Universidad de las Ciencias Informáticas

Facultad 1

Título: Módulo de OpenERP para firmas digitales de documentos.

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

Autores: Lissette Valdés García

Yander Morfa González

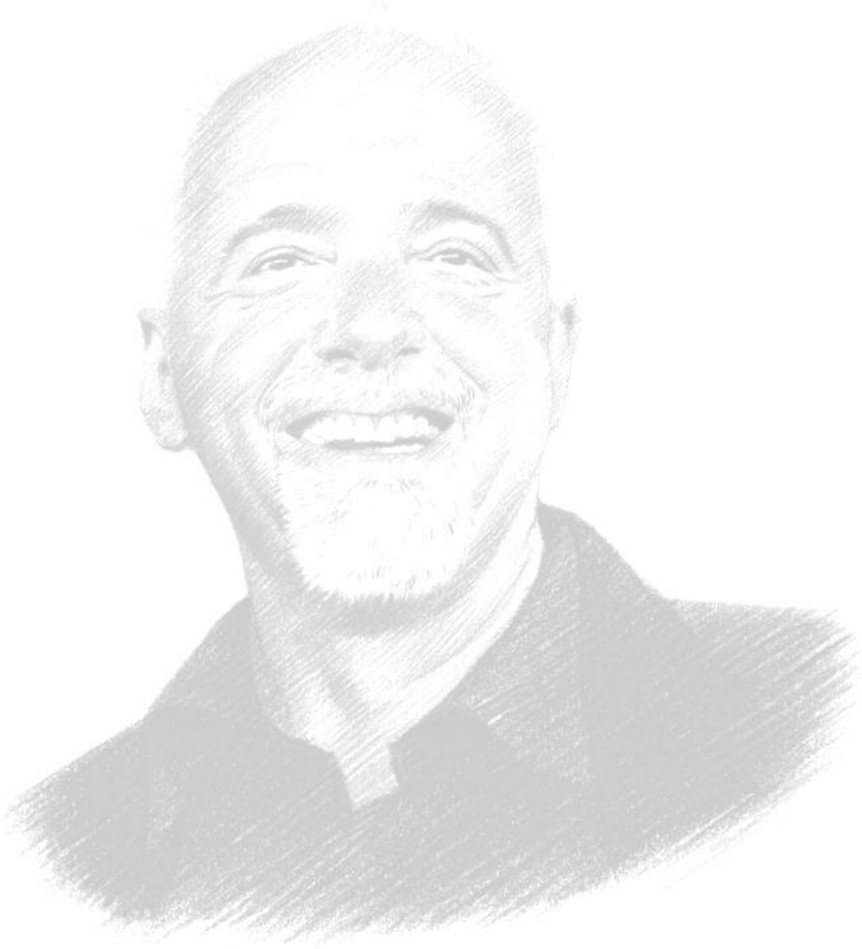
Tutores: MsC. Alexander Hernández Chapman

Ing. Jenny Crespo Cabezas

Consultante: MsC. Jorge Landrian García

La Habana, Junio, 17 de 2014

"Año 56 de la Revolución"



Nadie está a salvo de las derrotas. Pero es mejor perder algunos combates en la lucha por nuestros sueños, que ser derrotado sin saber siquiera por qué se está luchando.

Paulo Coelho

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ _____

Autor: Lissette Valdés García **Autor: Yander Morfa González**

_____ _____

Tutor: MsC. Alexander Hernández Chapman **Tutor: Ing. Jenny Crespo Cabezas**

Tutor: MsC. Alexander Hernández Chapman

Correo electrónico: alx@uci.cu

Tutor: Ing. Jenny Crespo Cabezas

Correo electrónico: jcrespo@uci.cu

Agradecimientos

A la persona que me ha dado la vida, mi madre, por todo su amor y apoyo incondicional, por confiar en mí y en que podía realizar este sueño que hoy es una realidad.

A mi padrastro, por enseñarme que el verdadero amor de padre nace del corazón, por su dedicación mientras mi madre estuvo lejos de nosotros y por darme todos los gustos.

A mi hermanito Leonardo, por ser esa personita especial que me enseñó que debemos decirle siempre "te quiero "a las personas que amamos.

A mis hermanos, Diover y Dober, a prima Yanielis, a mis tías Leonor, Loreto y Jenny, a mi papá y al resto de mi familia, por brindarme su apoyo cada vez que lo necesito.

