Universidad de las Ciencias Informáticas
Facultad 1

Título: Extensiones de firma digital de documentos en formato XML y archivos JAR para la solución ADVsigner

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

Autora:

Zoraimi Díaz Cabrera

Tutores:

Ing. Félix Alejandro Prieto Carratalá

Ing. Luis Fernandez Leyva

"La Habana. Junio, 2012"

Declaración de autoría

Título de trabajo de diploma: Extensiones de firma digital de documentos en formato XML y archivos JAR para la solución ADVsigner.

Autora: Zoraimi Diaz Cabrera

Declaro que soy la única autora de la presente investigación y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma con carácter exclusivo.

Para que así conste se firma la presente acta a los __ días del mes de _____ del año 2012.

_____
Autora
Zoraimi Díaz Cabrera

_____
Tutor
Ing. Felix Alejandro Prieto Carratalá

_____
Tutor
Ing. Luis Fernandez Leyva

Dedicatoria

A mi padres, ustedes son el motor impulsor que me renuevan las fuerzas cuando estas parecen haberse agotado. Son mi razón de ser, mi mayor orgullo, mi fuente de inspiración. Les debo todo lo que soy y seré. El haber llegado hasta este punto es fruto de su dedicación y esfuerzo. Muchas gracias mami y papi. Los amo.

A mi hermana, sabes que te quiero como si fueras mi madre, gracias por cuidar de mí, por estar ahí cada vez que me haces faltas, con solo estar a tu lado mi vida se llena de felicidad. Para ti, nuestros padres y tus hijos va dedicado con mucho amor este trabajo de diploma.

Agradecimientos

Pienso que ésta sección, quizás, no sea suficiente para mencionar a todos aquellos que han puesto un poquito de empeño y su ayuda desinteresada para cumplir éste, mi sueño, y si existe alguien que no sienta su presencia en mis palabras, reciba mis más sinceras disculpas y tenga la certeza que siempre tendrá mi gratitud.

Primero que todo agradecer a mis padres que me llevaron de la mano mi primer día de escuela y, de cerca o de lejos, han cuidado cada uno de mis pasos hasta aquí. Por el esfuerzo que hacen para que yo sea una mejor persona, le estoy eternamente agradecida y algún día les retribuiré todo lo que han hecho por mí.

A mi hermana y hermano por hacer que mi vida se llene de felicidad y por regalarme sobrinos maravillosos.

A toda mi familia por siempre confiar y creer en mí, en especial a mi abuela Nito por todo el apoyo y las fuerzas que me da para seguir adelante.

A mis tutores Félix y Luis les agradezco la ayudada que me han dado para perfeccionar el trabajo de diploma, muchas gracias de verdad.

A mi amigo Dariel Silot por tener que cargar conmigo en estos últimos años, por su apoyo brindado en la realización de este trabajo y por ayudarme a convertirme en una mejor ingeniera.

A Laura y su familia por las veces que me acogieron en su casa.

A Alíen Góngora que apareció en los últimos meses de la carrera dándome sus puntos de vista, su colaboración para el desarrollo de esta investigación, su optimismo y lo más valioso su manera de ver la vida.

A Roberto Llerena un amigo que no olvido pues me tendió la mano varias veces explicándome las clases de programación una y otra vez hasta lograr entenderlas.

A mis compañeras de clase y apartamento, especialmente a los que estuvieron cerca durante estos seis años.

De manera general a todos que de una manera u otra han estado presentes en mi vida, por todo el apoyo y la ayuda prestada muchas gracias.

Resumen

La solución ADVsigner permite la firma digital de varios documentos simultáneamente, pudiendo ser utilizada en ambientes donde el volumen de documentación electrónica es abundante como: Universidades, Empresas, Ministerios, entre otros. Admite el uso de dispositivos seguros que almacenan claves y contenedores PKCS12[1] utilizados para la firma digital de los documentos y archivos electrónicos. Presenta una arquitectura basada en extensiones o plugins que permite extender el sistema con nuevas funcionalidades de firma y validación digital.

Este trabajo abarca la creación de extensiones o plugins utilizados para firmar y validar documentos XML[2] y archivos JAR[3] a través de la solución ADVsigner, garantizando de esta manera la autenticación de la identidad del firmante, la integridad de la información y el no repudio de los datos firmados. Para realizar estas extensiones o plugins de firma digital fue necesario satisfacer las restricciones de diseño e implementación del marco arquitectónico utilizado en la solución ADVsigner, además de recurrir a una metodología de desarrollo de software y utilizar las herramientas que posibilitan generar los artefactos necesarios para el diseño y la posterior implementación de la solución propuesta.

Palabras Clave: ADVsigner, archivos JAR, documentos XML, firma digital, plugin o extensiones.

[1] Define un formato de fichero usado comúnmente para almacenar claves privadas con su certificado de clave pública protegida mediante clave simétrica ejemplo de esto son los ficheros P12 y PFX.
[2] Extensible Markup Language
[3] Java Archive

Índice de contenido

Índice de tablas

Índice de figuras

Introducción

Con el surgimiento de Internet se han abierto nuevas posibilidades para el intercambio de información. Las diversas formas de transmitir un documento electrónico y la variedad de tipologías de los mismos es cada día mayor. Al mismo tiempo, son cada vez mayores las amenazas relacionadas con la seguridad de los datos, la suplantación de identidad y la negación de la información trasmitida. Por lo que es necesario contar con un mecanismo que dé solución a los problemas antes mencionados, teniendo en cuenta que los documentos electrónicos pueden ser transmitidos por diferentes medios o canales de comunicación, ya sean seguros o inseguros.

La firma digital constituye el mecanismo esencial para establecer confianza en el intercambio de información a través de Internet (Pásaro Méndez, 2006). Una firma digital es un conjunto de datos asociados a un mensaje que permite garantizar la identidad del firmante, la integridad del mensaje y el no repudio del que originó el documento (SGP, 2011). Es utilizada generalmente para la distribución de software, transacciones financieras, comercio electrónico y otras áreas donde es importante detectar la falsificación y la manipulación de documentos electrónicos. Uno de los documentos electrónicos de mayor manipulación en estas áreas son los formatos de documentos portátiles (PDF) desarrollado por la empresa Adobe Systems. Un documento PDF es un estándar de la *International Organization for Standardization* (ISO) 32000 que permite almacenar y revisar información desde cualquier aplicación y en cualquier sistema informático. Gracias al lanzamiento del software Adobe Acrobat, los PDF son más seguros y dinámicos que nunca logrando cifrar e incorporar firmas digitales. Por ello, personas, empresas y administraciones públicas de todo el mundo confían en los archivos PDF para transmitir sus ideas y puntos de vista (Adobe Systems, 2012).

Otro de los documentos utilizados en estas áreas son los documentos XML, diseñados específicamente para el almacenamiento e intercambio de información a través de Internet (W3C, 2003). XML es un formato basado en texto, flexible y de fácil uso que sirve de base para el desarrollo de múltiples formatos y tecnologías como por ejemplo: *Really Simple Syndication* (RSS), *Simple Object Access Protocol* (SOAP) y *eXtensible HyperText Markup Language* (XHTML) (Robin Cover, 2005), además se ha convertido en la opción predeterminada para muchos productos de oficina, tales como: herramientas de Microsoft Office, OpenOffice y LibreOffice hacen uso de este estándar, sin embargo debido al crecimiento explosivo que tiene

en diversas áreas se hace necesario utilizar un mecanismo que permita garantizar autenticidad, integridad y no repudio de la información transmitida en ellos.

Un formato muy utilizado además, pero específicamente en entornos de desarrollo de software, son los archivos Java (JAR). Un archivo JAR es un formato multiplataforma que se utiliza para agrupar clases en el lenguaje Java, otorgándole seguridad y compresión. Con la nueva tendencia adoptada en la actualidad acerca de producir y utilizar herramientas en software libre los archivos JAR han tenido un crecimiento considerable, por lo que es importante comprobar que la información contenida en ellos no se encuentre modificada por terceros. Para resolver esto se utiliza el mecanismo antes mencionado firma digital que garantiza autenticidad, integridad y no repudio, no solo de los datos transmitidos y almacenados en los archivos JAR sino también en otros tipos de documentos electrónicos como es el caso de los PDF y XML.

Son varios los países que han aprobado la firma digital como un medio legal, otorgándole validez jurídica equivalente a la que proporciona la firma manuscrita, como son: Argentina, Venezuela, Puerto Rico, Uruguay, Brasil, Chile, Ecuador, España, entre otros países (Secretaría Nacional De Administración Pública Ecuador, 2009). En Cuba no existe legislación al respecto, pero pueden encontrarse investigaciones y publicaciones en universidades cubanas que se centran en el tema de la firma digital (Vlamir Rodríguez Fernández, 2011). Un ejemplo de esto es la Universidad de las Ciencias Informáticas (UCI) en la cual se encuentra el Centro de Identificación y Seguridad Digital (CISED) creado con el objetivo de desarrollar productos, servicios y soluciones integrales en el campo de la identificación y la seguridad digital, de alta confiabilidad y eficiencia económica.

Una de las soluciones creadas por el centro es ADVsigner, solución de firma digital, multiplataforma, ofrece un mecanismo basado en plugins que permite incorporar nuevas funcionalidades de firma y validación digital al sistema sin modificar el código base de la aplicación (Silot Ochoa, 2011). Esta solución gestiona de forma centralizada certificados digitales, cuenta con un almacén de llaves propias donde se pueden realizar diferentes operaciones, entre las que se encuentran la gestión y validación de los certificados digitales. ADVsigner permite firmar documentos electrónicos haciendo uso de un contenedor de clave o tarjeta inteligente, al mismo tiempo que aplica el sellado de tiempo en las firmas digitales, pero este sistema solo posee una extensión o plugin de firma digital utilizada para firmar documentos en formato PDF, lo que impide aplicar la firma digital a otros documentos y archivos electrónicos como es el caso de los documentos XML y archivos JAR, por lo que se

hace necesario agregar nuevos formatos de firma a la solución ADVsigner para garantizar autenticidad, integridad y no repudio de la información transmitida en estos documentos y archivos electrónicos. Con el análisis de lo antes expuesto y con el fin de darle solución a la problemática existente, este trabajo plantea el siguiente **problema de la investigación:**

¿Cómo firmar digitalmente documentos en formatos XML y archivos JAR a través de las funcionalidades de la solución ADVsigner para garantizar autenticidad, integridad y no repudio de la información transmitida en este tipo de documento y archivo electrónico?

El **objeto de estudio** queda enmarcado en el proceso de firma digital como herramienta de apoyo en estrategias de seguridad. Por lo que el **objetivo general** de esta investigación consiste en desarrollar extensiones o plugins de firma digital de documentos en formato XML y archivos JAR a través de las funcionalidades de la solución ADVsigner utilizando el estándar de firma digital XML y la especificación de los JAR de Java para garantizar autenticidad integridad y no repudio de la información transmitida en este tipo de documento y archivo electrónico

A partir del análisis del objetivo general se derivaron los siguientes **objetivos específicos:**

- Seleccionar las tecnologías, metodologías y herramientas a usar en el desarrollo del software.
- Especificar lo requisitos funcionales y no funcionales de las extensiones a desarrollar.
- Implementar la especificación estándar de firma digital de XML para extender la solución ADVsigner con las funcionalidades de firma de documentos en formato XML.
- Implementar el esquema de firma digital de archivos JAR para extender el software ADVsigner con las funcionalidades de firma de este tipo de archivo.
- Aplicar pruebas de integración, aceptación y unitarias los plugins desarrollados.

Derivado de la relación entre el problema de la investigación, objeto de estudio y objetivos generales el siguiente **campo de acción** consiste en el proceso de firma digital de documentos electrónicos en formato XML y archivos JAR.

Para el avance exitoso de la investigación se utilizan los siguientes **métodos investigativos:**

Métodos teóricos:

- Analítico-sintético: Mediante este método se identifican los conceptos básicos del proceso de firma digital y las principales tecnologías a usar. Esto permite un análisis ordenado de los conceptos identificados dentro del alcance de la investigación, además de elaborar una estructura adecuada del documento a partir de la información obtenida.

- Inductivo-Deductivo: Se generan deducciones para llegar a tener una visión clara de lo que se quiere hacer y adquirir así nuevos conocimientos sobre el funcionamiento de la firma digital y sus posibles implementaciones.

- Modelación: Este método permite realizar representaciones de la realidad mediante gráficos, diagramas y modelos construidos durante el proceso de desarrollo de las extensiones o plugins de firma digital.

Además de la utilización del siguiente **método empírico:**

- Observación: A través de este método se le presta atención a cada fase del proceso de desarrollo del software y se toma experiencia de cada tarea para aplicarse en otras. Además recoge la información de cada uno de los conceptos o variables definidas en el problema de la investigación para posteriormente, según lo observado, poder caracterizar y conocer el funcionamiento del sistema en su totalidad.

- Medición: Este método se refleja durante la utilización de técnicas de pruebas que arrojan como resultado los tiempos de respuesta de los plugins y la cantidad de líneas de código revisadas.

- Entrevista: Este método tiene como objetivo obtener información acerca del funcionamiento de la solución ADVsigner mediante consultas y entrevistas planificadas con especialistas y expertos en el tema de la firma digital en el CISED.

La investigación de este trabajo se basa en la confección de extensiones o plugins de firma digital de documentos electrónicos en formato XML y archivos JAR. Estos plugins son creados para suministrar y proveer a la solución ADVsigner con nuevas funcionalidades de firma y validación digital garantizando de esta manera la autenticación, integridad y no repudio de la información transmitida en documentos XML y archivos JAR. Concebido desde sus inicios para

firmar múltiples formatos de documentos electrónicos, pero aplicado en la práctica sólo posee una extensión o plugins de firma digital.

El contenido de la presente investigación está estructurado en tres capítulos, los cuales se describen a continuación:

Capítulo 1: Fundamentación teórica

Este capítulo presenta los elementos teóricos que soportan el trabajo realizado. Además, se realiza un estudio de los lenguajes, herramientas, marco de trabajo, entorno de desarrollo y metodología empleada durante el desarrollo de la propuesta de solución.

Capítulo 2: Propuesta de solución

En este capítulo se realiza un análisis general del sistema propuesto. Se presenta la fase de exploración y planificación especificada por la metodología XP, identificando los requisitos funcionales y no funcionales, además de realizar el plan de iteraciones y plan de entregas.

Capítulo 3: Implementación y prueba

En este capítulo se procede a desarrollar la implementación de los plugins diseñados, obteniendo los artefactos propios de la metodología seleccionada y finalmente se efectúan las pruebas unitarias, integración y aceptación necesarias para validar la propuesta de solución.

Capítulo I: Fundamentación teórica

En este capítulo se realiza una descripción de los conceptos que se consideran esenciales para el entendimiento del problema inicial y la solución propuesta, con el objetivo de construir las bases para comprender el proceso de firma digital de documentos XML y archivos JAR en el contexto de la situación problemática de la investigación. También se exponen las principales características de las herramientas propuestas para resolver el problema planteado, así como un estudio de las metodologías más utilizadas en el desarrollo de software.

1.1 Firma digital de documentos electrónicos.

La firma digital es un conjunto de datos electrónicos cifrados mediante un clave privada que identifican a una persona en concreto. Suelen unirse al documento de forma que el receptor del mensaje esté seguro de quién ha sido el emisor, así como que el mensaje no ha sido alterado o modificado, pudiendo comprobar la validez de la firma (Dra. María José Ruiz, 2003) (ver la Figura 1). El cifrado de datos se realiza utilizando algoritmos asimétricos o de llave pública, los más conocidos son: RSA[4], ElGamal[5], *Digital Signature Algorithm* (DSA)[6] y Diffie-Hellman[7] (Gobierno de Buenos Aires, 2010). Utilizando la criptografía de llave pública se ha creado una serie de herramientas para firmar digitalmente documentos electrónicos, según (Silot Ochoa, 2011) entre las aplicaciones de escritorio que se destacan en el ámbito de la firma digital se encuentran: Adobe Acrobat Profesional[8], E-Lock ProSigner[9] y Sinadura[10]; además dicho autor

[4] Desarrollado en 1977 por Ron **R**ivest, Adi **S**hamir, y Len **A**dleman. Es el algoritmo de llave pública más utilizado y es válido tanto para cifrar como para firmar digitalmente.
[5] Esquema de firma digital raramente utilizado en la práctica. Con más frecuencia se utiliza una de sus variantes llamada algoritmo de firma digital (DSA).
[6] Es un estándar del Gobierno Federal de los Estados Unidos de América o FIPS para firmas digitales desarrollado en 1991, como su nombre lo indica, sirve para firmar y no para cifrar información. Una desventaja de este algoritmo es que requiere mucho más tiempo de cómputo que RSA.
[7] Fue el primer algoritmo asimétrico desarrollado en 1976. Se emplea generalmente como medio para acordar claves simétricas que serán empleadas para el cifrado de una sesión.
[8] http://www.adobe.com/products/acrobatpro/features.html/. Es el lector de ficheros PDF más completo del mercado. Además, se pueden crear formularios en los documentos y darles seguridad a través de la firma digital y el cifrado. La desventaja está dada porque al ser una aplicación profesional su costo es elevado.
[9] http://www.elock.com/prosigner.html/. Software de escritorio para la firma y cifrado de documentos electrónicos. El mismo facilita a usuarios incrustar firmas visibles en documentos Word, Excel y PDF. Es un software propietario de costo elevado.
[10] http://www.sinadura.net/. Sinadura es un proyecto opensource orientado a ofrecer productos y servicios para la identidad digital y firma electrónica tanto para particulares como empresas.

plantea que los documentos en formatos PDF son los documentos electrónicos más firmados, muchas de las herramientas son propietarias y las que no lo son, firman solamente el formato antes mencionado, sin poder incorporar nuevos formatos de firma.

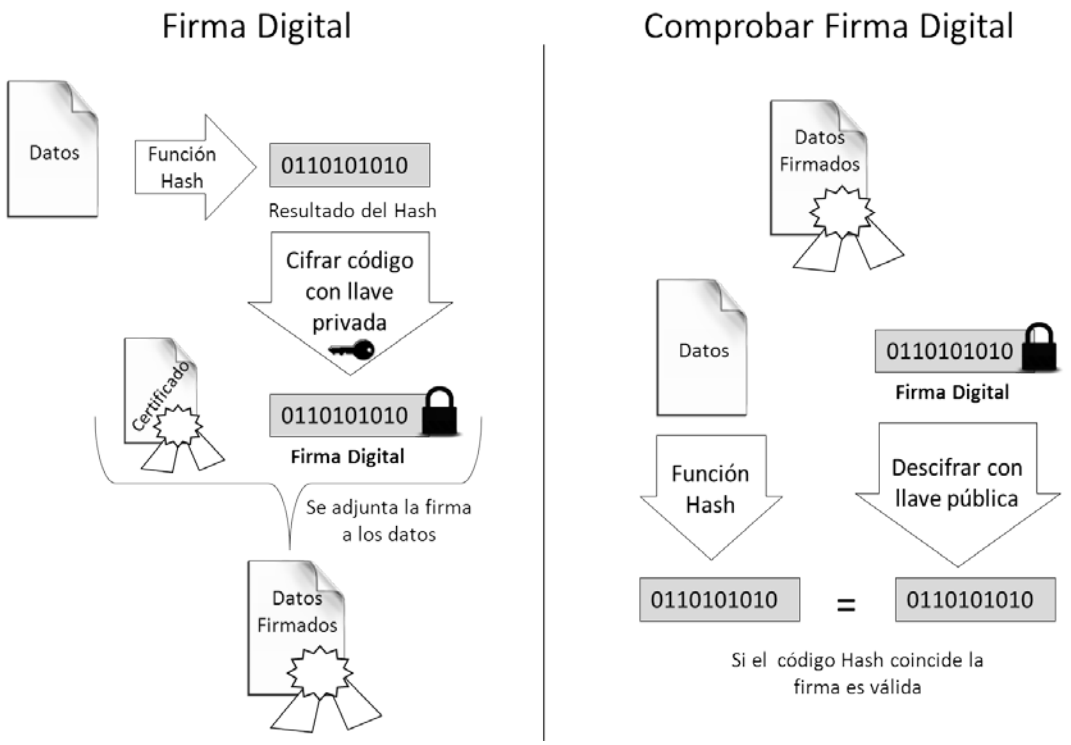


Figura 1: Proceso de firma y validación digital

Fuente: (Elaboración propia)

1.2 Solución de firma digital ADVsigner

La solución de firma digital ADVsigner fue creada con el objetivo de garantizar la autenticidad, integridad y el no repudio de la información contenida en documentos PDF. Actualmente este sistema cuenta con un almacén propio de llaves criptográficas, donde se puede gestionar claves privadas y certificados digitales ubicados en un contenedor de claves implementando el estándar PKCS[11]12, ejemplo de esto son los archivos P12 o PFX o utilizando una tarjeta

Firma digital documento PDF aunque una de sus últimas versiones permite firmar otros formatos de documentos o archivos electrónicos.

