = false;
}
} catch (RemoteException ex) {
return false;
} catch (Exception ex) {
return false;
}
return res;
}
```



Complejidad ciclomática *V(G)*

V(G) = N. de aristas - N. de nodos + 2

V(G) = 18 – 14 + 2

V(G) = 6

Caminos independientes:

1-2-3-13

1-2-4-5-7-13

1-2-4-5-8-13

1-2-4-6-9-11-13

1-2-4-6-9-12-13

1-2-4-6-10-13

Figura 23: Grafo de flujo para el método que verifica la firma de un documento

4.4 Casos de pruebas por método analizado

Los casos de prueba son un conjunto de condiciones o variables bajo las cuales se determina si el requisito de una aplicación es parcial o completamente satisfactorio.

Ejemplo 1

public boolean sign(String tempPath, String finalPath)	Precondi-ciones	Casos	Camino Indepen-diente	Entrada	Resultados esperados	Resultados obtenidos
	Creden-ciales de acceso e id del docu-mento a firmar son válidos.	Que el docu-mento no tenga firma.	1-2-3-5-6	**finalPath**=”/opt/ Alfresco/alf_data/.sign /xml/dffea50d-5ab5-4e6f-950c-ba107 8e74833.bin.xml”. **tempPath**=”/opt/ Alfresco/alf_data/.sign /xml/tem.xml”	true	true
	Creden-ciales de acceso e id del docu-mento a firmar son válidos.	Que el docu-mento tenga firma.	1-2-4-5-6	**finalPath**=” /opt/Alfresco/alf_data/. sign/xml/dffea50d-5ab5-4e6f-950c-ba1078e74833.bin. xml” **tempPath**=”/opt/ Alfresco/alf_data/.sign /xml/tem.xml”	true	true

Tabla 23: Casos de Prueba para el método que realiza la firma a un documento

Ejemplo 2

private boolean verify(String filePath)	Precondi-ciones	Casos	Camino indepen-diente	Entrada	Resultados esperados	Resultados obtenidos
	Credenciales de acceso e id del documento son válidos.	Que el documento no tenga firma.	1-2-3-13	**filePath**= ”/opt/ Alfresco/alf_data/.sign/xml/dffea50d-5ab5-4e6f-950c-ba1078e74833.bin.xml”	false	false
	Credenciales de acceso e id del documento para verificar la firma son válidos.	Que el documento tenga firma y se verifique la firma con el certificado y ésta sea válida.	1-2-4-5-7-13	**filePath**=”/opt/ Alfresco/alf_data/.sign/xml/dffea50d-5ab5-4e6f-950c-ba1078e74833.bin.xml”	true	true
	Credenciales de acceso e id del documento a firmar son válidos.	Que el documento tenga firma y se verifique la firma con el certificado y ésta no sea válida.	1-2-4-5-8-13	**filePath**=”/opt/ Alfresco/alf_data/.sign/xml/dffea50d-5ab5-4e6f-950c-ba1078e74833.bin.xml”	false	false

	Credenciales de acceso e id del documento a firmar son válidos.	Que el documento tenga firma y se verifique la firma con la clave pública y ésta sea válida.	1-2-4-6-9-11-13	**filePath**=”/opt/Alfresco/alf_data/.sign/xml/dffea50d-5ab5-4e6f-950c-ba1078e74833.bin.xml”	true	true
	Credenciales de acceso e id del documento a firmar son válidos.	Que el documento tenga firma y se verifique la firma con la clave pública y ésta no sea válida.	1-2-4-6-9-12-13	**filePath**=”/opt/Alfresco/alf_data/.sign/xml/dffea50d-5ab5-4e6f-950c-ba1078e74833.bin.xml”	false	false
	Credenciales de acceso e id del documento a firmar son válidos.	Que el documento tenga firma y se verifique la firma con la clave pública y el certificado, pero ambos	1-2-4-6-10-13	**filePath**=”/opt/Alfresco/alf_data/.sign/xml/dffea50d-5ab5-4e6f-950c-ba1078e74833.bin.xml”	false	false

		sean no válidos.				