A mi compañero de tesis Yander, por ser mi amigo y hermano durante estos cinco años de la carrera, por sus buenos consejos y por regañarme cuando era necesario.

A Yosvany, por robarme siempre una sonrisa cuando estaba triste, por hacer de mí una mejor persona y vivir momentos maravillosos a mi lado.

A mis grandes amigas Nadia y Yaiselis, por sus consejos, por preocuparse y estar conmigo en las buenas y en las malas.

A mis amigos Addis, Osmery, Gisselle, Frank, Yunier, Lianet y Abel por compartir y ser parte de mí.

A todos mis profesores de la Universidad que de una forma u otra han contribuido a mi formación como profesional.

Muchas gracias a todos los que han formado parte de mi vida.

Lissette Valdés

A mi familia por permitirme estar aquí.

Agradecimientos

A mi compañera de tesis por aguantarme todos estos años, por permitirme que hoy esté graduándome, ha sido de vital importancia en mi carrera, ha llegado a ser mi amiga, hermana y consejera, espero que nunca me olvide.

A los amigos del fútbol por compartir momentos muy bonitos, espero que nuestra amistad perdure en mi recuerdo, Yalber, Oscar, Rafael, Pollo, Reldy, Lissuan, Henry, Magdiel y Leo.

Al tribunal y oponente por sus consideraciones y grado de perfección durante la tesis.

A las enfermeras del hospital por ser tan pacientes conmigo, además de estar todo el tiempo a mi lado aconsejándome, Mailevys, Iraida, Cari, y al oftalmólogo Arian.

A los gastronómicos Leonel el chino, Lucy, Pedrito, Yoel, Jose, Rubio y el flaco, por todo lo que han hecho por mí en estos 5 años.

A todos muchas gracias….

Yander Morfa

A mi madre Norma, por ser mi guía y mi fuerza para lograr las metas que me propongo en la vida, por todo su amor y los sabios consejos que me ha dado.

A Adrián, por tratarme como si fuera su hija, por su preocupación y por velar que no me falte nada.

A todas esas personas maravillosas que existen en mi vida y han hecho de mí una mejor persona.

Lissette Valdés

Quiero dedicar esta tesis a una persona que no se encuentra con nosotros pero estoy seguro que estaría muy orgulloso de mí, quiero darle gracias a la vida por darme el regalo de ser idéntico a él, esa persona es mi padre biológico.

A pesar de la prueba que me tocó vivir en la vida sin él, soy una persona afortunada y por eso especialmente esta tesis va dedicada:

A una persona que ha sido un padre para mí, un hombre incondicional, donde ha estado todo el tiempo y nunca me ha abandonado, facilitándome todo y que no me falte nada, por darme la oportunidad de disfrutar la vida sin sentirme que me faltaba algo, por apoyarme y confiar en mí en este sueño, esa persona se llama Robe y quiero decirle que lo quiero como si fuera mi padre realmente y para mi significa amor, respeto e inteligencia.

A tres mujeres muy importantes en mi vida que las identifico como madres:

A mi madre biológica por darme la oportunidad de vivir, por su dedicación, apoyo incondicional y fe en este sueño hecho realidad, que a pesar de que piensa que no soy tan apegado a ella, tenerla como madre es lo mejor que me ha pasado en la vida.

A mi tía Magdalena por estar todo el tiempo preocupándose por mi desempeño como hombre y profesional, por prepararme a diario y ser otra madre para mí, por hacerme

sentir que puedo ser capaz de lograr lo que sea, si uno se lo propone, por quererme y ser parte de mi vida.

A mi prima Maidel, mi mamita más joven, por haberme educado y enseñado a ser tan exigente conmigo mismo, por sus regaños y exigencias, ha sido una persona crucial en mi evolución como hombre y profesional, para mí significa dedicación, amor, esfuerzo, respeto y educación.

A mis hermanos Frank y Yuleisy por ser motores de inspiración en mi vida, por enseñarme que todo se puede y que no hay que tener miedo a nada, siempre tirar adelante que de todo se sale, ojalá esta tesis pueda demostrarles todo lo que me han enseñado en la vida.

A mi prima Lizdainet, por ser mi hermanita menor y permitirme ser su hermano, espero que esta tesis le sirva como inspiración en su carrera deportiva y que pueda lograr todo lo que ella se proponga.