[11] PKCS (Public-Key Cryptography Standards): Grupo de estándares de criptografía de llave pública desarrollados por RSA Security Labs.

inteligente, implementando el estándar PKCS11[12]. Posee funcionalidades como la validación de certificados digitales y la firma simultánea de documentos electrónicos. Esta solución multiplataforma desarrollada en Java, posee una arquitectura basada en plugins que permite extender fácilmente el sistema añadiéndole nuevas funcionalidades de firma y validación digital sin modificar el código base, haciéndola diferente de otras herramientas similares. Para firmar documentos de diferentes formatos es obligatorio contar con una extensión o plugin que permita firmar un documento en específico, pues ADVsigner no cuenta con esta funcionalidad, solo gestiona y valida los certificados que tiene ubicado dentro de su almacén de llaves.

Según (Silot Ochoa, 2011) la arquitectura de la solución ADVsigner está basada en plugins con un estilo de Llamada y Retorno y el plugin aplica un sub-estilo del programa principal. Lo que posibilita al diseñador de software conseguir una estructura del sistema fácil de modificar y escalar. Al tener un sub-estilo del programa principal, las funciones se descomponen en una jerarquía de control donde el programa principal invoca los programas subordinados, los cuales a su vez pueden invocar a otros. La misma se divide en dos partes principales; la aplicación y el módulo plugin (ver Figura 2).

Arquitectura de la aplicación

GUI: Esta capa contiene las clases correspondientes a la interfaz gráfica de usuario. Son clases de uso general y son independientes del plugin o el documento a firmar.

CORE: Esta capa contiene paquetes principales para cubrir las necesidades de la aplicación.

- **KEYSTORES:** Este paquete se encarga de la gestión de los distintos certificados utilizados en el proceso de firma que pueden estar contenidos en el almacén de llaves de la aplicación o en una tarjeta inteligente.
- **SIGNER:** Este paquete contiene las clases encargadas de relacionar cada documento con el plugin adecuado para su firma y validación.
- **PLUGINS:** Este paquete contiene las clases necesarias para el manejo de plugins en la aplicación.

LIBRARIES: Librerías necesarias para el funcionamiento de la aplicación. Estas pueden ser de terceros o las que proporciona la Java Platform, Standard Edition (J2SE).

[12] PKCS11: Define un API genérico de acceso a dispositivos criptográficos como por ejemplo tarjetas inteligentes (smartcard).

COMMON: Esta capa contiene clases comunes que son utilizadas por las capas antes mencionadas.

- **CONFIG (Configuration):** Este paquete contiene clases relacionadas con la configuración de la aplicación.

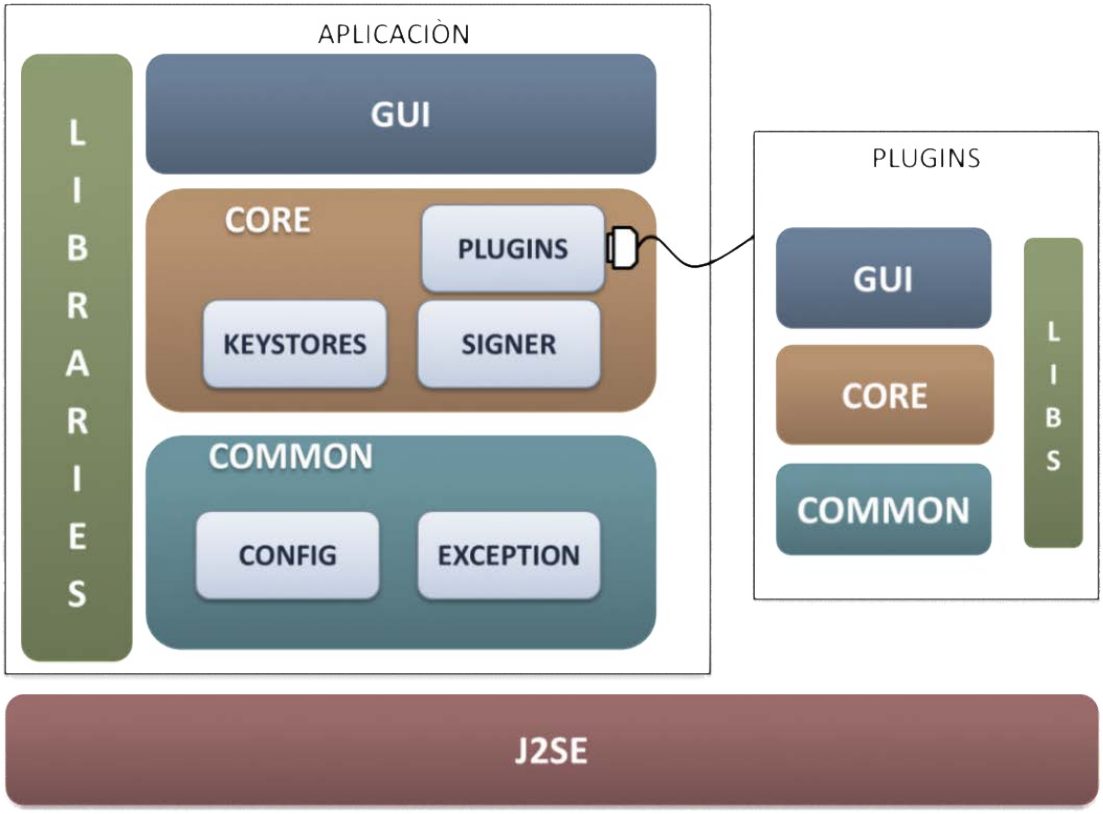


Figura 2: Arquitectura de la solución de firma digital ADVsigner

Fuente: (Silot Ochoa, 2011)

A continuación se detallan cada una de las capas en las que se compone el módulo plugin.

GUI: Esta capa contiene las clases correspondientes a la interfaz gráfica de usuario. Son clases dependientes del plugin y ofrecen opciones para manipular atributos del documento asociado al plugin.

CORE: Esta capa contiene la parte del proceso de firma y validación de documentos electrónicos.

COMMON: Clases comunes que son utilizadas para la configuración del plugin.

LIBS: Librerías necesarias utilizadas por los plugins. Estas pueden ser de terceros o las que proporcionan la J2SE.

J2SE: Plataforma de Java necesaria para la ejecución y desarrollo del sistema de firma digital del CISED.

En general ADVsigner posee algunas ventajas con respecto a los sistemas anteriormente referenciados entre las que se puede mencionar: es un sistema multiplataforma, permite la firma simultánea de documentos, facilidad de uso, basado en software libre, proporciona seguridad y validez legal a la información contenida en los documentos electrónicos y la principal característica que lo identifica es la escalabilidad y modularidad. Por lo que conviene tener en cuenta la posibilidad de extender la firma digital de la solución con nuevos formatos de documentos y archivos electrónicos.

1.3 Lenguaje Extensible de Marcas (XML)

XML es un formato basado en texto, específicamente diseñado para almacenar y transmitir datos (W3C, 2003). Estos documentos siguen una estructura jerárquica donde la etiqueta inicio es el elemento raíz, la etiqueta fin es la que cierra el documento y los datos comprendidos entre ambas, son los elementos o nodos que conforman un documento XML, logrando tener más elementos entre ellos. Para que un documento realmente cumpla con las normas de XML no solo debe respetar esta estructura jerárquica sino también debe incluir elementos obligatorios como la versión del XML utilizado y el tipo de codificación de lenguaje o prólogo. En la Figura 3 se muestra un ejemplo de la estructura de los documentos XML.

Los documentos XML se pueden usar para infinidades de trabajos y aportan muchas ventajas en amplios escenarios. Es muy utilizado en aplicaciones web, en la comunicación y migración de datos. Su implementación fácil, sencillez y flexibilidad han permitido que el XML tenga un crecimiento explosivo en el comercio electrónico. Pero con este crecimiento también se han incrementado los riesgos de su seguridad. Por ende se han desarrollado varios estándares de seguridad XML que se implementan hoy en día, tales como; firmas digitales con XML, cifrado XML y seguridad de los Servicios Web[13] (VeriSign, 2011).

[13] Un Servicio Web es una interfaz modular y auto-describible que puede publicarse, localizarse e invocarse a través de la red, mediante el intercambio de mensajes XML estandarizados, basados fundamentalmente en estándares y protocolos abiertos, tales como: XML y SOAP.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Etiquetas>
<Datos>
<Elementos Atributo="Valor del atributo">
            Los elementos pueden tener atributos,
            Que son una manera de incorporar características
            o propiedades a los elementos de un documentos.
</Elementos>
</Datos>
</Etiquetas>
```

Figura 3: Estructura de un documento XML

Fuente: (Elaboración propia)

1.3.1 Estándar de Firma Digital XML.

El estándar de firma digital XML (también llamado XMLDsig, DSig XML, XML-Sig) es una recomendación de la *World Wide Web Consortium* (W3C) y el *Internet Engineering Task Force* (IETF) que define las reglas de sintaxis y procesamiento para la creación y representación de las firmas digitales XML. Las firmas XML se pueden utilizar para firmar datos o recursos de cualquier tipo, normalmente documentos XML, pero cualquier objeto que sea accesible a través de una URI puede firmarse. Según (W3C, 2008) “las firmas XML se aplican a los objetos de datos del contenido digital a través de un direccionamiento indirecto. Los objetos de datos son digeridos, el valor resultante se coloca en un elemento (con otra información) y dicho elemento se firma criptográficamente”. La firma digital XML está representada por el elemento <Signature> que tiene la siguiente estructura (W3C, 2008) (ver Figura 4).

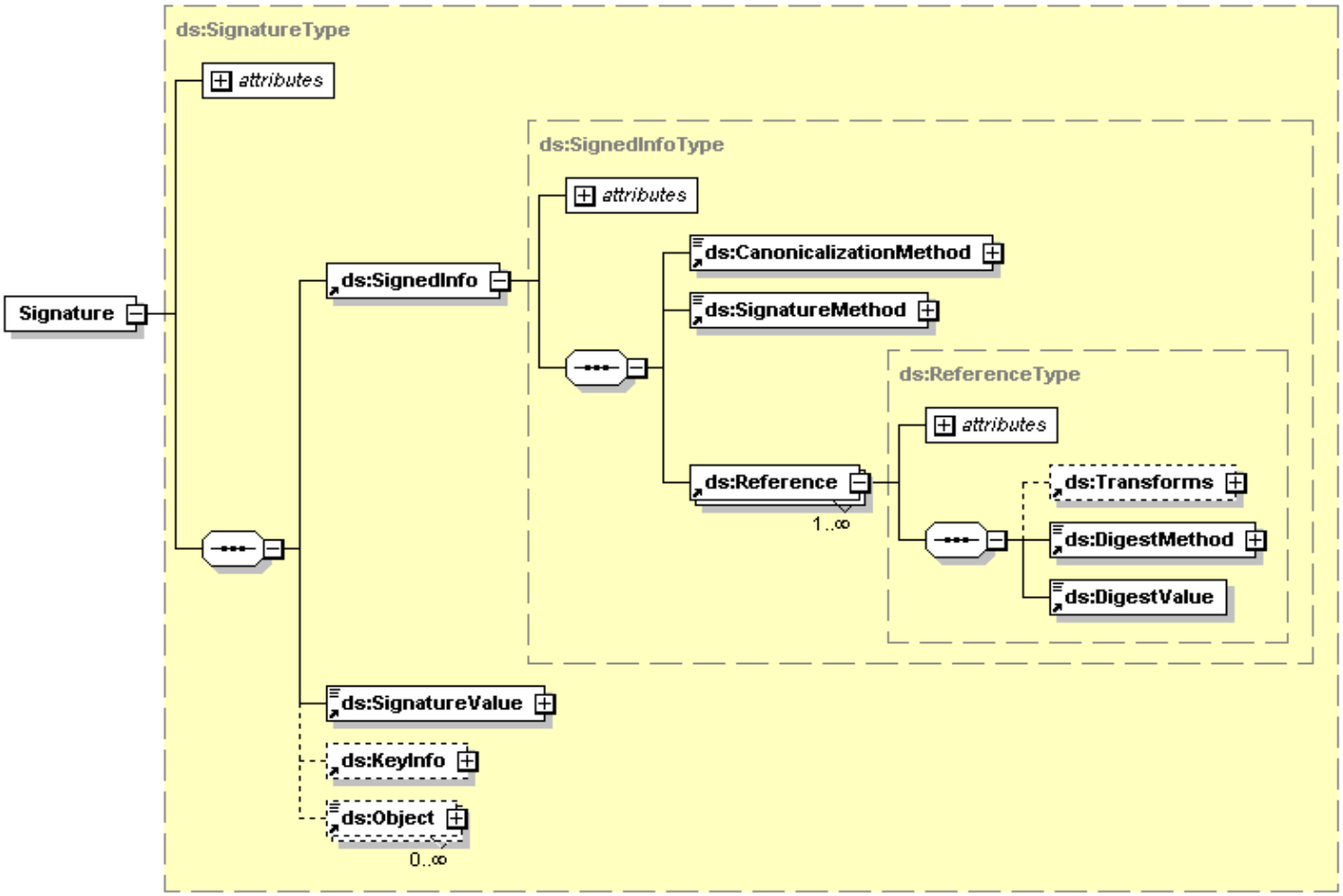


Figura 4: Estructura de la Firma Digital XML

Fuente: (Apache, XMLDsig-core-schema.xsd)

<SignedInfo>: Este elemento contiene la información necesaria para crear y validar la firma, encapsulando que es lo que se firma y como se firma.

- <CanonicalizationMethod>: Este método se utiliza para canonicalizar el <SignedInfo> antes de realizar la firma. Es el proceso de convertir el contenido XML en una representación física, llamada forma canónica[14], utilizada para eliminar cambios sutiles que pueden invalidar una firma digital sobre esos datos.
- <SignatureMethod>: Es el método utilizado para calcular el valor de la firma digital, convirtiendo la forma canónica del <SignedInfo> en la <SignatureValue> que es el resultado de la firma.

[14] La forma canónica de un documento XML es una manera de mostrar una representación física que asegura que si dos documentos XML son aparentemente diferentes, pero sus canónicos son iguales, los dos documentos son equivalentes.

- <Reference>: También este elemento se incluye dentro de <SignedInfo> contiene las referencias a los objetos que se van a firmar mediante una URI. Incluye además los siguientes elementos:
    - <Transforms>: Es cualquier transformación que vaya a aplicarse al recurso antes de firmar, esto se hace de forma opcional.
    - <DigestMethod>: Específica el algoritmo hash antes de aplicar la firma.
    - <DigestValue>: Contiene el resultado de aplicar el algoritmo hash al (o los) recurso (s) transformado (s).

<SignatureValue>: Contiene el resultado de una firma codificada en Base64 generada con los parámetros especificados en el elemento <SignatureMethod> del elemento <SignedInfo> después de aplicar el algoritmo especificado por el <CanonicalizationMethod>.

<KeyInfo>: Es un elemento opcional que indica la clave que ha de utilizarse para validar la firma. Dentro de los elementos que incluye se puede mencionar: nombre de las claves <KeyName>, algoritmos utilizados para la firma <KeyValue>, información sobre certificados <X509Data>, entre otros datos opcionales que se pueden añadir.

<Object>: Es un elemento opcional para la inclusión de datos en el elemento de la firma o en otro lugar.

Generación y validación de la firma XML

A continuación se listan pasos para generar el(los) elemento(s) <Reference>, y el elemento <SignatureValue> a partir de <SignedInfo> (Gobierno de Chile, 2011), en la Figura 5 se muestra cómo se genera una firma digital XML.

Generación de las referencias

Para cada objeto de datos a ser firmado:

1. Aplicar <Transforms> al objeto.

2. Calcular el valor de síntesis sobre el objeto resultante.

3. Crear el elemento <Reference>, incluyendo <DigestMethod>, <DigestValue>, y (de forma opcional) <Transforms>.

Generación de la firma

1. Crear el elemento <SignedInfo> con <CanonicalizationMethod>, <SignatureMethod>y <References>.

2. Calcular el valor de <SignatureValue> sobre <SignedInfo>, a partir de los algoritmos especificados en este último.

3. Construir el elemento <Signature> que incluya <SignedInfo>, <SignatureValue>, <Objects> y <KeyInfo>, siendo los dos últimos opcionales.

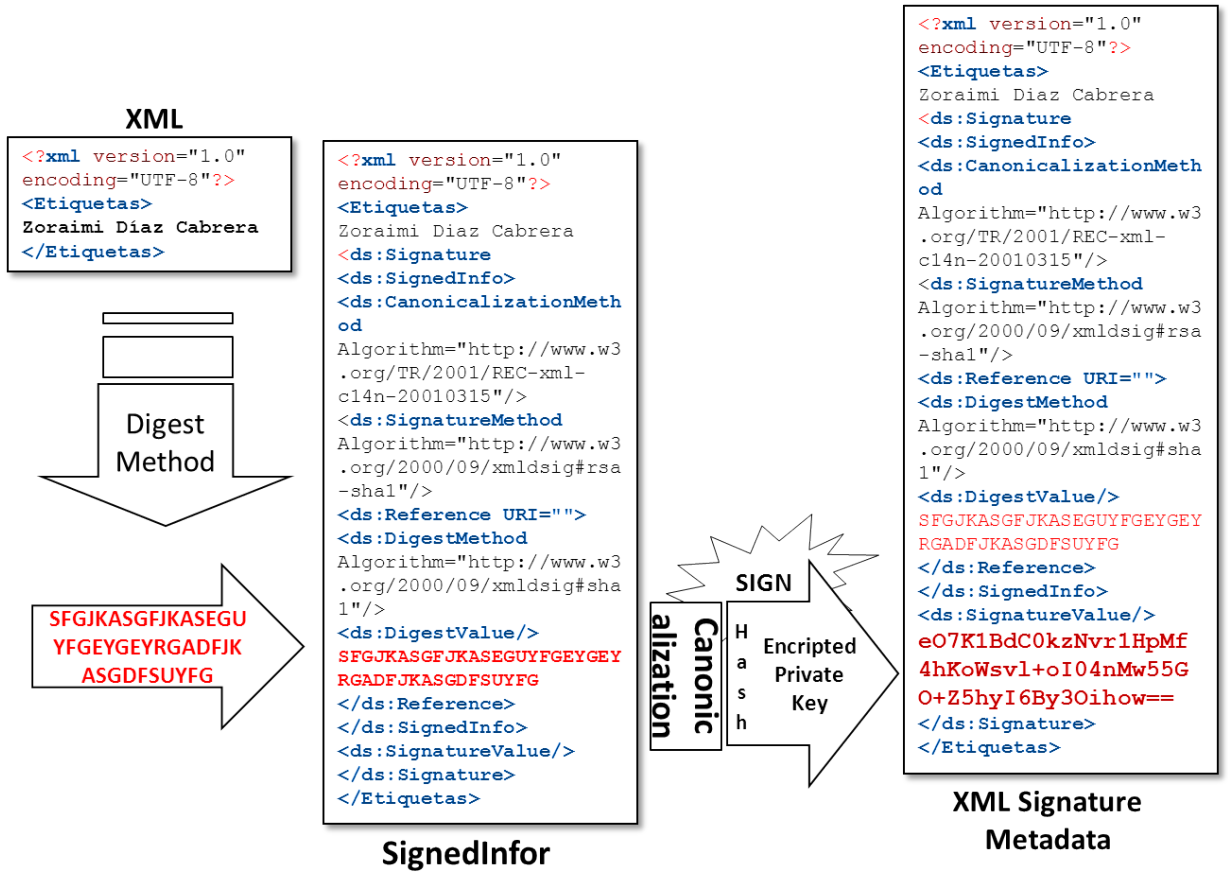


Figura 5: Generación de la Firma Digital XML.

Fuente: (Elaboración propia)

Validación de la firma

1. Obtener la información de la llave desde <KeyInfo>, o desde una fuente externa.

2. Obtener la forma canónica del elemento <SignatureMethod>, usando <CanonicalizationMethod>, y utilizar el resultado para confirmar el valor del elemento <SignatureValue> sobre el elemento <SignedInfo>.

Esquemas de firmas XML

Según la recomendación de la W3C existen tres maneras de firmar un documento XML. La Figura 6 muestra estas maneras.

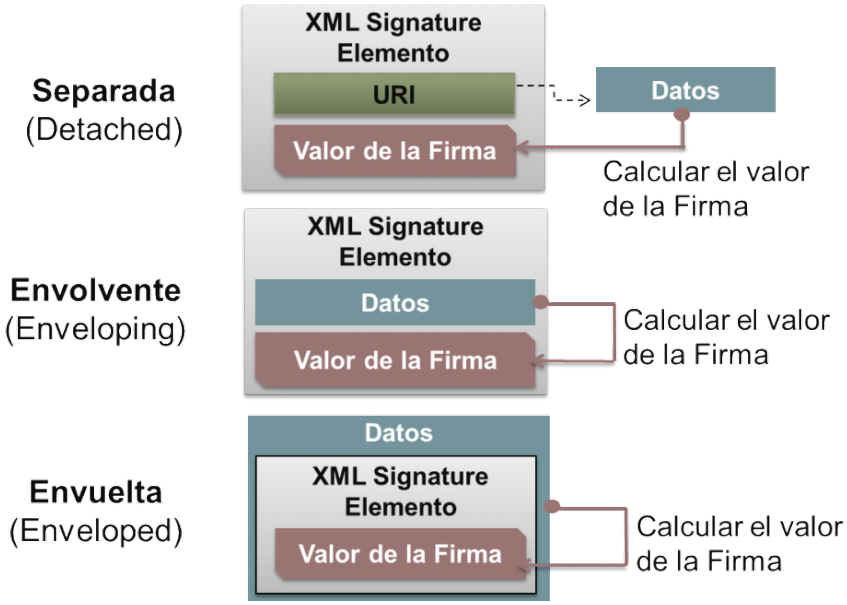


Figura 6: Esquemas de firmas XML

(**Fuente:** Elaboración propia)

Separada (Detached): la firma XML en modo *Detached* permite tener una firma de forma separada e independiente del contenido firmado, pudiendo relacionar la firma con el contenido firmado mediante una referencia de tipo URI. Este tipo de firmas es útil cuando no se puede modificar el contenido original pero se desea constatar su autenticidad y procedencia (ver Figura 7).

Envolvente (Enveloping): en este tipo de firma el documento se incluye dentro de la firma en la que se referencia lo firmado, como objeto insertado en la firma. Ya que se referencian los objetos <Object>, este modelo permitirá distinguir lo que se firma, pudiendo firmar el objeto entero o partes de él (asignando un id diferenciador) (ver Figura 8).

Envuelta (Enveloped): este formato está pensado para que un contenido XML pueda auto-contener su propia firma digital, insertándola en un nodo propio interno, al contrario que en los formatos anteriores, no es posible firmar contenido que no sea XML. Un ejemplo simple del resultado de una firma *Enveloped* se puede observar en la Figura 9.

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
<SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
<Reference URI="http://www.w3.org/TR/xml-stylesheet">
<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<DigestValue>60NvZvtdTB+7UnlLp/H24p7h4bs=</DigestValue>
</Reference>
</SignedInfo>
<SignatureValue> </SignatureValue>
</KeyInfo>
</KeyInfo>
</Signature>
```