Tabla 24: Casos de Prueba para el método que verifica la firma de un documento

En los casos de pruebas realizados los resultados fueron satisfactorios, pues los resultados obtenidos fueron iguales a los resultados esperados.

4.5 Conclusiones

En este capítulo se comenzó mostrando el diagrama de componentes. Se realizaron dos casos de prueba de caja blanca analizando los métodos más importantes en la implementación del plugin, estas pruebas guían la calidad del sistema, además, son las que determinan si están creadas las condiciones para continuar avanzando en el seguimiento del producto.

CONCLUSIONES

Con la realización de este trabajo se logró la implementación del plugin que integrado a Alfresco brinda seguridad, autenticidad y confidencialidad a la información gestionada por este gestor de contenido empresarial utilizado en gran parte del mundo para la gestión documental. Para lograr el resultado final se cumplieron los objetivos específicos trazados:

- Se realizó un estudio detallado sobre la firma digital de documentos adquiriendo vastos conocimientos para emprender la investigación.
- Se seleccionaron las tecnologías más adecuadas para la implementación del plugin, atendiendo a sus características.
- Se desarrolló el proceso de ingeniería de software, realizándose de manera correcta todos los flujos de trabajo desde Modelación del Negocio hasta Prueba.

RECOMENDACIONES

- Implementar nuevas funcionalidades para mejorar la calidad y robustez del producto.
- Validar el funcionamiento de la herramienta a gran escala.
- Presentar este trabajo en futuros eventos científicos.
- Continuar su perfeccionamiento para futuros trabajos de postgrado.

REFERENCIAS BIBLIOGRÁFICAS

1. Firma Digital. [En línea] [Citado el: 16 de enero de 2009.] http://www.corrientes.gov.ar/firmadigital/?page id =32.

2. **Fidanza, Jorge Alberto López.** *Firma Digital.* 2004.

3. Revista de Derecho Informático. [En línea] agosto de 2000. [Citado el: 2 de marzo de 2009.] http://www.alfa-redi.org/rdi-articulo.shtml?x=533.

4. ACCESS my LIBRARY. [En línea] 5 de enero de 2005. [Citado el: 2 de marzo de 2009.] http://www.accessmylibrary.com/coms2/summary_0286-32118092_ITM.

5. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos .* México : s.n., 1999.

6. *Ingeniería de Requerimientos.* 1995.

7. IT-Checklists.com. [En línea] [Citado el: 5 de marzo de 2009.] http://www.it-checklists.com/Plantilla-SRS-Requisitos-no-funcionales.es.html.

Bibliografía

BIBLIOGRAFÍA

1. **Fidanza, Jorge Alberto López.** *Firma Digital.* 2004.

2. **Devoto, Mauricio.** *La Firma Digital.*

3. **Carlino, Bernardo.** *Reuniones a Distancia.*

4. Proyecto de Firma Digital de la República de Argentina. [En línea] [Citado el: 27 de enero de 2009.] http://www.pki.gob.ar.

5. Corte constitucional de Colombia. [En línea] [Citado el: 27 de enero de 2009.] http://www.cortecontitucional.gov.co.

6. **Leiva, Renato Jijena.** *Federación Iberoamericana de Asociaciones de Informática y Derecho (FIADI) Firma Digital y Entidades Certificadoras.* Chile : s.n.

7. Legislación Informática de Panamá. [En línea] 3 de agosto de 2001. [Citado el: 29 de enero de 2009.] http://www.informatica-juridica.com/panama.asp.

8. Perú. [En línea] [Citado el: 29 de enero de 2009.] http://www.bernateygamboa.com/espanol/textos disponibles/comer elec/internacional/PE.htm..

9. DiarioLinux. [En línea] 12 de noviembre de 2008. [Citado el: 18 de febrero de 2009.] http://diariolinux.com/2008/11/12/sinadura-firma-digital-de-pdfs-en-linux.

10. javaHispano. [En línea] 10 de septiembre de 2007. [Citado el: 18 de febrero de 2009.] http://www.javahispano.org/contenidos/es/viafirma_org_11/?menuId=ANNOUNCEMENTS&onlypath=true.