A mi abuela Aida por estar controlándome todo estos años.

A mi novia Dayana por su comprensión, por estar siempre a mi lado hasta en los momentos más difíciles, te quiero y a pesar que el destino nos juntó en cuarto año, no me bastará una vida entera para demostrarle lo importante que ha sido para mí. Necesitaría dos vidas para demostrarle cuanto la quiero porque una realmente no me alcanza para expresar lo que siento por ella.

A mi amigo y hermano Yasmany, por ayudarme todos estos años, nunca olvidaré esta amistad que tenemos.

A mi amigo Marlon por apoyarme y ayudarme en todo lo que me ha hecho falta durante la tesis, por su ayuda incondicional, su capacidad y dotes, ojala siempre estés ahí.

Yander Morfa

Resumen

Actualmente el Centro de Identificación y Seguridad Digital está sentando las bases para llevar a cabo el proceso de migración hacia aplicaciones informáticas de Software Libre, siendo consecuente con la resolución No. 179/14 de la Universidad de las Ciencias Informáticas. Para cumplir con las políticas establecidas se propone la herramienta OpenERP, como alternativa para realizar la mayoría de los procesos empresariales. En el OpenERP se genera y gestiona una gran cantidad de documentos de los cuales es de gran importancia verificar su autenticidad, para así conocer con certeza, quién los genera o emite. El presente trabajo de diploma tiene como objetivo el análisis, diseño e implementación de un módulo que al ser incorporado al OpenERP, es capaz de realizar la firma digital a documentos en formato PDF. Para el desarrollo de la propuesta de solución se utilizó RSA como algoritmo asimétrico, SHA-1 como función resumen, como entorno de desarrollo integrado Eclipse y el lenguaje de programación Python, además de Visual Paradigm como herramienta de ingeniería de software asistida por computadoras, guiados por la metodología de desarrollo Programación Extrema. Con la realización del módulo se podrán firmar los documentos que se gestionan en la herramienta OpenERP, garantizando la integridad, el no repudio del origen y autenticidad de los mismos.

Palabras clave: autenticidad, documentos, firma digital, integridad, OpenERP.

Índice

Índice de Figuras

Índice de Tablas

Introducción

A medida que las personas necesitaban tener algún método o forma para identificarse como autores de una obra o propietarios de objetos, materiales e incluso lugares, surge la firma como un mecanismo para indicar que la información plasmada en cualquier documento, pertenece a su autor o propietario. De esta forma, la firma manuscrita acredita ante la sociedad la voluntad del firmante con respecto a un documento. El inconveniente de esta firma, es lo difícil que se hace probar su validez debido a lo complicado que es el procedimiento de verificarla. Con el uso de la firma, se dificulta demostrar que los documentos no han sido alterados o falsificados y se teme a que pierdan la integridad y autenticidad con que fueron creados.

En la actualidad ha sido necesario buscar nuevos métodos para realizar la firma. Con el desarrollo alcanzado por las Tecnologías de la Información y las Comunicaciones (TIC), han surgido nuevas formas para garantizar la autenticidad del origen de donde proviene la información y de quién la envía, brindando seguridad a la persona que la recibe de que la información no ha sido modificada por terceros. De esta forma, surge la “Firma Digital” como método para certificar la integridad de la información contenida en un documento y suplir la firma manuscrita utilizada tradicionalmente (Mehuron, 2000).

Las técnicas y normativas de la firma digital van cambiando conforme la tecnología avanza, teniendo cada día una mayor aceptación en los procesos financieros y comerciales principalmente. Cada una de estas técnicas debe ser usada y aprovechada según las necesidades, proporcionando diferentes niveles de seguridad, para lograr que las firmas sean generadas solamente por el firmante y no puedan ser falsificadas. Un ejemplo de técnica para lograr lo anterior, es la Infraestructura de Clave Pública (por sus siglas en inglés, PKI) utilizada para gestionar certificados digitales y permitir operaciones criptográficas como el cifrado y la firma digital.

Cuba, no se encuentra aislada de los adelantos en la tecnología, existen varios sectores que llevan a cabo todo un proceso de informatización como son la salud, educación, el sector jurídico, entre otros. Son cuantiosas las instituciones que deberían utilizar la firma digital como un mecanismo para aumentar la seguridad en la información que manejan.