Figura 7: Ejemplo de firma separada

Fuente: (Elaboración propia)

```
<?xml version="1.0" encoding="UTF-8"?>
<Signature xmlns:ds="http://www.w3.org/2000/09/XMLDsig#">
<SignedInfo>
<CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
<SignatureMethod Algorithm="http://www.w3.org/2000/09/XMLDsig#rsa-sha1"/>
<Reference URI="#obj">
<DigestMethod Algorithm="http://www.w3.org/2000/09/XMLDsig#sha1"/>
<DigestValue/>
</Reference>
</SignedInfo>
<SignatureValue/>
<ObjectId="obj">SFGJKASGFJKASEGUYFGEYGEYRGADFJKASGDFSUYFG=</Object>
</Signature>
```

Figura 8: Ejemplo de firma envolvente

Fuente: (Elaboración propia)

```
<?xml version="1.0" encoding="UTF-8"?>
<Letter>
<Message>msg body</Message>
<Signature xmlns: ds="&ds;">
<SignedInfo>
<CanonicalizationMethod Algorithm= "http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
<SignatureMethod Algorithm= "http://www.w3.org/2000/09/XMLDsig#rsa-sha1"/>
<Reference URI="">
<Transforms>
<Transform Algorithm="&enveloped;"> </Transform>
</Transforms>
<DigestMethod Algorithm="& digest;"/>
<DigestValue> </DigestValue>
</Reference>
</SignedInfo>
<SignatureValue/>
</Signature>
</Letter>
```

Figura 9: Ejemplo de firma envuelta

Fuente: (Elaboración propia)

Asociado al estándar XMLDsig se encuentra el estándar de firmas electrónicas avanzadas XML (XAdES). Mientras que XMLDsig es un entorno general para firmar digitalmente documentos XML, XAdES específica perfiles precisos de XMLDsig para ser usados como firma electrónica reconocida con el sentido de la directiva 1999/93/EC de la Unión Europea (W3C, 2003). Según el nivel de protección de los datos XML, XAdES incluye algunos elementos más a la firma XMLDsig como son: el sellado de tiempo, información sobre las políticas de firmas, datos sobre la validación del certificados entre otros perfiles. En la actualidad existen varias implementaciones del estándar XMLDsig utilizando el lenguaje de programación Java. Sin embargo, Java no tiene soporte para XAdES y las implementaciones públicas para este estándar son difíciles de encontrar (Goncalves, 2011).

1.3.2 Implementaciones de la firma XMLDsig utilizando Java

Existen diversas implementaciones de firma digital XML desarrolladas en lenguaje Java lo cual ha motivado la búsqueda de un conjunto de APIs estándar. En este sentido, *Java Community Process* (JCP) publica la *Java Specification Request* (JSR) 105 que proporciona los servicios para implementar la firma digital XML haciendo referencia a la recomendación de la W3C sobre la sintaxis y procesamiento de firmas XML (Oracle Corporation, 2011). La JSR 105 ofrece un API llamado *Java XML Digital Signature API Specification* compuesto por 6 paquetes:

- Javax.xml.crypto.dsig
- Javax.xml.crypto.dsig.keyinfo
- Javax.xml.crypto
- Javax.xml.crypto.dsig.spec
- Javax.xml.crypto.dom
- Javax.xml.crypto.dsig.dom

Las implementaciones realizadas utilizando esta API son un poco complejas pues la construcción de la estructura de la firma se tiene que realizar a través de la creación de cada uno de los elementos que ella posee. Sin embargo otra implementación que está dirigida a proporcionar las principales normas de seguridad para XML es el proyecto de Apache Santuario. Actualmente cuenta con la biblioteca apache.xml.security.signature basada también en el estándar JSR 105 que proporciona una implementación más fácil y cómoda a la hora de trabajar con las firmas XML, siendo la idónea para desarrollar el plugin de firma digital XML (Apache Santuario, 2012). Algunas ventajas de utilizar esta librería son:

- Reducción de memoria, menos creación de objetos.
- Incluye canonicalización para XML 1.0 y XML 1.1
- Reutilización del elemento <Signature> en la misma clase siempre que la clave sea la misma.
- Apache es una tecnología gratuita de código fuente abierto por lo que se puede encontrar abundante documentación en línea.

1.4 Estructura de un Archivo Java

Un archivo Java por sus siglas en inglés *Java Archive* (JAR) es un formato desarrollado por la empresa Sun Microsystems que permite agrupar las clases diseñadas en el lenguaje Java. Se utiliza tanto en aplicaciones web como de escritorio para almacenar ficheros de clases y recursos auxiliares asociados con applets y aplicaciones, otorgándole un nivel de compresión y portabilidad al distribuir clases de forma eficiente y sencilla en este lenguaje. Su estructura se basa en el formato estándar del archivo ZIP con un archivo de manifiesto llamado MANIFEST.MF contenido en el directorio META-INF (Sun Microsystems, 1997). La Figura 10 muestra las estructura de un archivo JAR. El fichero **MANIFEST.MF** porta información sobre el contenido del JAR, dependiendo de la información que contenga, se le pueden proporcionar funcionalidades avanzadas a estos archivos. Algunas de las funcionalidades que brindan son:

algoritmos de cifrado, firma digital, nombres de las clases, creador del manifiesto, clases importadas y control de versiones.

El **fichero de firma** con extensión .**SF** es un fichero que se crea automáticamente al firmarse el archivo JAR. Se crea un fichero .SF para cada entidad que firme el archivo JAR (Netscape Corporation, 2012). Para más información sobre este fichero ver epígrafe 1.4.1.

La extensión del **fichero de bloque de firma** se crea en dependencia al algoritmo de firma utilizado. Normalmente tienen la extensión **.RSA** o **.DSA**. Para cada fichero de firma se crea un fichero .RSA o .DSA. (Oracle Corporation, 2010). Para más información sobre este fichero ver epígrafe 1.4.1.

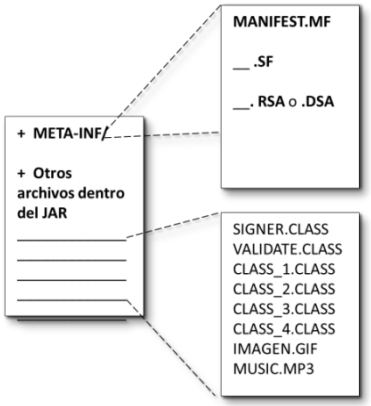


Figura 10: Estructura de un archivo JAR

Fuente: (Elaboración propia)

1.4.1 Firma digital en archivos JAR

Para asociar una firma digital con un archivo JAR se necesitan varios ficheros en el directorio META-INF. La Figura 11 muestra los diferentes ficheros generados a partir de una firma digital.

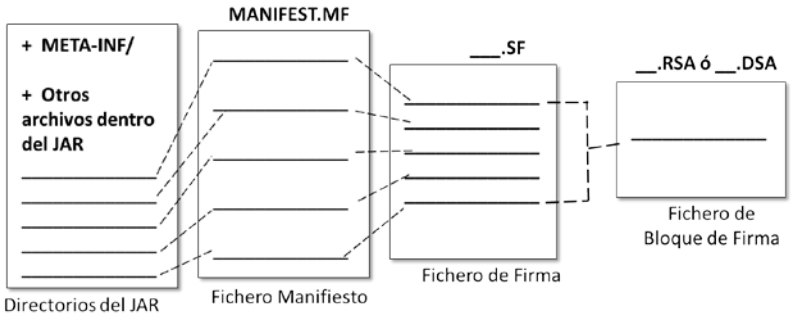


Figura 11: Generación de la firma digital en archivos JAR

Fuente: (Elaboración propia)

El fichero MANIFEST.MF contiene el resumen (hash) de cada uno de los ficheros contenidos en el JAR. Ejemplo del archivo Manifest en la siguiente Figura 12.

```
Manifest-Version: 1.0
Created-By: SignatureFile JDK 1.6

Name: test/classes/Manifest.class
SHA1-Digest: TD1GZt8G11dXY2p4olSZPc5Rj64 =

Name: test/classes/SignatureFile.class
SHA1-Digest: oBmrvIkBnSxdNZzPh5iLyF0S + = se

Name: test/images/manifest-concept.gif
SHA1-Digest: kdHbE7kL9ZHLgK7akHttYV4XIa0=
```

Figura 12: Ejemplo del fichero Manifest .MF

Fuente: (Elaboración propia)

Primero se registra la versión del manifiesto <Manifest-Version>. Luego en el elemento <Name> se específica la ruta de los ficheros o clases contenidas dentro del JAR e inmediatamente después se muestra el elemento <SHA1-Digest> con el resultado de aplicar el algoritmo hash utilizado para realizar la firma digital.

El fichero de firma **.SF** contiene cada una de las entradas del resumen del manifiesto (Figura 13). La línea SHA1-Digest-Manifest contiene el resumen del fichero de manifiesto. Los valores de resumen son representaciones codificadas en Base64 del contenido de los ficheros en el momento en que fueron firmados. El resumen de un fichero cambiará solo si cambia el propio fichero. Cuando se verifica un archivo JAR firmado, se calcula el resumen de cada fichero y se compara con el resumen almacenado en el fichero de manifiesto para asegurarse de que el contenido del fichero JAR no ha cambiado desde que se firmó (Sun Microsystems, 1997).

El fichero de bloque de firma .RSA o .DSA es un formato binario codificado en Base64 que no puede ser leído por los usuarios (Torrijos, 2011).Contiene dos elementos esenciales para la verificación del JAR:

1. La firma digital para el archivo JAR que fue generada por la clave privada del firmante.

2. La clave pública del firmante, junto con su certificado, para ser utilizadas por cualquiera que quiera verificar el fichero JAR.

```
Signature-Version: 1.0
SHA1-Digest-Manifest:
h1yS+K9T7DyHtZrtI+LxvgqaMYM=
Created-By: SignatureFile JDK 1.6

Name: test/classes/Manifest.class
SHA1-Digest: fcav7ShIG6i86xPepmitOVo4vWY=

Name: test/classes/SignatureFile.class
SHA1-Digest: xrQem9snnPhLySDiZyclMlsFdtM=

Name: test/images/manifest-concept.gif
SHA1-Digest: xrQv7ShIG6i86xPepmivclMlsF=
```

Figura 13: Ejemplo del fichero de firma .SF

Fuente: (Elaboración propia)

1.4.2 Implementaciones de la firma digital de Archivo JAR utilizando Java.

La JDK6 contiene un grupo de herramientas entre las que se encuentra jarsigner, usada para firmar ficheros JAR y verificar la autenticidad de las firmas de estos ficheros, además entre las librerías que proporciona Java se encuentra un API para manejar JAR (java.util.jar) y otras librerías que proporcionan seguridad a documentos y archivos electrónicos como por ejemplo:

- sun.security.pkcs.PKCS7[15]
- java.security.cert.

El proyecto APACHE ANT también es una herramienta en el lenguaje de programación Java que contiene las librerías y la documentación necesaria para trabajar con archivos JAR. Su funcionamiento se basa en un archivo de configuración XML y clases Java para la realización de las distintas tareas. Entre sus tareas definidas se encuentra <signjar> utilizada para firmar archivos JAR.

[15]PKCS#7 es una estándar sobre la sintaxis de los mensajes criptográficos. Permite firmar, crear resúmenes (hash), autentificar o cifrar cualquier tipo de datos.

1.5 Herramientas y lenguajes utilizados

Para el desarrollo de las extensiones o plugins de firma digital, se decide utilizar las mismas herramientas y lenguajes que dieron solución a la creación del sistema ADVsigner, buscando compatibilidad e integración con la misma.

1.5.1 Java Development Kit 6

Es un paquete de programación de software para producir programas en Java, incluye herramientas tales como los compiladores y depuradores necesarios para el desarrollo de applets y aplicaciones. Este kit proporciona la máquina virtual Java (JVM), y otros componentes para ejecutar applets y aplicaciones escritas en el lenguaje de programación Java. (Oracle, 2011)

1.5.2 NetBeans 6.9

Es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java, con una gran base de usuarios. La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de Java escritas para interactuar con las APIs de NetBeans. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software (Oracle Corporation, 2011).

1.5.3 Visual Paradigm 6.4

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación (Paradigm, 2011).

Se integra con varios ambientes de desarrollo integrados (IDE) lo cual posibilita pasar del código al modelado y viceversa. Establece interoperabilidad con otras aplicaciones como el Visio30 y el Rational Rose. Disponible en múltiples lenguajes y plataformas: Microsoft Windows (98, 2000, XP, o Vista) Linux, Mac OS X, Solaris o Java.

1.5.4 Lenguaje de Modelado Unificado

Es el lenguaje gráfico de modelado (UML) orientado a objetos estándar de la industria para especificar, visualizar, construir y documentar los elementos de los sistemas de software. UML proporciona una forma estándar de escribir los planos de un sistema, cubriendo tanto aspectos conceptuales, tales como procesos del negocio y funciones del sistema, como las clases escritas en un lenguaje de programación específico, esquemas de bases de datos y componentes de software reutilizables. Simplifica el complejo proceso de análisis y diseño de software, facilitando un plano para la construcción de los artefactos del software (OMG, 2012).

1.5.5 Java

Es un lenguaje de programación orientado a objetos, de alto nivel, desarrollado por Sun Microsystems a principios de los años 1990. Desde su creación el entorno Java ha tenido presentes los problemas de seguridad y ha definido un modelo para controlar y limitar el acceso a los recursos desde los programas y aplicaciones. Se utiliza este leguaje por haberse utilizado además en la implementación del sistema ADVsigner.

Java fue diseñado para ofrecer las siguientes medidas de seguridad básicas:

- Uso de un lenguaje de programación seguro.
- Integración de un sistema de control de permisos para los programas. Java define un mecanismo que controla qué se le permite hacer a un programa y cómo accede a los recursos.
- Encriptación y uso de certificados: Se definen mecanismos para que los programadores puedan firmar el código, de manera que los usuarios puedan verificar quién es el propietario del código y que este no ha sido modificado después de ser firmado.

1.6 Metodologías de desarrollo del software

El proceso de desarrollo de software debe estar guiado por una metodología que sirva como plano y oriente al equipo de desarrollo. Entre las metodologías de desarrollo más utilizadas actualmente se encuentran: Extreme Programming (XP), Rational Unified Process (RUP) y SCRUM (José H. Canós, 2009).

- **Rational Unified Process**

RUP es una metodología robusta que junto al lenguaje de modelado UML, constituye la metodología estándar más utilizada para el diseño, implementación y documentación de los sistemas orientados a objetos (P.Letelier). Se caracteriza por:

- La forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
- Pretende implementar las mejores prácticas en Ingeniería de Software.
- Desarrollo iterativo.
- Administración de requisitos.
- Control de cambios.
- Modelado visual del software.
- Verificación de la calidad del software.

Se caracteriza de forma general por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Su ciclo de vida define cuatro fases.