11. eParapher. [En línea] 29 de enero de 2009. [Citado el: 18 de febrero de 2009.] http://maven.eparapher.com..

12. Scribd. [En línea] [Citado el: 20 de febrero de 2009.] http://www.scribd.com/doc/297224/RUP.

13. Free Download Manager. [En línea] [Citado el: 20 de febrero de 2009.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p.

14. NetBeans. [En línea] [Citado el: 15 de marzo de 2009.] http://www.netbeans.org.

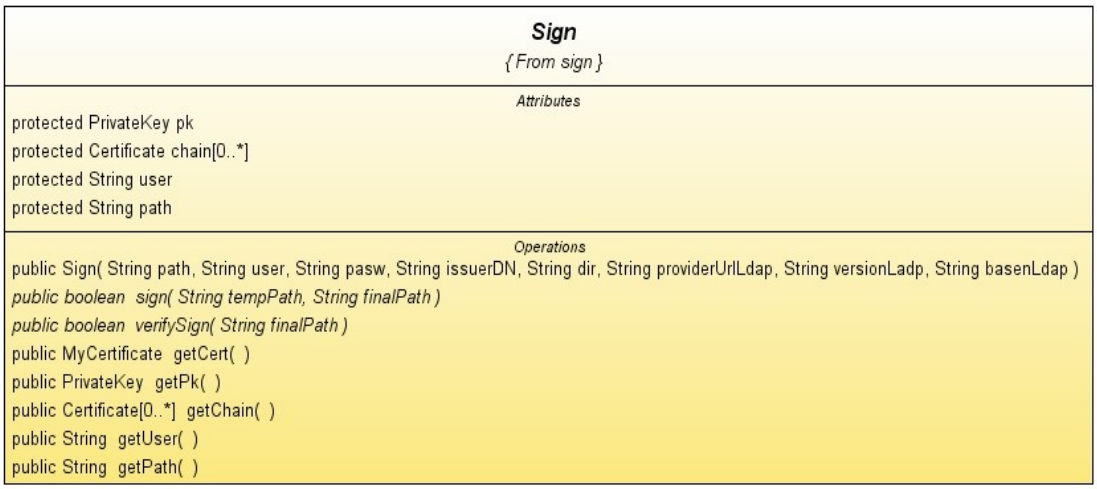
15. Alfresco. [En línea] [Citado el: 15 de marzo de 2009.] http://www.alfresco.com/es/media/coverage/2008/12/alfrescoiberiaroadshow5/avalon_raquel.pdf.

16. anetcom. [En línea] [Citado el: 20 de abril de 2009.] http://www.anetcom.es/servicios/consumidoryusuario/glosario.asp?ini=0.

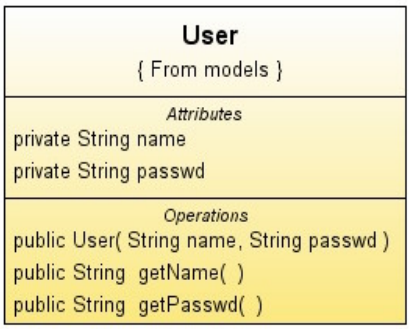
17. Linux Reflejo. [En línea] [Citado el: 20 de abril de 2009.] http://linuxreflejo.wordpress.com/2008/07/25/conceptos-de-criptografia-y-firma-digital.

18. aula-ee . [En línea] [Citado el: 20 de abril de 2009.] http://aula-ee.com/html/ca/temp/old/algoritmica/glosario.htm#c.