La Universidad de las Ciencias Informáticas (UCI), es una de las instituciones creada con el objetivo de informatizar el país y contribuir al desarrollo económico del mismo. Actualmente esta universidad está llevando a cabo todo un proceso de migración hacia el Software Libre en las estaciones de trabajo de

sus centros de desarrollo (Nicado, 2014). El Centro de Identificación y Seguridad Digital (CISED) perteneciente a dicha universidad está sentando las bases para desarrollar esta tarea.

En el CISED se desarrollan numerosas aplicaciones que manipulan gran volumen de información referente a la identificación de personas, por tanto, es de vital importancia que estos documentos digitales estén completamente seguros. Para cumplir con las políticas establecidas en la Universidad, la herramienta que se propone como alternativa para la gestión de documentos es OpenERP.

El OpenERP es una herramienta libre que comprende la mayoría de los procesos empresariales, incluye la gestión de documentos, donde potencia una buena utilización de la información y permite a la vez una mejor localización y recuperación de los documentos electrónicos. Actualmente el módulo de gestión documental de OpenERP no utiliza ninguna técnica que proporcione seguridad a los documentos que se generan en él, por lo que se dificulta garantizar la autenticidad, el no repudio del origen e integridad en los documentos que gestionan los usuarios. Además se hace difícil probar que la información sea enviada o elaborada por quien dice ser.

Para realizar la firma digital a los documentos gestionados en el módulo de Gestión Documental de la herramienta OpenERP, hay que exportar estos fuera de la misma, para que así puedan ser firmados por una segunda herramienta que realice dicha funcionalidad. Lo anterior trae asociado varias desventajas como pueden ser la demora en la realización de los procesos y la más importante, graves violaciones que comprometen la seguridad de información clasificada a la hora de exportar los documentos fuera de la herramienta.

Por los elementos expuestos anteriormente surge el siguiente **problema de la investigación**: ¿Cómo garantizar la integridad, el no repudio del origen y la autenticidad en los documentos gestionados en la herramienta OpenERP?

La presente investigación tiene como **objeto de estudio** el proceso de firma digital a documentos, enmarcado en el **campo de acción** la firma digital de documentos en el módulo de gestión documental de la herramienta OpenERP.

Para dar solución al problema planteado anteriormente se propone como **objetivo general:** desarrollar un módulo en la herramienta OpenERP que realice la firma digital de documentos garantizando la integridad, el no repudio del origen y autenticidad de los documentos mediante el uso de tecnologías libres.

Este objetivo general se encuentra desglosado en los siguientes **objetivos específicos**:

- Construir el marco teórico referente a la firma digital y a sistemas que realizan este tipo de firma.
- Implementar para la plataforma OpenERP un módulo que permita la firma digital de documentos.
- Validar las funcionalidades del módulo propuesto para realizar la firma digital de los documentos almacenados en la plataforma OpenERP.

Se plantea como **idea a defender**: Desarrollando un módulo que realice la firma digital en documentos almacenados en la herramienta OpenERP, se podrá garantizar la integridad, el no repudio del origen y autenticidad de los documentos.

Para dar cumplimiento a los objetivos específicos se planificaron las siguientes **tareas de la investigación:**

- Realización de una búsqueda bibliográfica referente a las firmas digitales de documentos y al análisis de herramientas que realizan la firma digital para elaborar el marco teórico-conceptual de la investigación.
- Selección de las tecnologías, metodologías y herramientas necesarias para la implementación de la solución.
- Definición de los requisitos funcionales y no funcionales del módulo propuesto para la plataforma OpenERP.
- Realización de la documentación acorde a la metodología utilizada para desarrollar el módulo.
- Implementación de las funcionalidades necesarias para realizar la firma digital de documentos almacenados en la plataforma OpenERP.
- Realización de pruebas para comprobar las funcionalidades del módulo propuesto.

Para desarrollar la investigación se utilizan los siguientes métodos:

Métodos Teóricos

- **Analítico-Sintético**: Para analizar y comprender la teoría relacionada con la firma digital de documentos y la herramienta OpenERP particularmente el módulo de gestión documental, permitiendo tomar los elementos más significativos sobre estos temas.

- **Inducción-Deducción:** Para arribar a conclusiones luego del estudio de los temas relacionados con la problemática existente.