- Conceptualización (Concepción o Inicio): Es la fase de la idea, de la visión inicial del producto, su alcance, el esbozo de una posible arquitectura y las primeras estimaciones.
- Elaboración: Comprende la planificación de las actividades y del equipo, además de la especificación de las necesidades del producto y el diseño de la arquitectura.
- Construcción: Desarrollo del producto hasta que se encuentra disponible para su entrega a los usuarios.
- Transición: Traspaso del producto a los usuarios. Incluye: manufactura, envío, formación, asistencia y el mantenimiento hasta lograr la satisfacción de los usuarios.

La principal desventaja de RUP son los altos costos de cambios dados por la planificación a largo plazo y la trazabilidad que existe entre los diferentes artefactos, generando gran cantidad de documentación a lo largo de todas las fases.

- **Extreme Programming (XP)**

XP es una metodología ligera (ágil) de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado. Esta metodología trata de dar al cliente el software que él necesita y cuando lo necesita. Por tanto, se debe responder muy rápido a las necesidades del cliente. El segundo objetivo es potenciar al máximo el trabajo en grupo. Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del software (Beck, y otros, 2004).

El ciclo de vida ideal de XP consta de seis fases: Exploración, Planificación de la Entrega, Iteraciones, Producción, Mantenimiento y Muerte del Proyecto. En la Figura 14 se muestra las fases en las que se subdivide el ciclo de vida de un proyecto de software con XP.

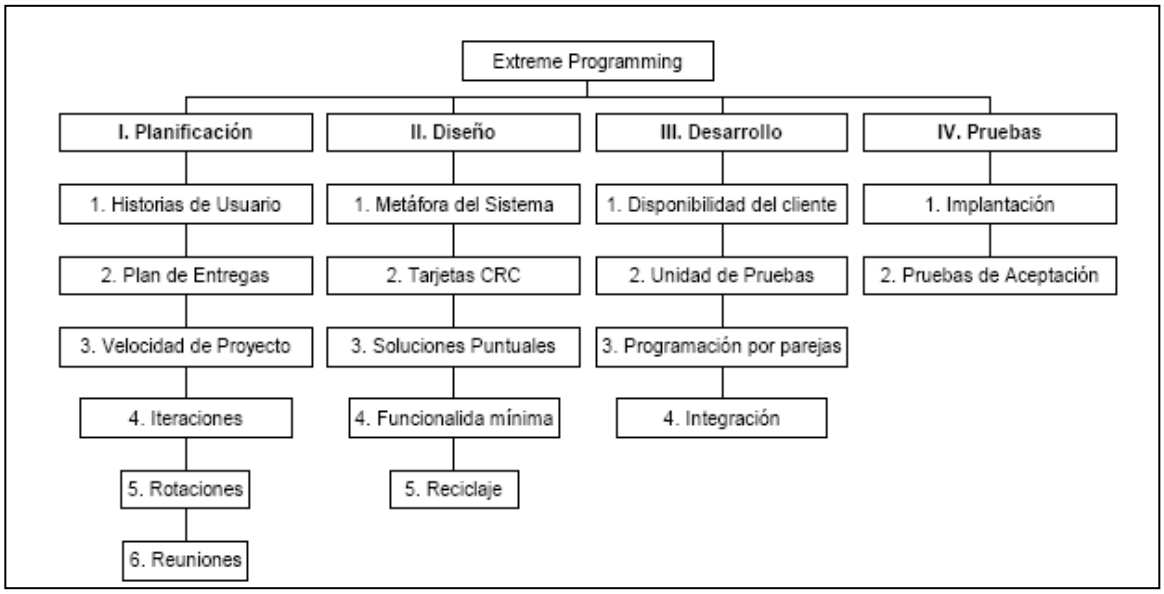


Figura 14: Ciclo de vida de trabajo con XP

Fuente: (Agile Alliance, 2010)

Planificación: en la cual la tarea más importante es la creación de las Historias de Usuario (HU) que son escritas por el cliente para expresar las funcionalidades que desea y son usadas para estimar el tiempo de desarrollo de la funcionalidad que describen.

Diseño: en esta fase se crean las Tarjetas CRC las cuales definen una clase expresando las funcionalidades de esta y las otras clases con las que colabora.

Desarrollo: es donde se definen las tareas de desarrollo para que los desarrolladores tengan una guía para implementar todas las HU.

Pruebas: en esta fase se le realizan diferentes test a cada una de las HU para probar que cumplen con las funcionalidades que desea el cliente.

- **Scrum**

Esta es, después de XP, la metodología ágil mejor conocida y la que otros métodos ágiles recomiendan como complemento (Reynoso, 2004). Es un proceso ágil que nos ayuda a poner nuestro mayor esfuerzo en entregar el valor más importante de un producto en el tiempo más corto posible. Nos permite enviar, inspeccionar y evaluar periódicamente versiones funcionales del producto (cada 2-6 semanas aproximadamente) y cada 30 días se puede ver el software real, que funciona, decidiendo si se entrega o se sigue refinando en el siguiente ciclo. No es una metodología de análisis, ni de diseño, es una metodología de gestión del trabajo. Ofrece las siguientes ventajas:

- Fácil de explicar y entender.
- Máximo valor por el esfuerzo.
- Equipos auto-organizados, multi-aprendizaje.
- Liberación de versiones totalmente funcionales, aunque con funcionalidad limitada, en cada iteración.
- Orientación al cliente: lo involucra y le permite establecer prioridades.
- Satisface al cliente con entregas tempranas y continuas de productos funcionales.
- Provee a la gerencia o dirección con una vista diaria sobre del nivel de productividad y los factores que la afectan.
- Mayor productividad en el desarrollo de software.
- Permite producir software de forma regular, consistente y competitiva.

- Útil en entornos con requisitos inestables, se aceptan requisitos cambiantes, incluso en etapas avanzadas.

1.7 Conclusiones parciales

En este capítulo se abordaron temas relacionados con la firma digital permitiendo conocer sobre los principales algoritmos criptográficos de llave pública y las principales características de las herramientas utilizadas para este fin.

Con el análisis de la estructura y de las principales funcionalidades de la solución de firma digital ADVsigner queda evidenciada la necesidad de garantizar autenticidad, integridad y no repudio de la información transmitidas en otros tipos de documentos utilizando extensiones o plugins de firma digital a través de este sistema.

El estudio realizado sobre la firma digital XML a través del estándar W3C y la firma digital de los archivos JAR utilizando la especificación de JAVA proporcionan la base teórica para implementar y entender la estructura de la firma digital en cada uno de estos documentos y archivos electrónicos.

Es de vital importancia la selección de las herramientas y lenguajes utilizados, debido a que de ello depende el proceso de desarrollo de los plugins de firma digital facilitando además el modelado de la solución.

El estudio de diferentes metodologías de software contribuyó a la selección de una metodología que se adapte a las condiciones del proyecto proporcionando resolver el problema planteado de la manera más ágil posible.

Capítulo II: Solución propuesta

Este capítulo muestra la evolución de la solución durante las fases iniciales de la metodología de desarrollo seleccionada, así como los diferentes artefactos generados en cada una de sus fases; además, se analizan los características fundamentales del sistema ADVsigner, donde surge la necesidad de las extensiones y la propuesta de solución; y por último, se representan los principales conceptos del modelo de dominio de la solución y los tipos de objetos más importantes en el contexto del sistema, a través de la identificación de las entidades principales y las relaciones entre ellas.

2.1 Descripción del problema

La solución de firma digital ADVsigner solo posee una extensión o plugin de firma utilizada para firmar documentos electrónicos en formatos PDF, impidiendo que se garantice autenticidad, integridad y no repudio de la información transmitida mediante otros documentos y archivos electrónicos como es el caso de los documentos XML y archivos JAR, formatos muy utilizados en nuestros días. Por lo que se hace necesario desarrollar extensiones de firma digital para proveer con nuevas funcionalidades de firma y validación digital la solución ADVsigner.

2.2 XP metodología seleccionada

Se decide utilizar XP debido a que se adapta en gran medida tanto al tipo de proyecto a desarrollar cómo a las condiciones de trabajo. A continuación se exponen varias de las razones que llevaron a la selección de esta metodología.

- El proyecto es pequeño. XP está concebida para ser utilizada dentro de proyectos pequeños.

- El cliente forma parte del equipo de desarrollo, en este proyecto, como no se tiene un cliente en específico, los mismos desarrolladores son los clientes o usuarios del sistema.

- Imposibilidad, para un grupo de desarrollo pequeño, de asumir una metodología robusta, debido a la cantidad excesiva de roles y documentación generada en el ciclo de vida del proyecto.

- Debido al corto tiempo de entrega planteado y a los continuos cambios de requerimientos, XP se utiliza porque se adapta perfectamente a proyectos cuyos requerimientos cambian a menudo (Agile Alliance, 2010).

2.3 Propuesta de la solución a desarrollar

Se propone como solución para garantizar autenticidad, integridad y no repudio de la información trasmitida en documentos XML y archivos JAR a través de la solución ADVsigner, la implementación de una extensión para firmar documentos en formatos XML y otra para firmar archivos JAR. La firma XML se podrá hacer de tres formas según la especificación estándar de la firma digital recomendada por la W3C.

- Envuelta
- Separada
- Envolvente

2.3.1 Modelo de dominio

"Un modelo de dominio es una representación de las clases conceptuales del mundo real, no de componentes de software. No se trata de un conjunto de diagramas que describen clases de software ni objetos de software con responsabilidades, sino más bien representa las clases conceptuales u objetos del mundo real en un dominio de interés. El modelo de dominio se debe concebir como un diccionario visual de abstracciones que será utilizado en fases posteriores y cuya función principal es ayudar a comprender el problema a tratar" (Bddigital, 2009).

Por la relativa simplicidad del entorno donde se desarrolla la extensión, basta con el modelo de dominio para capturar los principales conceptos alrededor del problema que la solución resuelve, representado mediante un diagrama UML. Se decide realizar el modelo del dominio debido a que las fronteras y los procesos del negocio no están claramente definidos. A continuación se muestra en la Figura 15 el modelo del dominio propuesto.

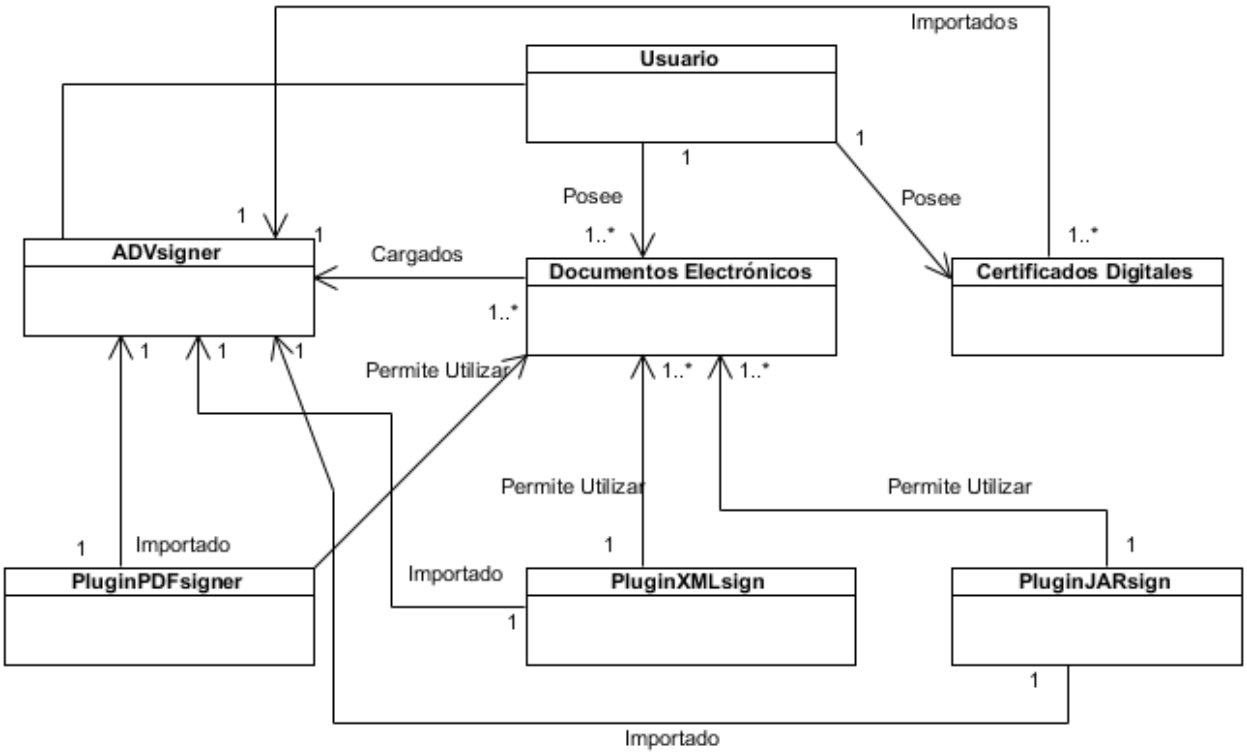


Figura 15: Modelo de Dominio

Fuente: (Elaboración Propia)

2.3.2 Entidades y conceptos del modelo de dominio

Para desarrollar un trabajo correcto con el modelo de dominio se hace 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 de las extensiones o plugins de firma.

- ADVsigner: es una solución de firma digital que el usuario utiliza para realizar todo el proceso de firma y validación de documentos electrónicos. Sistema que haciendo uso de certificados digitales y plugins instalados en el mismo podrá firmar documentos electrónicos de diferentes formatos.

- PluginPDFsign: extensión que se añade al sistema ADVsigner, permitiendo firmar documentos en formato PDF.

- PluginXMLsign: extensión que se le añade al sistema ADVsigner, permitiendo firmar documentos en formato XML.

- PluginJARsign: extensión que se le añade al sistema ADVsigner, permitiendo firmar documentos en formato JAR.

- Certificados Digitales: es un documento otorgado por una autoridad de certificación que garantiza la asociación de una persona física con una firma digital, teniendo asociado la llave privada con su correspondiente llave pública.

- Documentos Electrónicos: son los archivos electrónicos que el usuario firmará digitalmente, para poder realizar el proceso de firma y validación de documentos es necesario que los plugins sean importados con anterioridad a la aplicación.

2.3.3 Historias de Usuarios

El primer paso de cualquier proyecto que siga la metodología XP es definir las historias de usuario (HU) con el cliente, estas no son más que los requisitos funcionales que debe de tener la aplicación, asumiendo la misma funcionalidad que los casos de uso; constan de tres o cuatro líneas escritas por el cliente en un lenguaje no técnico, sin hacer mucho hincapié en los detalles; no se debe hablar ni de posibles algoritmos para su implementación ni de diseños de base de datos adecuados. Son usadas para estimar tiempos de desarrollo de la parte de la aplicación que describen. También se utilizan en la fase de pruebas, para verificar si el programa cumple con lo que especifica la historia de usuario. Para implementar una historia de usuario, el cliente y los desarrolladores se reúnen con el objetivo de concretar y detallar lo que tiene que hacer dicha historia. El tiempo de desarrollo ideal para una historia de usuario es entre una y tres semanas (Beck, y otros, 2004).

Tabla 1: HU1_ Firmar Documentos XML

Historia de Usuario	
Número: HU1	**Usuario:** Desarrollador
Nombre historia: Firmar Documentos XML	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 3	**Iteración asignada:** 1
Programador responsable: Zoraimi Díaz	
Descripción: - El usuario buscará desde un archivo el documento XML a firmar, esto estará en correspondencia con el plugin instalado en el sistema. - Posteriormente se selecciona el certificado digital que puede encontrarse en el almacén de llaves de la aplicación o en una tarjeta inteligente.	

– A continuación se selecciona el o los documentos a firmar. – Finalmente se aplica la firma digital sobre los documentos XML seleccionados, creando un nuevo documento con la firma incluida.
Observaciones:

Tabla 2: HU2_Validar Firma de Documento XML

Historia de Usuario	
Número: HU2	**Usuario:** Desarrollador
Nombre historia: Validar Firma de Documento XML	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Zoraimi Díaz	
Descripción: La opción permitirá validar la firma de los documentos firmados. Se importarán a la aplicación aquellos documentos previamente firmados y que se encuentren asociados a un plugin instalado en la aplicación. Concluido el proceso de validación se le informará al usuario el resultado del proceso.	
Observaciones:	

Tabla 3: HU3_Firmar Archivos JAR

Historia de Usuario	
Número: HU3	**Usuario:** Desarrollador
Nombre historia: Firmar Archivos JAR	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: Zoraimi Díaz	
Descripción: – El usuario buscará el archivo JAR a firmar, esto estará en correspondencia con el plugin instalado en el sistema. – Posteriormente se selecciona el certificado digital que se encontrará en el almacén de llaves de la aplicación o en una tarjeta inteligente. – A continuación se selecciona el o los archivos JAR a firmar. – Finalmente se aplica la firma digital sobre los archivos JAR seleccionados, creando un nuevo archivo con la firma incluida.	

Observaciones:

Tabla 4: HU4_Verificar Firma en los Archivos JAR

Historia de Usuario	
Número: HU3	**Usuario:** Desarrollador
Nombre historia: Verificar Firma en los Archivos JAR	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Zoraimi Díaz	
Descripción: Esta opción permitirá verificar la firma de los archivos JAR previamente firmados. La validación solo se hará a los archivos que se encuentren asociados a un plugin instalado en la aplicación. Concluido el proceso de validación se le informará al usuario el resultado del proceso.	
Observaciones:	

2.3.4 Requisitos no funcionales

Requisitos mínimos de hardware:

- Microprocesador Intel Pentium IV o semejante, con velocidad de procesamiento de 1.0 GHz o superior.
- Memoria RAM mínima de 256 Mb
- Al menos 60 Mb de espacio en disco duro.

Requisitos mínimos de software:

- Debe tenerse instalado el Java Runtime Environment (JRE) versión 6.
- Se debe disponer de sistemas operativos Linux, Windows XP/Vista/7 para la instalación del sistema ADVsigner.

Requisitos de diseño e implementación:

- Las extensiones deben ser implementadas utilizando el lenguaje Java y el NetBeans como entorno de desarrollo.

- Las extensiones creadas deben ser compatibles con el sistema ADVsigner.
- Las extensiones deben permitir la reutilización del código, así como facilidad en la actualización del mismo.
- La herramienta case a utilizar debe de ser el Visual Paradigm 6.4.

2.3.5 Metáfora

Después de haberse definido las funcionalidades que el sistema debe cumplir y los requerimientos no funcionales, el equipo procede a la creación de la Metáfora. Esta es una breve descripción de cómo debe funcionar el sistema en su totalidad. La Metáfora guía todo el desarrollo como una gran historia de usuario ayudando al equipo a entender los elementos básicos del sistema y sus relaciones.

Tabla 5: Metáfora del sistema

Metáfora del Sistema
Para utilizar las extensiones de firma digital es necesario que hayan sido importadas con anterioridad al sistema ADVsigner, luego se selecciona un certificado digital que también será importado con anterioridad o en el momento de firma, desde un archivo o utilizando tarjetas inteligentes, y posteriormente se procede a firmar los documentos o archivos electrónicos.

2.3.6 Estimación de Tiempo

De acuerdo a las historias de usuarios escritas, los programadores estiman el esfuerzo asociado a la implementación de las mismas utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias de usuarios generalmente valen de uno a tres puntos. Si la estimación indica que la implementación de esta se toma más de tres semanas entonces pasa nuevamente a manos del cliente, dividiéndose en dos o más historias de usuarios debido a su complejidad, posibilitando esto mayor flexibilidad a la hora de la implementación. Teniendo en cuenta lo anteriormente explicado se planifica la siguiente entrega priorizando una lista de historias de usuario ordenadas por el cliente de acuerdo al valor del negocio que aporten.