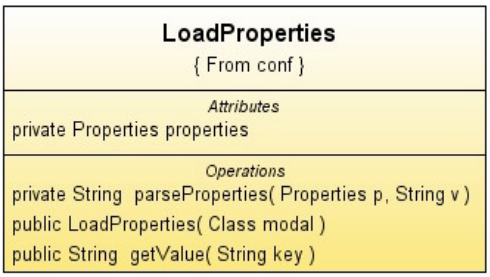
ANEXOS



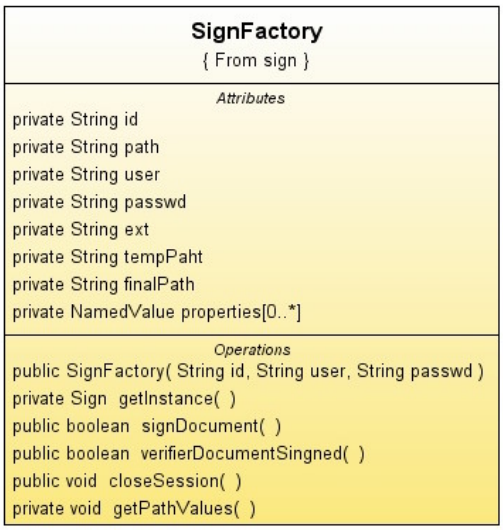
Anexo 1: Clase Sign



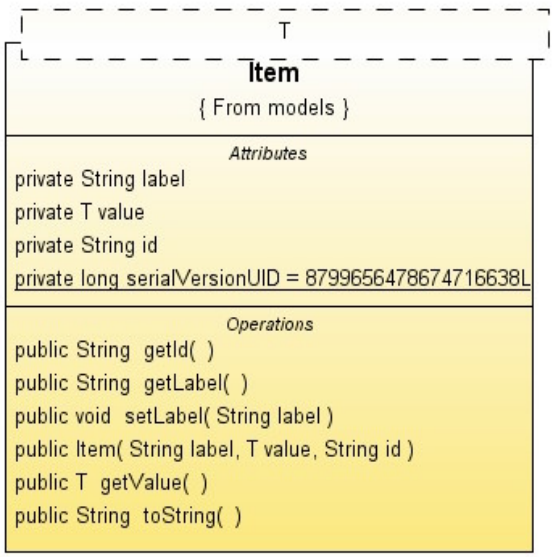
Anexo 2: Clase User



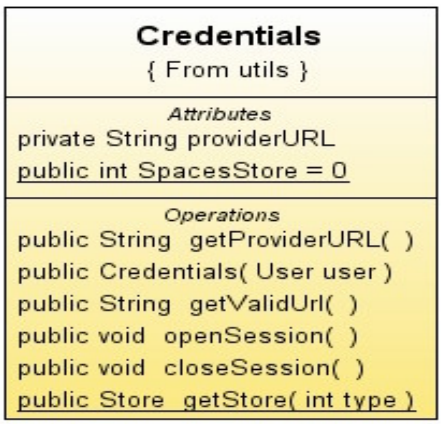
Anexo 3: Clase LoadProperties



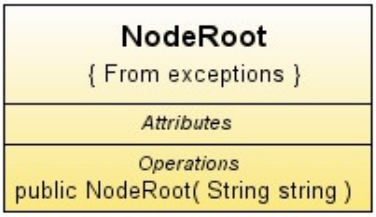
Anexo 4: Clase SignFactory



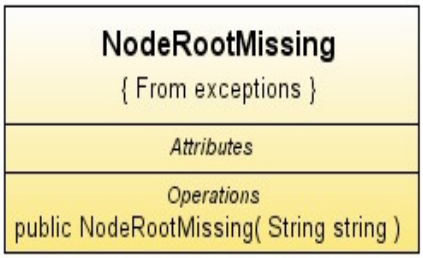
Anexo 5: Clase Item



Anexo 6: Clase Credentials

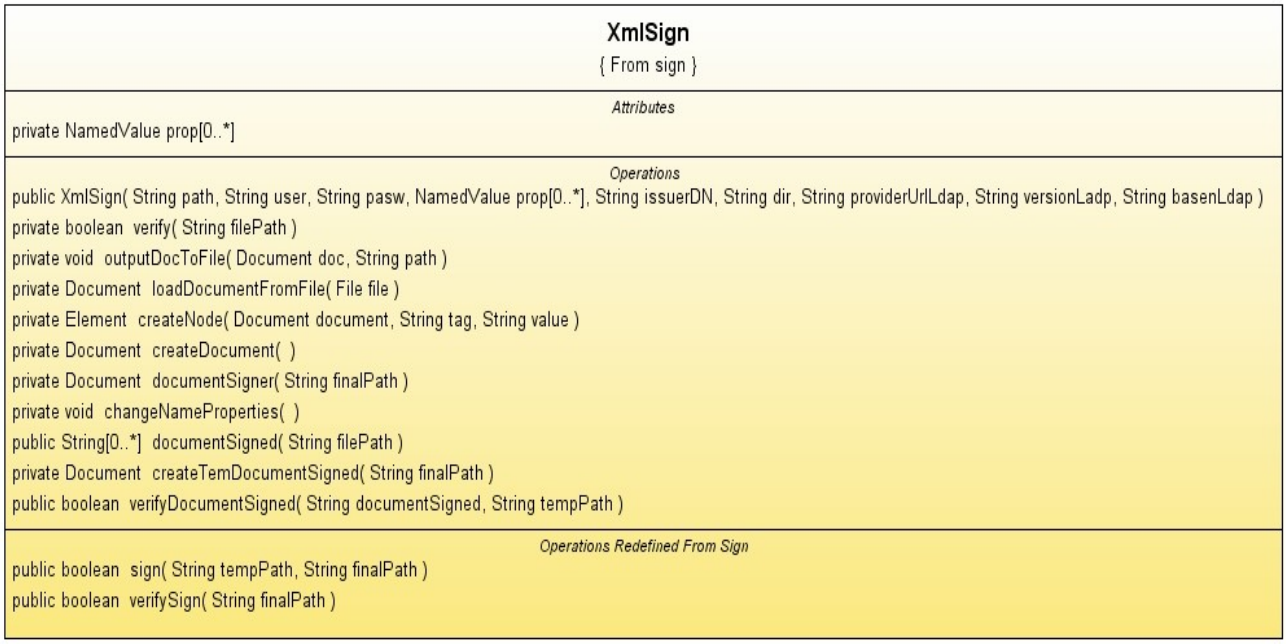


Anexo 7: Clase NodeRoot

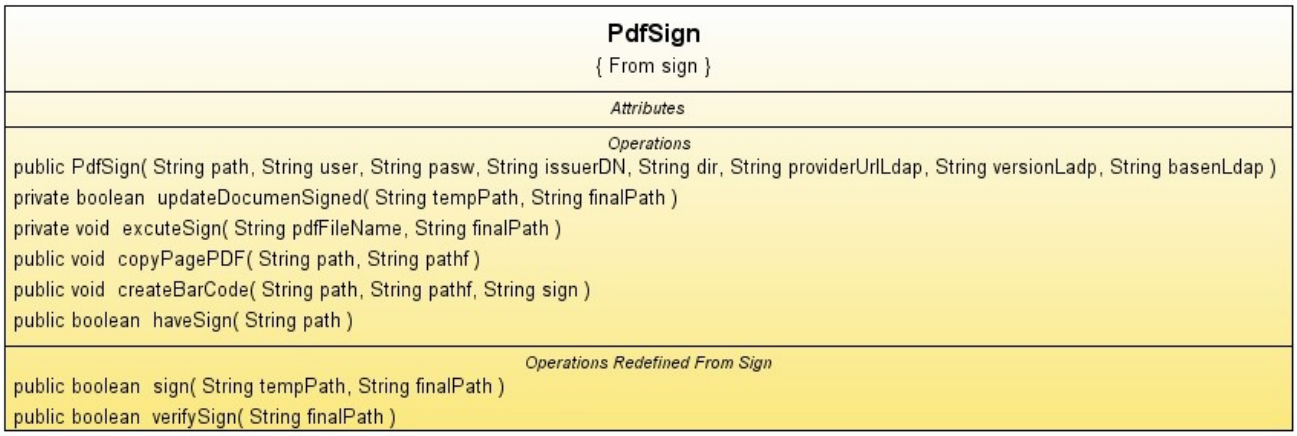


Anexo 8: Clase NodeRootMissing

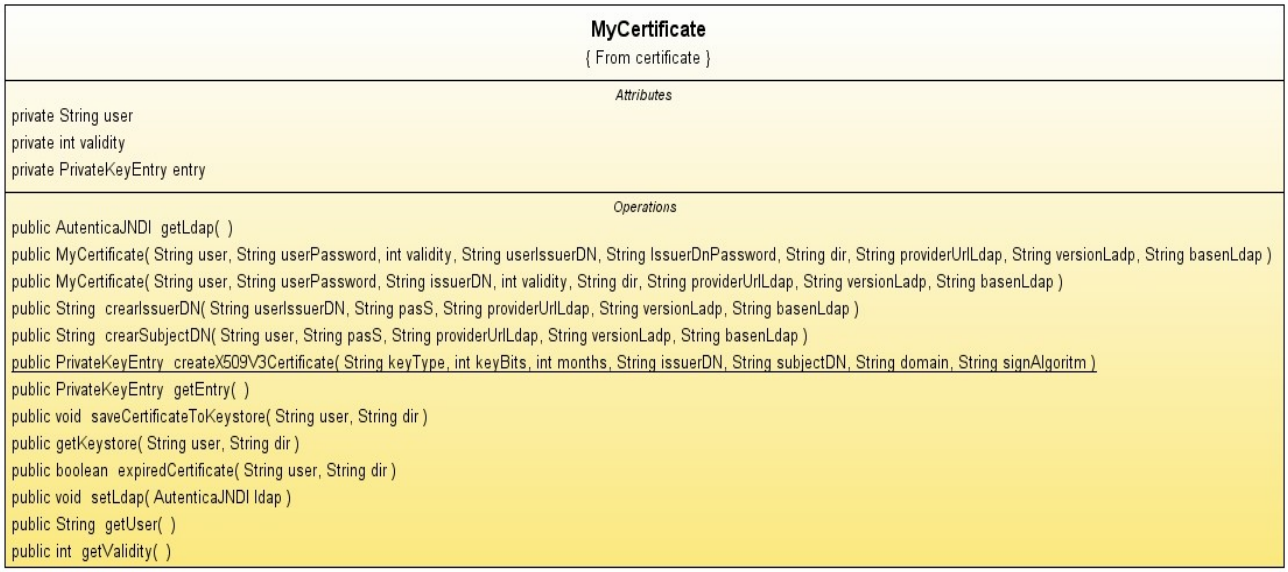
Anexo 9: Clase ContentHandler