Tabla 6: Estimación de Tiempo

Historia de Usuario	Estimación(semanas)
Firmar Documentos XML	3
Validar Firma de Documentos XML	2
Firmar Archivos JAR	3
Verificar Firma en los Archivos JAR	2
Total	**10**

2.3.7 Plan de Iteraciones

Como parte del ciclo de vida de un proyecto usando la metodología XP se crea el plan de duración de cada una de las iteraciones definidas, que tiene como objetivo mostrar la duración y el orden en que serán implementadas las historias de usuarios dentro de cada iteración. Generalmente las historias de usuarios con mayor prioridad son asignadas a las primeras iteraciones, y las de mediana y menor prioridad a las iteraciones posteriores. En la presente solución se han identificado cuatro historias de usuarios y se definieron dos iteraciones; para una duración total de cinco semanas en ambas iteraciones. En la Tabla 7 se muestra el plan de iteraciones.

Tabla 7: Plan de Iteraciones

Iteración	No. HU	Historia de Usuario	Duración Estimada
Iteración 1	HU1	Firmar Documentos XML	5
	HU2	Validar Firma de Documentos XML	
Iteración 2	HU3	Firmar Archivos JAR	5
	HU4	Verificar Firma en los Archivos JAR	

2.3.8 Plan de Entrega

Para finalizar la fase de planeamiento se acuerda con el cliente el plan de entregas que define las fechas en que serán liberadas las versiones funcionales del producto. Este artefacto cumple con el principio de la metodología XP de las “liberaciones frecuentes”, generalmente asociadas al fin de un grupo de iteraciones (Agile Alliance, 2010). Al finalizar cada iteración se hace entrega de la versión funcional de los plugins. La Tabla 8 muestra la planificación de la entrega.

Tabla 8: Plan de Entrega

Iteración	Historias de Usuarios	Entregable
Iteración 1	HU1: Firmar Documentos XML	Plugin XMLsign Versión 1.0 Mayo 2011
	HU2: Validar Firma de Documentos XML	
Iteración 2	HU3: Firmar Archivos JAR	Plugin JARsign Versión 1.0 Junio 2011
	HU4: Verificar Firma en los Archivos JAR	

2.4 Diseño de la solución

Una vez desglosadas las historias de usuario por iteraciones, algunos miembros del equipo de desarrollo comienzan a generar ideas de cómo ejecutarlas. Estas ideas se unifican en las sesiones de diseño previas a cada iteración. Consiguiendo diseños simples y sencillos, procurando hacerlo todo lo menos complicado posible (Beck, y otros, 2004). Para representar el diseño del sistema se realiza el diagrama de clases (ver Figura 16) en notación UML. Además de utilizar las tarjetas CRC (Cargo o clase, Responsabilidad y Colaboración) para representar las clases principales de las extensiones o plugins a desarrollar. Las tarjetas CRC en sí representan un objeto donde las clases se examinan, se filtran y se refinan en base a sus responsabilidades respecto a las funcionalidades que el sistema requiere, y las clases con las que necesitan colaborar para completar sus responsabilidades.

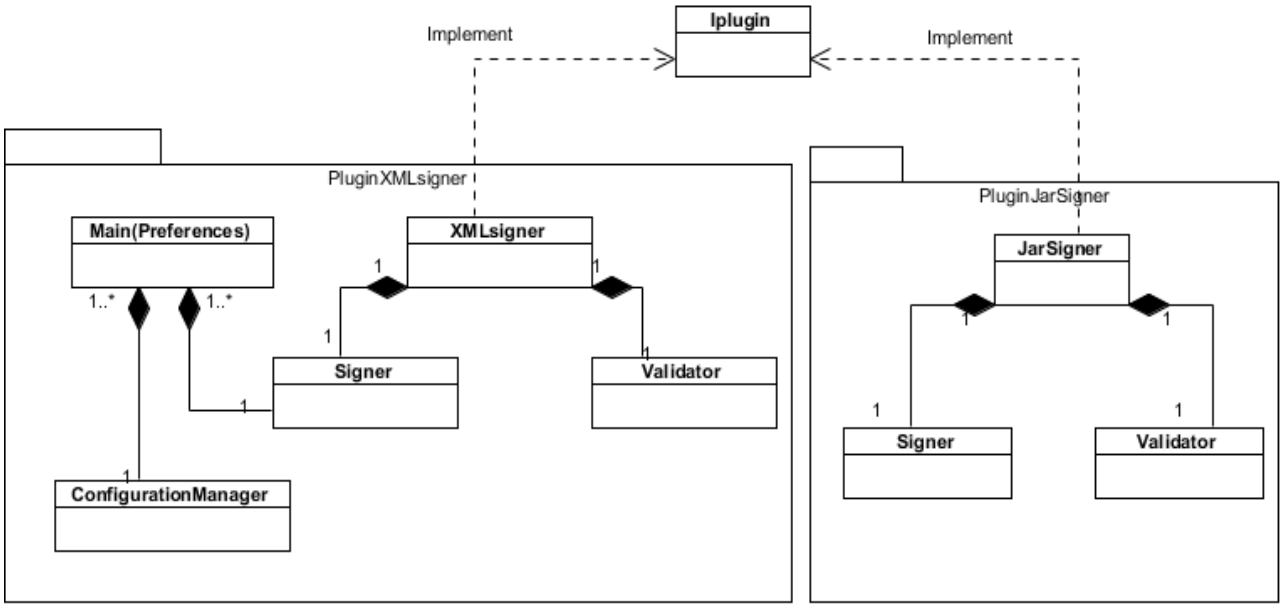


Figura 16: Diagrama de clases para los plugins

Fuente: (Elaboración propia)

Según (Silot Ochoa, 2011) el diseño de los plugin se encuentra estructurado en capas, permitiendo cierto grado de independencia entre las partes de la aplicación, característica que la convierten en una solución escalable y de fácil mantenimiento. A su vez los paquetes se dividen en sub-paquetes que agrupan las clases atendiendo a sus funcionalidades (ver epígrafe: 1.2 Solución de firma digital ADVsigner). Al aplicar el sub-estilo del programa principal, las funciones se descomponen en una jerarquía de control donde el programa principal invoca los programas subordinados, los cuales a su vez pueden invocar a otros. Por lo que utilizar este diseño para desarrollar los nuevos plugins de firma digital posibilita al diseñador de software conseguir una estructura fácil de modificar, escalar, mejorando así la modularidad y extensibilidad del sistema.

2.4.1 Tarjetas CRC

Tabla 9: Tarjeta CRC XMLsigner

XMLsigner	
Funcionalidades	Colaboración
Obtener Formato de Firma Obtener Resumen Mostrar Preferencias	Signer Validator

Tabla 10: Tarjeta CRC Signer

Signer	
Funcionalidades	Colaboración
Obtener Fichero a Firmar Obtener Hash Modificar Hash Estructurar el Formato de Firma Salvar Fichero Modificado	XMLsigner Main ConfigurationManager

Tabla 11: Tarjeta CRC Validator

Validator	
Funcionalidades	Colaboración
Validar Firma	XML_Signer

Tabla 12: Tarjeta CRC JARsigner

JARsigner	
Funcionalidades	Colaboración
Estructurar el Formato de Firma	Signer

Obtener Hash Obtener Fichero a Firmar Salvar Fichero Modificado	Validator

El uso de patrones en el diseño permite reutilizar código, ahorrar grandes cantidades de tiempo y tener un sistema con una estructura conocida por todos los programadores. Un patrón de diseño es una solución estándar para un problema común de programación, una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios. Se entiende también por patrón de diseño a un proyecto o estructura de implementación que logra una finalidad determinada, una manera más práctica de describir ciertos aspectos de la organización de un programa (Shalloway, 2004). A continuación se hace referencia a los principales patrones a utilizar y su relación con las clases implementadas. El patrón alta cohesión se ve reflejado en las clases Validator y Signer ya que estas clases realizan una labor única dentro del sistema no desempeñado por el resto de los elementos. Otro patrón identificado es el alto acoplamiento que existe en la clase XMLsigner y JARsigner debido a que esta clase depende de un conjunto de métodos que ya están definidos para poder ser implementada. El patrón Instancia Única es utilizado en la creación de la clase ConfiguratioManager, garantizando que exista una sola instancia de la clase en la solución.

2.5 Conclusiones parciales

El presente capítulo sirvió de base para comenzar con el desarrollo de la solución propuesta, generando los principales artefactos que propone XP en sus fases iniciales por lo que se llega siguiente conclusión.

- La metodología de software seleccionada, da la medida de la cantidad de etapas y acciones que se deben tener en cuenta para la realización e implementación del software. Se potencia la utilización de la metodología ágil XP para optimizar el tiempo de desarrollo.

- Con la elección del modelo de dominio se describieron los principales conceptos asociados a la solución así como las relaciones entre estos conceptos permitiendo entender el problema a resolver.

- La representación de los requisitos funcionales a través de historias de usuarios permitió describir las funcionalidades del sistema y estimar el tiempo de desarrollo de

los plugin plasmados en el plan de estimación, además se definieron los requisitos no funcionales que brindaron las cualidades a tener en cuenta para desarrollar una solución adecuada.

- Con la representación del diagrama de clases y la descripción de las clases en tarjetas CRC fue posible realizar el diseño de la solución de la forma más sencilla posible según hace referencia la metodología XP.

Capítulo III: Implementación y prueba

El presente capítulo describe la implementación de la solución propuesta y se detallan las iteraciones llevadas a cabo durante la etapa de construcción del sistema, mostrando las tareas generadas por cada historia de usuario, así como las pruebas realizadas al sistema.

3.1 Implementación de la solución propuesta

La implementación de la solución propuesta a través de la metodología XP se realiza de forma iterativa e incremental, obteniendo en cada iteración un producto funcional, el cual debe ser probado y mostrado al cliente, permitiendo de esta forma una constante retroalimentación. Esta metodología promueve la programación basada en estándares, de manera que el código sea fácilmente entendible por todo el equipo de desarrollo, y que facilite la recodificación (Joskowicz, 2008). El estilo de codificación que se utiliza para implementar la solución está basado en el estándar recomendado por Sun Microsystems, difundido y aceptado ampliamente por toda la comunidad Java, y herramientas como el Net Beans IDE hacen uso de él facilitándole el trabajo a los desarrolladores (Flower, 2010).

Para comenzar con la implementación de las extensiones o plugins de firma digital es necesario saber cómo funciona el sistema ADVsigner en su totalidad. Esto comienza cuando el usuario decide firmar un documento (ver Figura 17), el cual es enviado al plugin quien extrae el hash (la huella digital del documento) a firmar y lo envía a la aplicación. Recibido el contenido en la aplicación es encriptado utilizando la llave privada del firmante y el resultado es la firma digital del documento. Obtenida la firma digital es enviada al plugin, donde este último la adjunta al documento original y crea un nuevo documento firmado digitalmente (Silot Ochoa, 2011).

Para la confección del Plugin es necesario tener la librería IPLugin.jar, donde se localizan las clases necesarias para la creación de la extensión. La interfaz IPlugin.java ubicada dentro de la librería, provee el punto de extensión que deben implementar todos los plugins. Es necesario además contar con el archivo plugin.xsd para la confección del archivo XML perteneciente al plugin (ver Figura 18). El fichero XML del plugin es obligatorio, en él se definen aspectos importantes para la carga del plugin en el sistema. Un plugin no será cargado en el sistema sino presenta su correspondiente fichero XML. Si el plugin necesita librerías extras para su

funcionamiento, las mismas deberán estar, ya sea al lado o en una carpeta dedicada a este fin o ubicada junto al JAR principal.

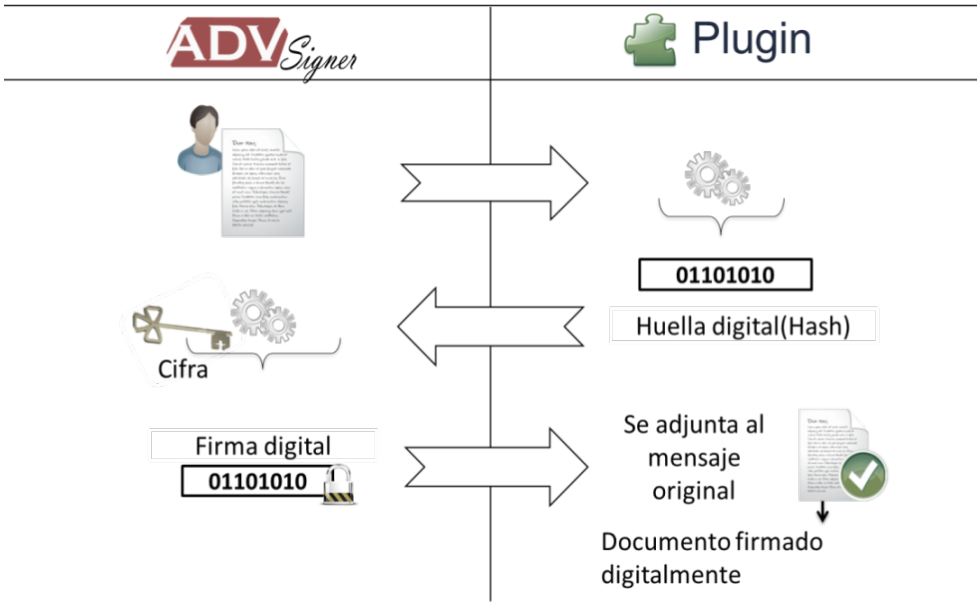


Figura 17: Proceso de firma digital utilizando el Sistema ADVsigner

Fuente: (Silot Ochoa, 2011)

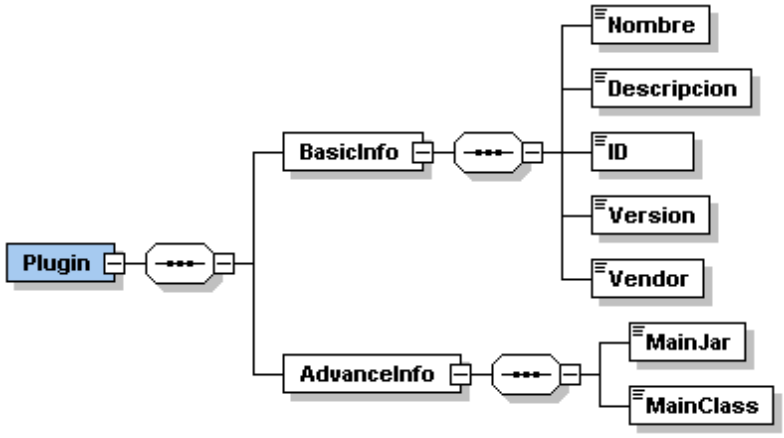


Figura 18: Estructura del archivo XML del Plugin

Fuente: (Elaboración Propia)

<BasicInfo>: este elemento posee información básica sobre el plugin a desarrollar, como por ejemplo: nombre del plugin, descripción, identificador, versión y autor del mismo.

<AdvanceInfo>: este elemento especifica datos necesarios para cargar el plugin en el sistema; Cuenta con dos secciones MainJar y MainClass.

- <MainJar>: contiene el nombre del JAR principal en caso contrario el plugin será descartado por el sistema.
- <MainClass>: especifica el nombre de la clase principal del plugin a desarrollar.

La recta final en la confección de un plugin comienza con el empaquetamiento. Se empaquetan en un archivo ZIP las librerías utilizadas, el archivo XML con la información del plugin y el JAR principal del proyecto. Obtenido el archivo final se cambia la extensión .ZIP por .SIL ya que es la extensión que utiliza la solución ADVsigner para cargar los plugins.

Teniendo como premisa la planificación realizada en el capítulo anterior se lleva a cabo la implementación de las dos iteraciones para el desarrollo de la solución propuesta. Durante el inicio de cada iteración, se realiza una revisión del plan de iteraciones y se modifica de ser necesario. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, permitiendo que al final se logren las extensiones o plugins de firma digital con las características específicas para cada documento o archivo electrónico, estudiados en capítulos anteriores.

3.1.1 Iteración 1

En esta primera iteración se desarrollan las historias de usuario de mayor prioridad en el dominio del problema con el objetivo de obtener la primera versión de la extensión. Ver la estimación de tiempo para cada historia de usuario en la Tabla 13.

Tabla 13: HU abordadas en la primera iteración

Iteración	No. HU	Historia de Usuario	Estimación
Iteración 1	HU1	Firmar Documentos XML	3
	HU2	Validar Firma de Documentos XML	2

3.1.2.1 Tareas de la ingeniería Iteración 1

Tabla 14: Tareas de la ingenieria a primera iteración

Iteración	No	Historia de Usuario	Tareas
Iteración 1	HU1	Firmar Documentos XML	– Cargar plugins para firmar documentos XML – Cargar documento XML a firmar. – Firmar documentos XML.

	HU2	Validar Firma de Documentos XML	– Cargar documentos a validar. – Validar documento XML firmado

Tareas de la ingeniería detalladas Iteración 1

HU Firmar Documentos XML

Tabla 15: HU1_T1

Tarea	
Número de la Tarea: HU1_T1	**HU:** Firmar Documentos XML
Nombre: Cargar plugins para firmar documentos XML	
Tipo: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio:	**Fecha Fin:**
Responsable: Zoraimi Diaz	
Descripción: El usuario cargará de manera visual desde archivo el plugin para firmar documentos XML. El plugin para ser cargado tiene que tener extensión .sil[16].	

Tabla 16: HU1_T2

Tarea	
Número de la Tarea: HU1_T2	**HU:** Firmar Documentos XML
Nombre: Cargar documento XML a firmar	
Tipo: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio:	**Fecha Fin:**
Responsable: Zoraimi Diaz	
Descripción: El usuario seleccionará de manera visual el/los documento(s) o una carpeta que contenga documentos XML a firmar. Los documentos a cargar deben estar en correspondencia con los plugins instalados (ver HU1_T1).	

Tabla 17: HU1_T3

Tarea	
Número de la Tarea: HU1_T2	**HU:** Firmar Documentos XML

[16] Extensión de los plugins de firma digital del sistema ADVsigner.

Nombre: Firmar documentos XML.	
Tipo: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio:	**Fecha Fin:**
Responsable: Zoraimi Diaz	
Descripción: El usuario seleccionará el documento a firmar previamente cargado en la aplicación (ver HU1_T1). Para realizar esta operación es necesario haber seleccionado con anterioridad un certificado digital del almacén de llaves de la solución. El resultado del proceso será informado al usuario.	

HU Validar Firma

Tabla 18: HU2_T1

Tarea	
Número de la Tarea: HU2_T1	**HU:** Validar Firma de Documentos XML
Nombre: Cargar documentos a validar	
Tipo: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio:	**Fecha Fin:**
Responsable: Zoraimi Diaz	
Descripción: El usuario seleccionará de manera visual el/los documento(s) o una carpeta que contenga documentos XML a validar. Los documentos a cargar deben estar en correspondencia con los plugins instalados.	

Tabla 19: HU2_T2

Tarea	
Número de la Tarea: HU2_T2	**HU:** Validar Firma de Documentos XML
Nombre: Validar documento XML firmado	
Tipo: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio:	**Fecha Fin:**
Responsable: Zoraimi Diaz	
Descripción: El usuario seleccionará el documento XML firmado a validar previamente cargado en la aplicación (ver HU2_T1). Concluido el proceso de validación, se le informará al usuario el resultado.	

3.1.2 Iteración 2

En esta segunda iteración se desarrollan las historias de usuarios con media o baja complejidad. Ver la siguiente tabla con las historias de usuarios definidas para esta iteración.

Tabla 20: HU abordadas en la segunda iteración

Iteración	No. HU	Historia de Usuario	Estimación
Iteración 1	HU1	Firmar Archivos JAR	3
	HU2	Verificar Firma en los Archivos JAR	2

3.1.2.2 Tareas de la ingeniería iteración 2

Tabla 21: Tareas de la ingenieria a segunda iteración

Iteración	No	Historia de Usuario	Tareas
Iteración 2	HU3	Firmar Archivos JAR	– Cargar plugins para firmar documentos JAR. – Cargar archivos JAR a firmar. – Firmar archivo JAR.
	HU4	Verificar Firma en los Archivos JAR	– Cargar archivo JAR a validar. – Validar la firma de archivos JAR

Tareas de la ingeniería detalladas Iteración 2

HU Firmar Archivos JAR

Tabla 22: HU3_T1

Tarea	
Número de la Tarea: HU3_T1	**HU:** Firmar Archivos JAR
Nombre: Cargar Plugins para firmar archivos JAR	
Tipo: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio:	**Fecha Fin:**
Responsable: Zoraimi Díaz	
Descripción: El usuario cargará desde una ubicación el plugin para firmar archivos JAR. El plugin para ser cargado tiene que tener extensión .sil.	

Tabla 23: HU3_T2

Tarea

Número de la Tarea: HU3_T2	**HU:** Firmar Archivos JAR
Nombre: Cargar archivos JAR a firmar	
Tipo: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio:	**Fecha Fin:**
Responsable: Zoraimi Diaz	
Descripción: El usuario seleccionará de manera visual el/los archivos(s) o una carpeta que contenga los archivos JAR a firmar. Los archivos a cargar deben estar en correspondencia con los plugins instalados (ver HU3_T1).	

Tabla 24: HU3_T3

Tarea	
Número de la Tarea: HU3_T3	**HU:** Firmar Archivos JAR
Nombre: Firmar un archivo JAR	
Tipo: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio:	**Fecha Fin:**
Responsable: Zoraimi Díaz	
Descripción: El usuario seleccionará el archivo JAR a firmar previamente cargado en la aplicación (ver HU3_T2). Para realizar esta operación será necesario haber seleccionado un certificado digital del almacén de llaves de la solución. El resultado del proceso será informado al usuario.	

HU Verificar Firma en los Archivos JAR

Tabla 25: HU4_T1

Tarea	
Número de la Tarea: HU4_T1	**HU:** Verificar Firma en los Archivos JAR
Nombre: Cargar archivo JAR a validar.	
Tipo: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio:	**Fecha Fin:**
Responsable: Zoraimi Díaz	
Descripción: El usuario seleccionará de manera visual el/los archivo(s) o una carpeta que contenga los JAR a validar. Los archivos a cargar deben estar en correspondencia con los plugins instalados (ver HU3_T1).	

Tabla 26: HU4_T2

Tarea	
Número de la Tarea: HU4_T2	**HU:** Verificar Firma en los Archivos JAR
Nombre: Validar la firma de archivos JAR	
Tipo: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio:	**Fecha Fin:**
Responsable: Zoraimi Díaz	
Descripción: El usuario seleccionará el archivo JAR firmado a validar previamente cargado en la aplicación (ver HU3_T2). Concluido el proceso de validación, se le informará al usuario el resultado.	

3.2 Pruebas realizadas

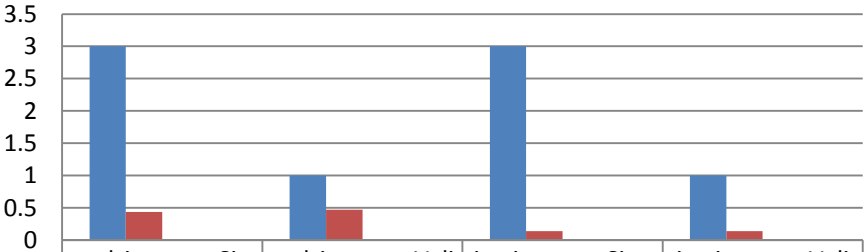
Las pruebas son los procesos que permiten verificar y revelar la calidad de un producto de software. Se utilizan para identificar posibles fallos de implementación, calidad, o usabilidad de un software. XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones.

XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código, diseñada por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida, diseñadas por el cliente final. Cada vez que se da cumplimiento a una o varias historias de usuario se presenta el software debidamente probado al cliente. Además cuando se reúnen un número suficiente de funcionalidades que representan una versión útil y parcialmente completa de la aplicación se produce una liberación, lo cual es una versión funcional de la aplicación que aporta valor al negocio y que debe ser mantenida a la par que se desarrollan las siguientes funcionalidades (Beck, y otros, 2004).

3.2.1 Pruebas Unitarias

El framework JUnit desarrollado por Erich Gamma y Kent Beck, permite realizar de manera controlada este tipo de pruebas a aplicaciones desarrolladas en Java. Se utiliza para evaluar si cada uno de los métodos de las clases se comporta como debería esperarse. Actualmente el IDE NetBeans 6.9 cuenta con la versión 4.5 de JUnit, la cual permite crear pruebas de clases de Java de manera automática, facilitando al programador enfocarse en la prueba y el resultado esperado, dejando a la herramienta la creación de las clases que permiten coordinar las pruebas (Oracle Corporation, 2012).

Se construye un caso de prueba con los métodos críticos de cada clase, de esta manera se logra la disminución del tiempo en el ciclo compilación-ejecución, permitiendo al equipo de desarrollo concentrarse en la codificación de la solución. Además se ejecutan las pruebas cuantas veces sean necesarias, logrando determinar en forma inmediata si el cambio realizado a un módulo del sistema se puede integrar al mismo sin alterar el funcionamiento actual (Beck, y otros, 2004). En el desarrollo de los plugins de firma digital se tienen varias funcionalidades que se verifican mediante casos de pruebas individuales o *test cases* utilizando JUnit, en este caso la herramienta de prueba arroja como resultado el tiempo transcurrido en realizar la prueba, posibles fallas y errores encontrados en los métodos implementados en las clases. En la Figura 19 se muestra un resumen de las pruebas unitarias realizadas a cada una de las clases de los plugins desarrollados. El resultado de las pruebas generadas por la herramienta se muestra en el Anexo I y Anexo II.



	xmlsign.core.SignerTest	xmlsign.core.ValidatorTest	jarsign.core.SignerTest	jarsign.core.ValidatorTest
Cantidad de Métodos	3	1	3	1
Tiempo transcurrido (seg)	0.437	0.469	0.141	0.141
Fallos	0	0	0	0
Errores	0	0	0	0

Figura 19: Resumen de pruebas unitarias

Fuente: (Elaboración propia)

3.2.2 Pruebas de Integración

Según (José A. Mañas, 1994) las pruebas de integración se llevan a cabo durante la construcción del sistema, involucran a un número creciente de módulos y terminan probando el sistema como conjunto. Estas pruebas se plantean desde un punto de vista estructural o funcional, cubren todo el sistema y pretenden cubrir plenamente la especificación de requisitos del usuario. Las pruebas estructurales de integración son similares a las pruebas unitarias, pero trabajan a un nivel conceptual superior. En lugar de referirse a las sentencias del lenguaje, se refiere a llamadas entre módulos, mientras que las pruebas funcionales de integración son similares a las pruebas de aceptación. Aquí se trata de encontrar fallos en la respuesta de un módulo cuando su operación depende de los servicios prestados por otro(s) módulo(s). En la Figura 20 se muestra un resumen del caso de prueba funcional, el cual se revisa 4 veces detectando algunos errores que fueron corregidos y muestra además las dos últimas revisiones realizadas como exitosas debido a que no se encontró ningún error. Para ver el caso de prueba funcional de integración dirigirse a el Anexo IV.

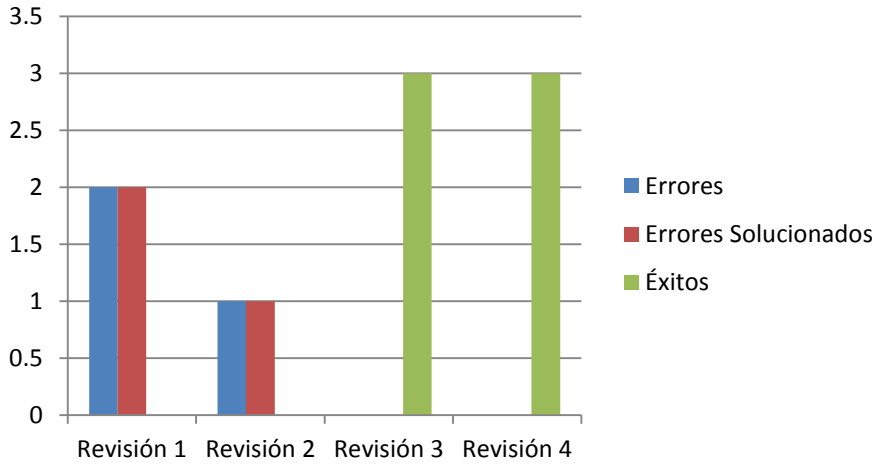


Figura 20: Resumen del caso de prueba de Integración

Fuente: (Elaboración propia)

Las pruebas estructurales de integración se realizan a la clase que implementa la interfaz IPlugin necesaria para cargar los plugin en el sistema ADVsigner, además se prueba el sistema como un conjunto con el objetivo de eliminar fallos y posibles errores en las llamadas entre módulos. Para ver el resultado de este caso de prueba dirigirse al Anexo III.

3.2.3 Pruebas de Aceptación

Las pruebas de aceptación son definidas por el cliente para cada historia de usuario y tienen como objetivo asegurar que las funcionalidades del sistema cumplen con lo que se espera de ellas. Marcan el camino a seguir en cada iteración, indicándole al equipo de desarrollo hacia donde tiene que ir y en qué puntos o funcionalidades debe poner el mayor esfuerzo y atención. "Las pruebas de aceptación permiten al cliente saber cuándo el sistema funciona, y que los programadores conozcan qué es lo que resta por hacer." (Jeffries, 1999). Estas pruebas tienen una importancia crítica para el éxito de una iteración. Por lo tanto el probador debe tenerlas lo antes posible a partir del comienzo de la iteración, y lograr que el cliente las apruebe para poder presentárselas cuanto antes al equipo de desarrollo (Joskowicz, 2008). Para examinar los valores válidos e inválidos de las entradas existentes en el software, se definen cuatro casos de prueba para verificar si el producto satisface los requerimientos del cliente, tal como se describe en las historias de usuario.

Tabla 27: Prueba de aceptación Firmar documentos XML

Caso de Prueba de Aceptación	
Código: HU1_P1	**HU:** Firmar Documentos XML
Nombre: Firmar Documentos XML	
Descripción: Prueba de funcionalidad para la firma de documentos XML	
Condiciones de ejecución: Tener certificados digitales en el almacén de llaves	
Entrada / Pasos de Ejecución: 1. El usuario importa el plugin para firmar este tipo de documento. 2. El usuario importa el/los documento(s) o una carpeta que contenga documentos a firmar a la aplicación. 3. El usuario selecciona el/los documento(s) o una carpeta que desea firmar. 4. Selecciona del almacén de llaves un certificado digital para realizar la firma.	
Resultado esperado: Los documentos son firmados correctamente, se muestra un mensaje informando al usuario del éxito de la operación.	
No conformidades:	
Evaluación de la prueba: Prueba Satisfactoria	

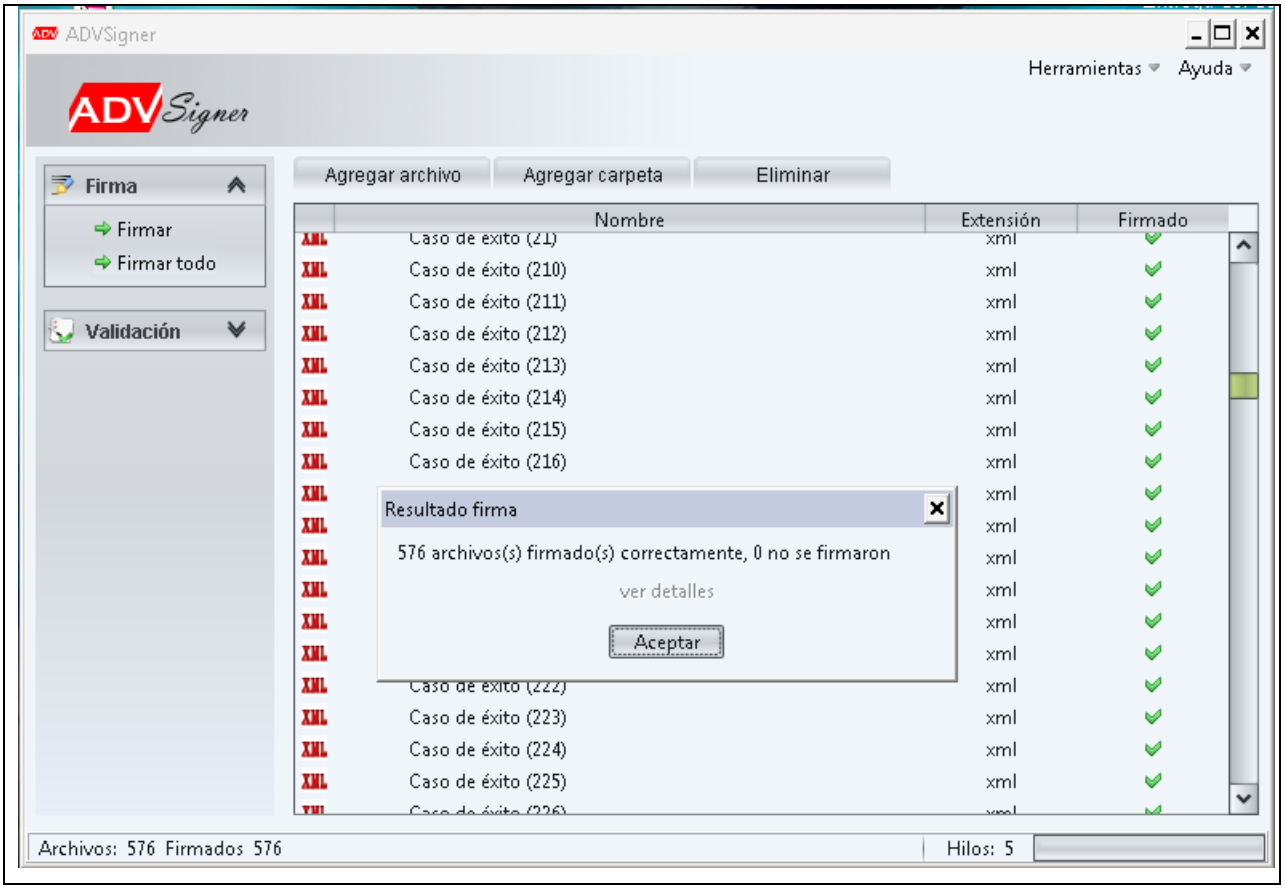


Tabla 28: Prueba de aceptación Validar firma de documentos XML

Caso de Prueba de Aceptación	
Código: HU2_P2	**HU:** Validar Firma de Documentos XML
Nombre: Validar Firma de Documentos XML	
Descripción: Prueba de funcionalidad para la validación de los documentos XML firmados	
Condiciones de ejecución:	
Entrada / Pasos de Ejecución: 1. El usuario selecciona un documento firmado previamente importado a la aplicación, para realizar la validación. 2. El usuario puede validar todos los documentos firmados previamente cargados en la aplicación.	
Resultado esperado: La aplicación informará al usuario el resultado del proceso.	
No conformidades:	
Evaluación de la prueba: Prueba Satisfactoria	

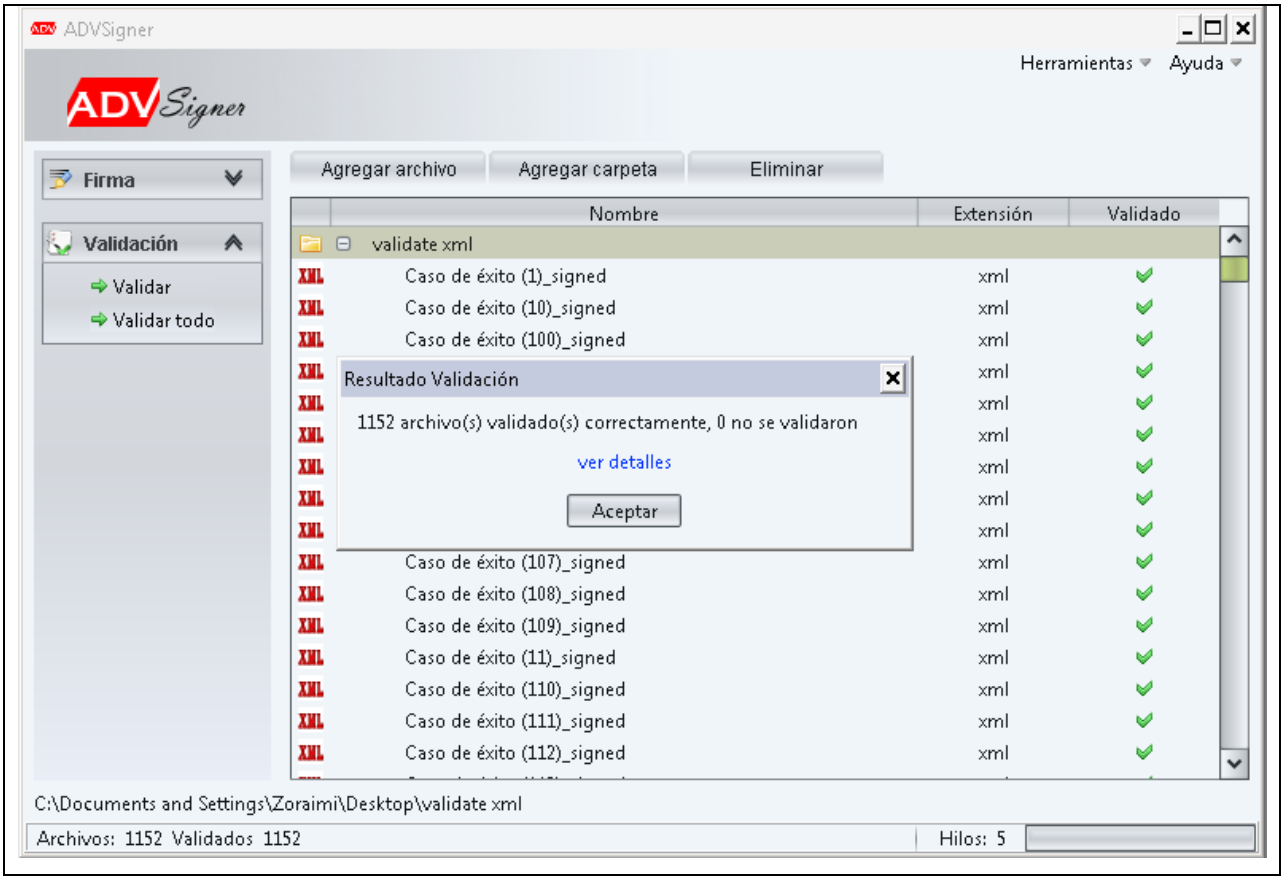


Tabla 29: Prueba de aceptación Firmar archivos JAR

Caso de Prueba de Aceptación	
Código: HU3_P3	**HU:** Firmar Archivos JAR
Nombre: Firmar Archivo JAR	
Descripción: Prueba de funcionalidad para la firma de archivos JAR	
Condiciones de ejecución: Tener certificados digitales en el almacén de llaves	
Entrada / Pasos de Ejecución: 1. El usuario importa el plugin para firmar este tipo de archivo. 2. El usuario importa el/los archivo(s) o una carpeta que contenga archivos a firmar a la aplicación. 3. El usuario selecciona el/los archivo(s) o una carpeta para firmar. 4. Selección de un certificado digital para realizar la firma.	
Resultado esperado: Los archivos JAR son firmados correctamente, se muestra un mensaje informando al usuario del éxito de la operación.	
No conformidades:	