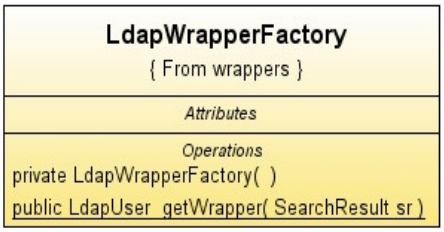
Anexo 10: Clase XmlSign



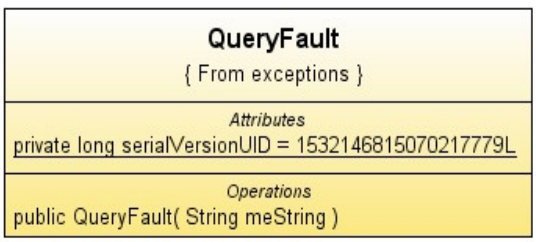
Anexo 11: Clase PdfSign



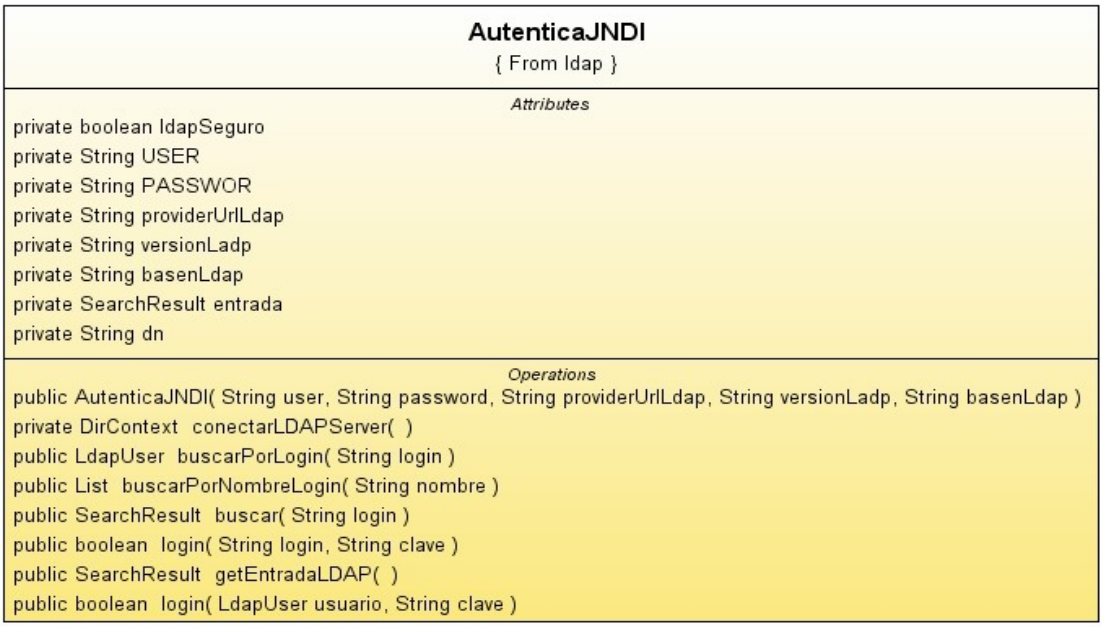
Anexo 12: Clase MyCertificate



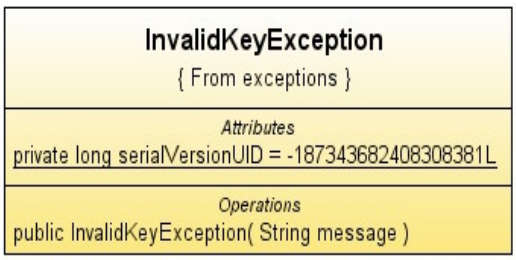
Anexo 13: Clase LdapWrapperFactory



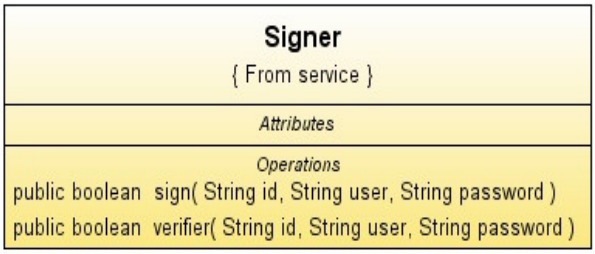
Anexo 14: Clase QueryFault



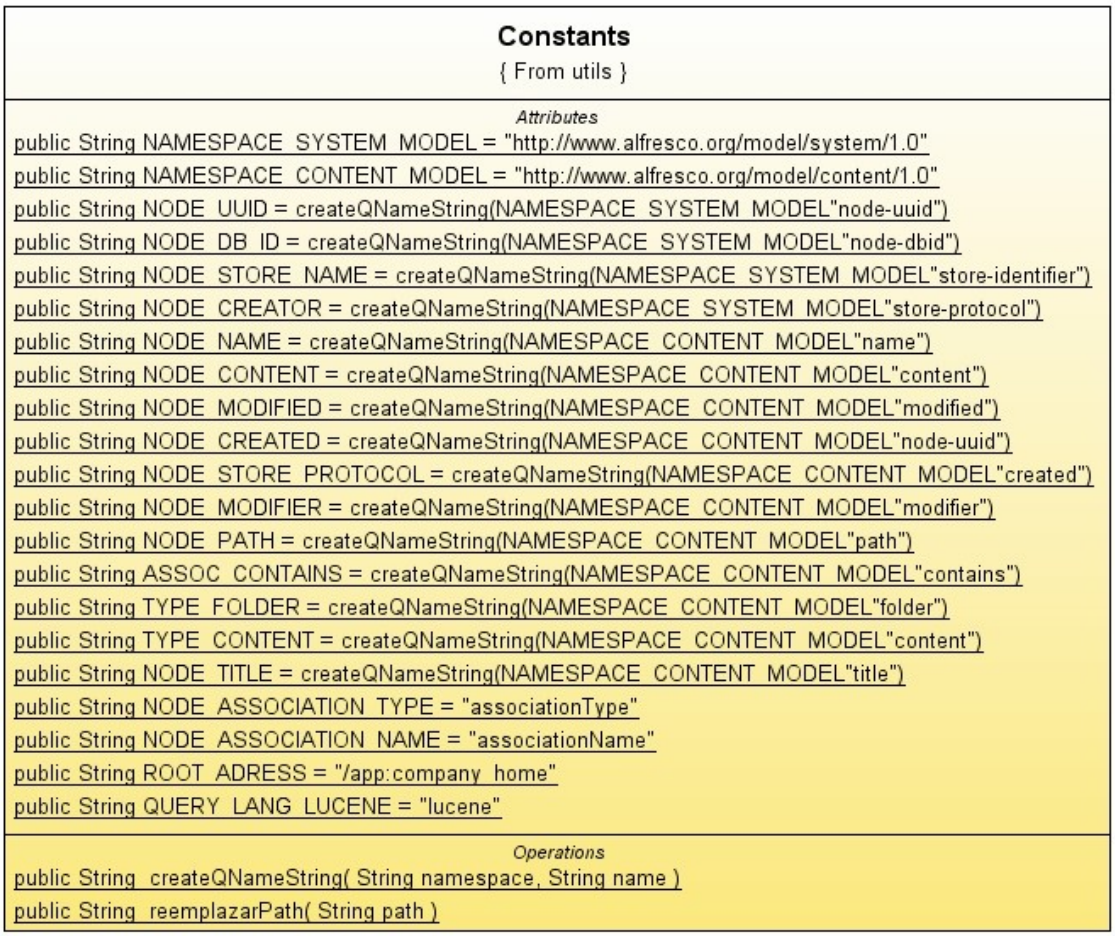
Anexo 15: Clase AutenticaJNDI



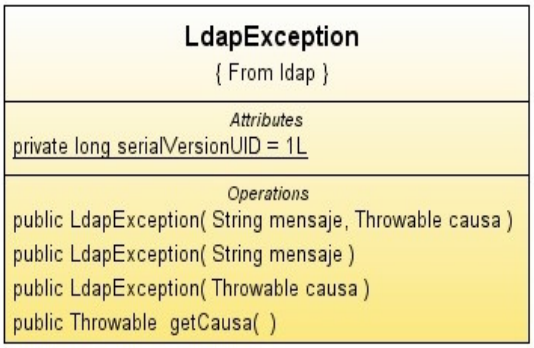
Anexo 16: Clase InvalidKeyException



Anexo 17: Clase Signer



Anexo 18: Clase Constants



Anexo 19: Clase LdapException

Anexo 20: Clase LdapUser