Evaluación de la prueba: Prueba Satisfactoria

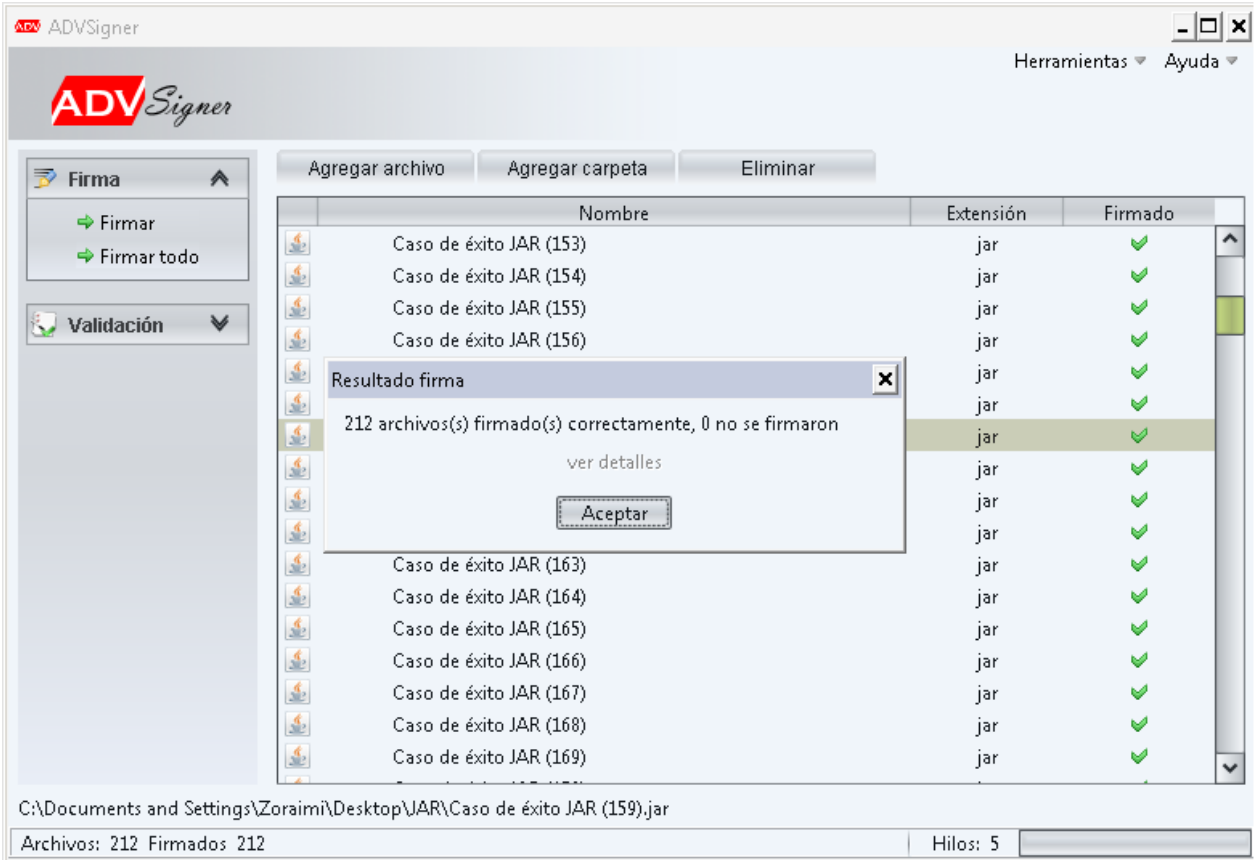
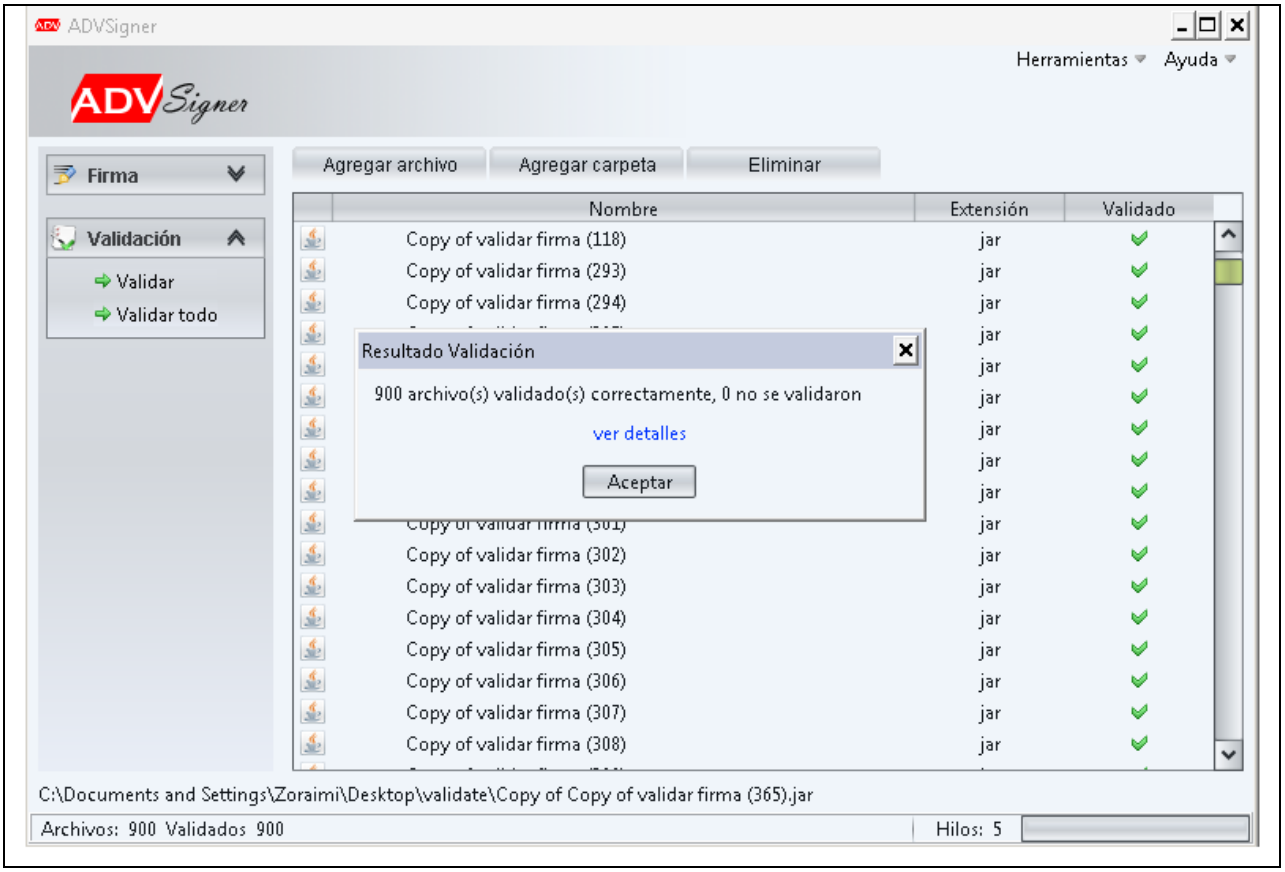
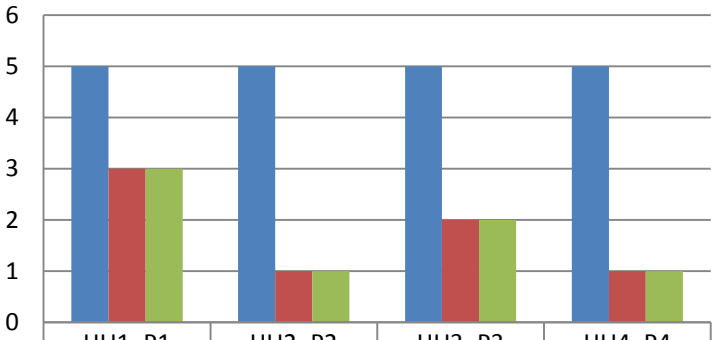


Tabla 30: Prueba de aceptación Verificar firma en los archivos JAR

Caso de Prueba de Aceptación	
Código: HU4_P4	**HU:** Firma en los Archivos JAR
Nombre: Verificar Firma en los Archivos JAR	
Descripción: Prueba de funcionalidad para la validación de los archivos JAR firmados	
Condiciones de ejecución:	
Entrada / Pasos de Ejecución: 1. El usuario selecciona un archivo JAR firmado previamente importado a la aplicación, para realizar la validación. 2. El usuario puede validar todos los archivos firmados previamente cargados en la aplicación.	
Resultado esperado: La aplicación informará al usuario el resultado del proceso.	
No conformidades:	
Evaluación de la prueba: Prueba Satisfactoria	



Estos casos de pruebas se ejecutan cinco veces cada uno, detectando para cada funcionalidad los errores encontrados y los solucionados. La Figura 21 muestra un resumen de los casos de pruebas de aceptación.

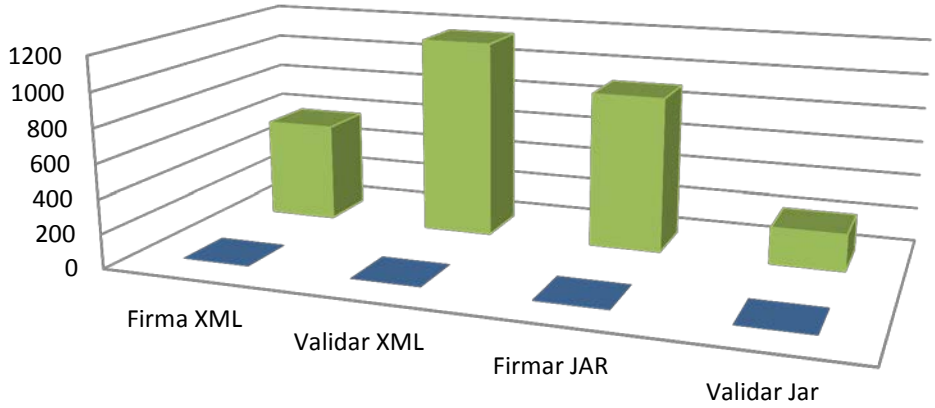


	HU1_P1	HU2_P2	HU3_P3	HU4_P4
Revisiones Ejecutadas	5	5	5	5
Errores	3	1	2	1
Errores solucionados	3	1	2	1

Figura 21: Resumen de prueba de aceptación

Fuente: (Elaboración Propia)

Para comprobar el éxito de los casos de prueba de aceptación también se ejecutan estas pruebas en base a los valores límites de entrada que soporta la solución ADVsigner, teniendo en cuenta las condiciones del hardware en el que se prueban. Esto se realiza con el objetivo de saber la cantidad de documentos y archivos electrónicos que se pueden firmar y validar en un período de tiempo determinado tratando de no provocar fallos en el rendimiento del sistema. En la siguiente Figura 22 se muestra un resumen con la cantidad de documentos y archivos electrónicos que se pueden firmar y validar en un plazo de tiempo determinado, no se recomienda utilizar valores mayores que esto pues el sistema comienza a presentar fallas.



	Firma XML	Validar XML	Firmar JAR	Validar Jar
Tiempo transcurrido (minutos)	2	0.3	4	1
Cantidad de documetos y archivos firmados	576	1152	900	212

Figura 22: Resumen caso de éxito para ambos plugins

Fuente: (Elaboración Propia)

3.3 Conclusiones parciales

En este capítulo se abordaron los temas correspondientes a la Implementación y Pruebas de todo el trabajo realizado, obteniendo varios artefactos que posibilitaron llegar a las siguientes conclusiones:

- Las tareas de ingeniería fue una buena práctica de desarrollo que les mostró a los programadores las funcionalidades específicas a implementar en cada una de las iteraciones.

- Con el diseño de pruebas unitarias, integración y aceptación se logró probar cada una de las funcionalidades que componen los plugins de firma digital, permitiendo así validar la solución propuesta.

Conclusiones Generales

Después de haber realizado la investigación y llevar a cabo el desarrollo del software se arribaron a las siguientes conclusiones:

- Las herramientas y lenguajes utilizados en el desarrollo del sistema ADVsigner permitieron diseñar, implementar y probar el producto de la manera más factible posibilitando reducir el tiempo de trabajo en el equipo de desarrollo. Además la metodología de software seleccionada permitió establecer las diferentes fases del ciclo de vida del proyecto.

- La identificación de los requisitos funcionales a través de historias de usuario sirvió para estimar el tiempo de desarrollo de los plugin y en la etapa de prueba para verificar que el producto cumple con las funcionalidades especificadas.

- Las implementaciones del estándar W3C para firmar documentos XML y la especificación de Java para los archivos JAR proporcionaron una guía a tener en cuenta durante el proceso de desarrollo de los plugins de firma digital.

- La verificación del funcionamiento de cada una de las funcionalidades a través de los casos de pruebas diseñados permitió la detección de no conformidades y la corrección de las mismas durante el desarrollo de los plugins, obteniendo un producto con la calidad requerida.

Por lo anteriormente expuesto este trabajo finaliza dando cumplimento a los objetivos trazados en el inicio de la investigación. Se logró el desarrollo de nuevos plugins de firma digital para la solución de firma ADVsigner realizados con el objetivo de garantizar autenticidad, integridad y no repudio de la información transmitida en documentos XML y archivos JAR.

Recomendaciones

Después del estudio realizado para la construcción de los plugins de firma digital y en vista de seguir perfeccionando el sistema ADVsigner, sería recomendable tener en cuenta los siguientes aspectos.

- Investigar e implementar los perfiles de seguridad que brinda la firma avanzadas XML (XAdES) en la solución propuesta.
- Seguir desarrollando otras extensiones o plugin para firmar nuevos formatos de documentos y archivos electrónicos.

Referencias Bibliográficas

Adobe Systems. 2012. *Archivos PDF | Formato de documento portátil de Adobe: Acrobat.* [En línea] 2012. [Citado el: 30 de mayo de 2012.] http://www.adobe.com/es/products/acrobat/adobepdf.html.

Agile Alliance, Menber. 2010. Programación Extrema. [En línea] 14 de octubre de 2010. [Citado el: 10 de enero de 2012.] http://www.programacionextrema.org/.

Apache Santuario. 2012. Welcome to Apache Santuario. [En línea] 2012. [Citado el: 9 de marzo de 2012.] www.apache.org/.

Bddigital. 2009. Modelo de Dominio. [En línea] 2009. [Citado el: 22 de marzo de 2012.] <http://bdigital.eafit.edu.co/bdigital/.

Beck, Kent y Andres, Cynthia. 2004. *Extreme Programming Explained: Embrace Change (2nd Edition) (Paperback).* 2004.

Crispin Lisa. 2001. Extreme Rules of the Road. [En línea] 2001. [Citado el: 19 de abril de 2012.] http://www.testing.com/agile/crispin-xp-article.pdf.

Dra. María José Ruiz. 2003. La firma electronica: Mayor seguridad en la red. [En línea] 2003. [Citado el: 20 de abril de 2012.] http://www.delitosinformaticos.com/firmaelectronica/fe-seguridad.pdf.shtml.

Flower. 2010. Java Foundations: Java - Estándares de programación:. [En línea] 2 de julio de 2010. [Citado el: 2 de junio de 2012.] http://javafoundations.blogspot.com/2010/07/java-estandares-de-programacion.html.

Gobierno de Buenos Aires. 2010. Algoritmos de firma digital. [En línea] 2010. [Citado el: 27 de octubre de 2011.] http://www.buenosaires.gob.ar/areas/sistemas/FirmaDigital.pdf.

Gobierno de Chile. 2011. *Documento de Recomendación de Uso de Firma Digital en Comunicación PISEE.* Santiago de Chile : Ministerio Secretaría General de la Presidencia, 2011.

Goncalves, Luis. 2011. Welcome XAdES4J the Project. [En línea] 2011. http://code.google.com/p/xades4j/.

Jeffries, Ron. 1999. Extreme Testing. [En línea] 1999. [Citado el: 18 de marzo de 2012.] http://www.xprogramming.com/publications/SP99 Extreme for Web.pdf.

José A. Mañas. 1994. *Prueba de Programas.* 1994.

José H. Canós, Patricio Letelier y Mª Carmen Penadés. 2009. "Métodologías Ágiles" en el Desarrollo de Software. [En línea] 2009. [Citado el: 19 de abril de 2012.] http://www.willydev.net/descargas/prev/TodoAgil.Pdf.

Joskowicz, José. 2008. *Reglas y Prácticas en eXtreme Programming.* España : Universidad de Vigo, 2008.

Netscape Corporation. 2012. THE JAR FORMAT. [En línea] 2012. [Citado el: 10 de mayo de 2012.] http://isp.vsi.ru/library/Java/jarfile/jar.htm&usg=ALkJrhgVckXeJEV2wnDyr4zQfGAZLY0EnA#421768.

OMG, Object Management Group. 2012. UML. [En línea] 2012. [Citado el: 30 de abril de 2012.] http://www.uml.org/.

Oracle Corporation. 2010. Especificación archivos JAR. [En línea] 2010. [Citado el: 25 de marzo de 2012.] http://docs.oracle.com/javase/7/docs/technotes/guides/jar/jar.html&usg=ALkJrhjWyKPb9O4E-l39_UBqyXLypSwWnA.

—. 2010. Java XML Digital Signatures. [En línea] 2010. [Citado el: 25 de marzo de 2012.] http://java.sun.com/developer/technicalArticles/xml/dig_signatures/.

—. 2011. JSRs: Java Specification Requests. *JSR 105: XML Digital Signature APIs.* [En línea] 2011. [Citado el: 15 de mayo de 2012.] http://jcp.org/en/jsr/detail?id=105.

—. 2010. The Java Archive (JAR) File Format. [En línea] 2010. [Citado el: 17 de julio de 2011.] http://java.sun.com/developer/Books/javaprogramming/JAR/basics/.

—. 2012. Writing JUnit Tests in NetBeans IDE. [En línea] 2012. [Citado el: 5 de mayo de 2012.] http://netbeans.org/kb/docs/java/junit-intro.html.

P.Letelier. *Rational Unified Process (RUP).* España : Universidad Politécnica de Valencia.

Paradigm, Visual. 2011. Visual Paradigm for UML. [En línea] 2011. [Citado el: 22 de 03 de 2012.] http://www.visual-paradigm.com/.

Pásaro Méndez, Iago. 2006. La firma digital: seguridad del empresario en la red. [En línea] 2006. [Citado el: 2 de octubre de 2011.] http://noticias.juridicas.com/articulos/20-Derecho%20Informatico/200604-4959142921063871.html.

Reynoso, Carlos. 2004. Metodologías de Desarrollo de Software Ágiles. [En línea] 2004. [Citado el: 19 de mayo de 2012.] http://www.sel.unsl.edu.ar/ApuntesMaes/2004/Metodologias%20Agiles.doc.

Secretaría Nacional De Administración Pública Ecuador. 2009. Firma y certificación digital en varios países latinoamericanos. [En línea] 03 de julio de 2009. [Citado el: 04 de febrero de

2012.] http://www.informatica.gob.ec/index.php/destacadoshistorico/gobierno-historico-destacados/254-firma-y-certificacion-digital-en-varios-paises-latinoamericanos.

SGP, Autoridad Certificante de la Subsecretaría de la Gestión Pública Argentina. 2011. PKI Firma Digital FAQ - Preguntas más frecuentes. [En línea] 2011. [Citado el: 2012 de 04 de 10.] http://ca.sgp.gov.ar/faq.html.

Silot Ochoa, Dariel. 2011. *Extensión de arquitectura para sistema de firma digital del CISED.* Ciudad de la Habana : Universidad de la Ciencias Informáticas, 2011.

Sun Microsystems. 1997. JAR Guía. [En línea] 1997. [Citado el: 16 de marzo de 2012.] http://info.krc.karelia.ru/java/docs/guide/jar/index.html.

Torrijos, Ricard Lou. 2011. Programación en Castellano. *Ficheros JAR (Java ARchives).* [En línea] 2011. [Citado el: 22 de 03 de 2008.] http://www.programacion.com/articulo/ficheros_jar_java_archives_92/10.

VeriSign. 2011. Seguridad XML: Su Importancia en el E-Comercio. [En línea] 2011. http://www.razonypalabra.org.mx/anteriores/n49/bienal/Mesa%205/XML.pdf.

Vlamir Rodríguez Fernández, Lianet Pineda De la Nuez, Yarina Amoroso Fernández3. 2011. Uso de la firma digital, aspectos legales que afectan en el entorno cubano. [En línea] 2011. [Citado el: 5 de junio de 2012.] http://www.bibliociencias.cu/gsdl/collect/eventos/archives/HASH019d/9077ffcf.dir/doc.pdf.

W3C. 2003. Extensible Markup Language (XML). [En línea] 2003. [Citado el: 10 de enero de 2012.] http://www.w3.org/XML/.

—. 2003. XML Advanced Electronic Signatures (XAdES). [En línea] 02 de 2003. [Citado el: 10 de enero de 2012.] http://www.w3.org/TR/XAdES/.

—. 2008. XML Signature Syntax and Processing (Second Edition). [En línea] 10 de junio de 2008. [Citado el: 4 de mayo de 2011.] http://www.w3.org/TR/xmldsig-core/.

Bibliografía

Adobe Systems. 2012. Archivos PDF | Formato de documento portátil de Adobe: Acrobat. [En línea] 2012. [Citado el: 30 de mayo de 2012.] http://www.adobe.com/es/products/acrobat/adobepdf.html.

Agile Alliance, Menber. 2010. Programación Extrema. [En línea] 14 de octubre de 2010. [Citado el: 10 de enero de 2012.] http://www.programacionextrema.org/.

Álvarez Marañón, G. 2003. Seguridad en servicios web XML. [En línea] 2003. [Citado el: 19 de abril de 2012.] http://www.iec.csic.es/criptonomicon/boletines/boletin90.txt.

Apache Santuario. 2012. Welcome to Apache Santuario. [En línea] 2012. [Citado el: 9 de marzo de 2012.] www.apache.org/.

Bddigital. 2009. Modelo de Dominio. [En línea] 2009. [Citado el: 22 de marzo de 2012.] <http://bdigital.eafit.edu.co/bdigital/.

Beck, Kent y Andres, Cynthia. 2004. Extreme Programming Explained: Embrace Change (2nd Edition) (Paperback). 2004.

Crispin Lisa. 2001. Extreme Rules of the Road. [En línea] 2001. [Citado el: 19 de abril de 2012.] http://www.testing.com/agile/crispin-xp-article.pdf.

Dra. María José Ruiz. 2003. La firma electronica: Mayor seguridad en la red. [En línea] 2003. [Citado el: 20 de abril de 2012.] http://www.delitosinformaticos.com/firmaelectronica/fe-seguridad.pdf.shtml.

Flower. 2010. Java Foundations: Java - Estándares de programación:. [En línea] 2 de julio de 2010. [Citado el: 2 de junio de 2012.] http://javafoundations.blogspot.com/2010/07/java-estandares-de-programacion.html.

Gobierno de Buenos Aires. 2010. Algoritmos de firma digital. [En línea] 2010. [Citado el: 27 de octubre de 2011.] http://www.buenosaires.gob.ar/areas/sistemas/FirmaDigital.pdf.

Gobierno de Chile. 2011. Documento de Recomendación de Uso de Firma Digital en Comunicación PISEE. Santiago de Chile : Ministerio Secretaría General de la Presidencia, 2011.

Goncalves, Luis. 2011. Welcome XAdES4J the Project. [En línea] 2011. http://code.google.com/p/xades4j/.

IETF. 2008. XML-Signature Syntax and Processing. [En línea] 2008. [Citado el: 9 de marzo de 2012.] http://www.ietf.org/rfc/rfc3275.txt.

Jeffries, Ron. 1999. Extreme Testing. [En línea] 1999. [Citado el: 18 de marzo de 2012.] http://www.xprogramming.com/publications/SP99 Extreme for Web.pdf.

Jeffries, Ron, Anderson, Ann and Hendrickson, Chet. 2000. Extreme Programming Installed. 2000. 0-201-70842-6.

José A. Mañas. 1994. Prueba de Programas. 1994.

José H. Canós, Patricio Letelier y Mª Carmen Penadés. 2009. "Métodologías Ágiles" en el Desarrollo de Software. [En línea] 2009. [Citado el: 19 de abril de 2012.] http://www.willydev.net/descargas/prev/TodoAgil.Pdf.

Joskowicz, José. 2008. Reglas y Prácticas en eXtreme Programming. España : Universidad de Vigo, 2008.

Lopez, Manuel Jose Lucena. 2011. Criptografía y seguridad. s.l. : Universidad de Jaen, 2011. Versión 4-0.9.0.

Netscape Corporation. 2012. THE JAR FORMAT. [En línea] 2012. [Citado el: 10 de mayo de 2012.] http://isp.vsi.ru/library/Java/jarfile/jar.htm&usg=ALkJrhgVckXeJEV2wnDyr4zQfGAZLY0EnA#421768.

OMG, Object Management Group. 2012. UML. [En línea] 2012. [Citado el: 30 de abril de 2012.] http://www.uml.org/.

Oracle Corporation. 2010. Especificación archivos JAR. [En línea] 2010. [Citado el: 25 de marzo de 2012.] http://docs.oracle.com/javase/7/docs/technotes/guides/jar/jar.html&usg=ALkJrhjWyKPb9O4E-l39_UBqyXLypSwWnA.

—. 2011. Java Platform Standard Edition 7 Documentación. [En línea] 2011. [Citado el: 25 de marzo de 2012.] http://www.oracle.com/technetwork/java/javase/downloads/jdk-7-netbeans-download-432126.html.

—. 2011. Java XML Digital Signature API Specification (JSR 105). [En línea] 2011. [Citado el: 25 de marzo de 2012.] http://docs.oracle.com/javase/7/docs/technotes/guides/security/xmldsig/overview.html.

—. 2011. JSRs: Java Specification Requests. JSR 105: XML Digital Signature APIs. [En línea] 2011. [Citado el: 15 de mayo de 2012.] http://jcp.org/en/jsr/detail?id=105.

—. 2010. The Java Archive (JAR) File Format. [En línea] 2010. [Citado el: 17 de julio de 2011.] http://java.sun.com/developer/Books/javaprogramming/JAR/basics/.

—. 2010. Uso de los archivos JAR: conceptos básicos. [En línea] 2010. [Citado el: 26 de marzo de 2012.] http://java.sun.com/developer/Books/javaprogramming/JAR/basics/.

—. 2012. Writing JUnit Tests in NetBeans IDE. [En línea] 2012. [Citado el: 5 de mayo de 2012.] http://netbeans.org/kb/docs/java/junit-intro.html.

P.Letelier. Rational Unified Process (RUP). España : Universidad Politécnica de Valencia.

Paradigm, Visual. 2011. Visual Paradigm for UML. [En línea] 2011. [Citado el: 22 de 03 de 2012.] http://www.visual-paradigm.com/.

Pásaro Méndez, Iago. 2006. La firma digital: seguridad del empresario en la red. [En línea] 2006. [Citado el: 2 de octubre de 2011.] http://noticias.juridicas.com/articulos/20-Derecho%20Informatico/200604-4959142921063871.html.

Quiñones Bondartchuk , Roberto. 2009. Firma Digital de Documentos utilizando Smart Cards. Ciudad Habana : Universidad de las Ciencias Informáticas, 2009.

Reynoso, Carlos. 2004. Metodologías de Desarrollo de Software Ágiles. [En línea] 2004. [Citado el: 19 de mayo de 2012.] http://www.sel.unsl.edu.ar/ApuntesMaes/2004/Metodologias%20Agiles.doc.

Robin Cover. 2005. Cover Pages: XML Applications and Initiatives:. [En línea] 25 de junio de 2005. [Citado el: 10 de junio de 2012.] http://xml.coverpages.org/xmlApplications.html.

RSA Laboratories. 2009. PKCS #11: Cryptographic Token Interface Standard. [En línea] 2.30, 28 de octubre de 2009. [Citado el: 17 de enero de 2011.] ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-30/pkcs-11v2-30-d1.pdf.

—. 2010. PKCS #12: Personal Information Exchange Syntax Standard. [En línea] 2010. [Citado el: 1 de junio de 2012.] ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-12/pkcs-12v1.pdf.

Secretaría Nacional De Administración Pública Ecuador. 2009. Firma y certificación digital en varios países latinoamericanos. [En línea] 03 de julio de 2009. [Citado el: 04 de febrero de 2012.] http://www.informatica.gob.ec/index.php/destacadoshistorico/gobierno-historico-destacados/254-firma-y-certificacion-digital-en-varios-paises-latinoamericanos.

SGP, Autoridad Certificante de la Subsecretaría de la Gestión Pública Argentina. 2011. PKI Firma Digital FAQ - Preguntas más frecuentes. [En línea] 2011. [Citado el: 2012 de 04 de 10.] http://ca.sgp.gov.ar/faq.html.

Shalloway, Alan and Trott, James. 2004. Designs Patterns Explained. 2004. ISBN-13: 978-0321247148.

Silot Ochoa, Dariel. 2011. Extensión de arquitectura para sistema de firma digital del CISED. Ciudad de la Habana : Universidad de la Ciencias Informáticas, 2011.

Sun Microsystems. 1997. JAR Guía. [En línea] 1997. [Citado el: 16 de marzo de 2012.] http://info.krc.karelia.ru/java/docs/guide/jar/index.html.

Tamara Toro. 2011. Firma digital: sus avances y penetración en Latinoamérica. [En línea] 10 de febrero de 2011. [Citado el: 03 de abril de 2012.] http://tecno.americaeconomia.com/noticias/firma-digital-sus-avances-y-penetracion-en-latinoamerica.

Torrijos, Ricard Lou. 2011. Programación en Castellano. Ficheros JAR (Java ARchives). [En línea] 2011. [Citado el: 22 de 03 de 2008.] http://www.programacion.com/articulo/ficheros_jar_java_archives_92/10.

Universidad Argentina,John F. Kennedy. 2011. The Relational XML database project. [En línea] 2011. [Citado el: 10 de marzo de 2012.] http://eprints.rclis.org/bitstream/10760/5696/1/AHDI_Madrid16.pdf.

VeriSign. 2011. Seguridad XML: Su Importancia en el E-Comercio. [En línea] 2011. http://www.razonypalabra.org.mx/anteriores/n49/bienal/Mesa%205/XML.pdf.

Vlamir Rodríguez Fernández, Lianet Pineda De la Nuez, Yarina Amoroso Fernández3. 2011. Uso de la firma digital, aspectos legales que afectan en el entorno cubano. [En línea] 2011. [Citado el: 5 de junio de 2012.] http://www.bibliociencias.cu/gsdl/collect/eventos/archives/HASH019d/9077ffcf.dir/doc.pdf.

W3C. 2003. Extensible Markup Language (XML). [En línea] 2003. [Citado el: 10 de enero de 2012.] http://www.w3.org/XML/.

—. 2003. XML Advanced Electronic Signatures (XAdES). [En línea] 02 de 2003. [Citado el: 10 de enero de 2012.] http://www.w3.org/TR/XAdES/.

—. 2008. XML Signature Syntax and Processing (Second Edition). [En línea] 10 de junio de 2008. [Citado el: 4 de mayo de 2011.] http://www.w3.org/TR/xmldsig-core/.

Glosario de Término

A

Applets: Es un programa, generalmente de tamaño pequeño, creado en el lenguaje Java, que forma parte de una página web.

API: Interfaz de programación de aplicaciones es un conjunto de funciones y procedimientos o métodos (si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

B

Base64: Es un sistema de numeración posicional que usa 64 como base. Está diseñado para representar secuencias arbitrarias de octetos en una forma que no es humanamente legible. Se utiliza en aplicaciones de correo basadas en la privacidad, tal como se define en el RFC 1113.

C

Certificado Digital: También conocido como certificado de clave pública o certificado de identidad es un documento digital mediante el cual un tercero confiable (una autoridad de certificación) garantiza la vinculación entre la identidad de un sujeto o entidad (por ejemplo: nombre, dirección y otros aspectos de identificación) y una clave pública.

Criptografía: Técnica que trata de ocultar las representaciones caligráficas de una lengua.

I

IDE: Integrated Development Environment o entorno de desarrollo integrado es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, poder utilizarse para varios.

IEEE: Institute of Electrical and Electronics Engineers o Instituto de Ingenieros Eléctricos y Electrónicos es una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas. Es la mayor asociación internacional sin ánimo de lucro formada por profesionales de las nuevas tecnologías, como ingenieros eléctricos, ingenieros en electrónica, científicos de la computación, matemáticos aplicados, ingenieros en informática, ingenieros en biomédica e ingenieros en telecomunicación.

P

Plugin: Son programas que se agregan a otro ya existente para ofrecer una nueva función. Estos programas no funcionan de forma independiente, necesitan de un programa instalado previamente en el ordenador donde ellos se instalarán.

R

RSS: Siglas en ingles de Really Simple Syndication, RSS es un formato XML para sindicar o compartir contenido en la web, es parte de la familia de los formatos XML desarrollado específicamente para todo tipo de sitios que se actualicen con frecuencia y por medio del cual se puede compartir la información y usarla en otros sitios web o programa

S

Sellado de Tiempo: El sellado de tiempo o timestamping en inglés es un mecanismo online que permite demostrar que una serie de datos han existido y no han sido alterados desde un instante específico en el tiempo. Este protocolo se describe en el RFC 3161 y está en el registro de estándares de Internet.

SOAP: Siglas de Simple Object Access Protocoles un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Es uno de los protocolos utilizados en los servicios Web.

U

URI: Uniform Resource Identifier, por sus siglas en ingles identificador uniforme de recurso, es una cadena de caracteres corta que identifica un recurso accesible en una red o sistema.

X

XMLDsig: Estándar de sintaxis y procesamiento de firmas digital XML recomendado por la W3C.

XAdES: Sigla en inglés de XML Advanced Electronic Signatures (Firma electrónica avanzada XML) es un conjunto de extensiones a las recomendaciones XMLDsig haciéndolas adecuadas para la firma electrónica avanzada.

XHTML: Siglas del inglés eXtensible HyperText Markup Language. XHTML es básicamente HTML expresado como XML.

W

W3C: World Wide Web Consortium, abreviado W3C es un consorcio internacional que produce recomendaciones para la World Wide Web.

Anexos

Anexo I: Resultado de aplicar las pruebas unitarias al plugin JARSigner

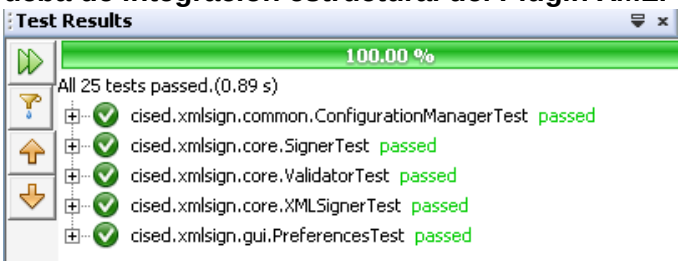
```
Test cised.jarsign.core.JarSignerTest
Testsuite: cised.jarsign.core.SignerTest
Tests run: 3, Failures: 0, Errors: 0, Time elapsed: 0.141 sec
------------- Standard Output ---------------
Load
getHash
setHash
------------- ---------------- ---------------

Test cised.jarsign.core.SignerTest
Testsuite: cised.jarsign.core.ValidatorTest
Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.141 sec
------------- Standard Output ---------------
verifySignatures
------------- ---------------- ---------------
```

Anexo II: Resultado de aplicar las pruebas unitarias al plugin XMLSign

```
Testsuite: cised.xmlsign.core.SignerTest
Tests run: 3, Failures: 0, Errors: 0, Time elapsed: 0.437 sec
------------- Standard Output ---------------
Load
getHash
setHash
------------- ---------------- ---------------

Testsuite: cised.xmlsign.core.ValidatorTest
Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.469 sec
------------- Standard Output ---------------
verifySignatures
------------- ---------------- ---------------
```

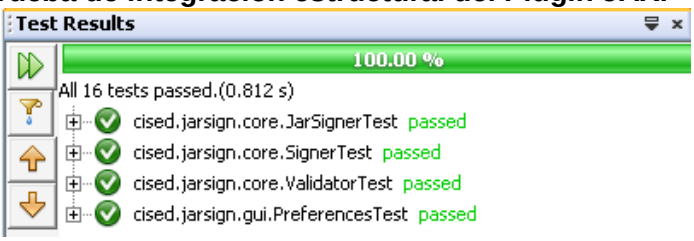
Anexo III: Resultados de las pruebas de integración estructurales.

Resultados de las prueba de integración estructural del Plugin XML.




```
Testsuite: cised.xmlsign.common.ConfigurationManagerTest
Tests run: 9, Failures: 0, Errors: 0, Time elapsed: 0.156 sec

Testsuite: cised.xmlsign.core.SignerTest
Tests run: 3, Failures: 0, Errors: 0, Time elapsed: 0.156 sec

Testsuite: cised.xmlsign.core.ValidatorTest
Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.141 sec

Testsuite: cised.xmlsign.core.XMLSignerTest
Tests run: 11, Failures: 0, Errors: 0, Time elapsed: 0.312 sec

Testsuite: cised.xmlsign.gui.PreferencesTest
Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.125 sec
```

Resultados de las prueba de integración estructural del Plugin JAR.




```
Testsuite: cised.xmlsign.common.ConfigurationManagerTest
Tests run: 9, Failures: 0, Errors: 0, Time elapsed: 0.156 sec

Testsuite: cised.xmlsign.core.SignerTest
Tests run: 3, Failures: 0, Errors: 0, Time elapsed: 0.156 sec

Testsuite: cised.xmlsign.core.ValidatorTest
Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.141 sec

Testsuite: cised.xmlsign.core.XMLSignerTest
Tests run: 11, Failures: 0, Errors: 0, Time elapsed: 0.312 sec

Testsuite: cised.xmlsign.gui.PreferencesTest
Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.125 sec
```

Anexo IV: Pruebas de integración funcionales.

Pruebas Funcionales

Caso de Prueba Integridad
Nombre: Cargar plugins en el sistemas ADVsigner
Descripción: Prueba para integrar los plugins XML y JAR en la solución ADVsigner
Condiciones de ejecución:
Entrada / Pasos de Ejecución: 1. El usuario selecciona en el menú herramientas de las solución ADVsigner la opción Plugins 2. Se abre una interfaz con diferentes opciones entre las que se encuentra: Importar, Eliminar y Desactivar. 3. El usuario seleccioná la opción Importar, posteriormente se abre una ventana para buscar el plugin XML o JAR ubicados en un archivo de la computadora, luego de encontrados se acepta la ventana y finalmente se muestran los plugins cargados en el sistemas ADVsigner.
Resultado esperado: Se muestran las interfaz Opciones con los nuevos plugins de firma instalados, XMLSign y JARSign
No conformidades:
Evaluación de la prueba: Prueba Satisfactoria