GLOSARIO DE TÉRMINOS

A:

Alfresco: es la alternativa de Código Abierto **para la gestión de contenido empresarial** (ECM), proporcionando gestión documental, colaboración, gestión de registros, gestión de información, gestión del contenido web e imágenes.

Auditabilidad: permite identificar y rastrear las operaciones llevadas a cabo por el usuario dentro de un sistema informático cuyo acceso se realiza mediante la presentación de certificados.

C:

Certificado: acreditación emitida por una entidad o un particular debidamente autorizados garantizando que determinado dato (por ejemplo, una firma electrónica o una clave pública) pertenece realmente a quien se supone.

Confidencialidad: característica de la información por la cual sólo puede ser revelada a los usuarios autorizados.

Criptografía: término formado a partir del griego kruptos, oculto. Significa, según el diccionario académico, "Arte de escribir con clave secreta o de un modo enigmático". Es criptográfico cualquier procedimiento que permita a un emisor ocultar el contenido de un mensaje de modo que sólo personas en posesión de determinada clave puedan leerlo, tras haberlo descifrado.

E:

Encriptar: forma de proteger una información determinada para que no pueda ser leída o modificada sin la utilización de una clave**.**

P:

Plug-in: es una aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica, generalmente muy específica, como por ejemplo servir como driver (controlador) en una aplicación, para hacer así funcionar un dispositivo en otro programa. Ésta aplicación adicional es ejecutada por la aplicación principal.

R:

Repositorio: es un término utilizado en el dominio de las herramientas CASE. El repositorio podría definirse como la base de datos fundamental para el diseño; no sólo guarda datos, sino también algoritmos de diseño y, en general, elementos software necesarios para el trabajo de programación.