La investigación ha sido estructurada en tres capítulos distribuidos de la siguiente manera:

Capítulo 1. Fundamentos teóricos sobre la firma digital: Se realiza un estudio de los aspectos fundamentales sobre firma digital y el análisis de sistemas existentes que realizan este tipo de firma, así como los principales algoritmos que se utilizan en el proceso de firmar. Además se muestran las herramientas y tecnologías para desarrollar el módulo.

Capítulo 2. Análisis y diseño del módulo de OpenERP para firmas digitales de documentos: Se presenta la propuesta de solución que se quiere desarrollar. Se muestran los artefactos generados durante la fase de planificación, así como los requisitos funcionales y no funcionales que debe cumplir el módulo a implementar y la descripción de las arquitecturas a utilizar. Además se realiza el modelo de diseño a través de las tarjetas CRC (Clase-Responsabilidades-Colaboradores) para guiar la construcción de la solución propuesta.

Capítulo 3. Implementación y prueba del módulo de OpenERP para firmas digitales de documentos: Se desarrollan los artefactos propuestos para la fase de implementación y se muestra el conjunto de pruebas de calidad realizadas al módulo, con el propósito de verificar que las funcionalidades implementadas respondan al problema planteado.

Capítulo 1: Fundamentos teóricos sobre la firma digital

En el presente capítulo se muestran algunas definiciones necesarias para un mejor entendimiento de los conceptos relacionados a la investigación, así como un estudio de soluciones existentes que realizan la firma digital. Se define la metodología a utilizar, además de las herramientas y tecnologías seleccionadas para el proceso de desarrollo.

1.1 Conceptos asociados a la Firma Digital

1.1.1 Definición de Firma Digital

La firma digital es una herramienta tecnológica que permite garantizar la autoría e integridad de los documentos digitales, posibilitando que estos presenten una característica que únicamente era propia de los documentos en papel. Se trata de un conjunto de datos asociados a un mensaje digital que permite garantizar la identidad del firmante y la integridad del mensaje (Martínez Equihua, 2007).

Una firma digital se utiliza para autenticar información digital, como documentos, mensajes de correo electrónico o marcos, utilizando para ellos cifrado de la información haciendo uso de la función *hash* (Microsoft Office, 2010).

Una firma digital está destinada al mismo propósito que una manuscrita. Sin embargo, una firma manuscrita es sencilla de falsificar mientras que la digital es imposible mientras no se descubra la clave privada del firmante (Fernández Domínguez, 2006).

Los autores de la presente investigación definen como firma digital: un mecanismo que posee características técnicas y normativas para asegurar la identidad del firmante, integridad de los datos y el no repudio del origen.

1.1.2 Importancia y funcionamiento de la firma digital

Debido al desarrollo que ha alcanzado el mundo en la esfera tecnológica y el gran uso de Internet, la firma digital se convierte en un aspecto importante, permitiendo la autoría del firmante e integridad de los documentos que viajan en la red. Además esta firma, permite al usuario realizar cualquier operación de una forma más fácil, disminuyendo costos, recursos, tiempo e inconvenientes a la hora de utilizar el papel. Al mismo tiempo facilita el trabajo a partir de la utilización de un sistema confiable y cómodo desde cualquier lugar.

La firma digital funciona utilizando complejos algoritmos matemáticos que relacionan la información propia del firmante con el documento firmado, garantizando que se pueda reconocer la identidad del

autor además de certificar que su contenido no sea transformado. Esta firma está basada en criptografía asimétrica complementada con un certificado digital, otorgando una clave pública y otra privada a cada usuario.

Para firmar electrónicamente un documento, se utiliza una función *hash*, algoritmo que trasforma una secuencia de bits en otra menor y de igual tamaño, aplicándose tanto para la creación como para la verificación de la firma digital. De esta forma se obtiene un resumen específico para ese mensaje, el resumen obtenido se cifra con la clave privada del usuario obteniéndose de esta forma el mensaje con la firma digital. El receptor del mensaje sólo tiene que aplicar la clave pública para descifrar el mensaje (BA, 2013).

Para comprobar la integridad de un documento es necesario tener el documento original, se utiliza una función *hash* y se obtiene un resumen de este documento. Luego teniendo el documento firmado, se le aplica la clave pública del receptor para conseguir un resumen del mismo. Finalmente se verifican que ambos resúmenes sean iguales demostrando que los documentos no han sido modificados. A continuación se muestra en la figura 1 cómo ocurre el proceso de verificar una firma digital:

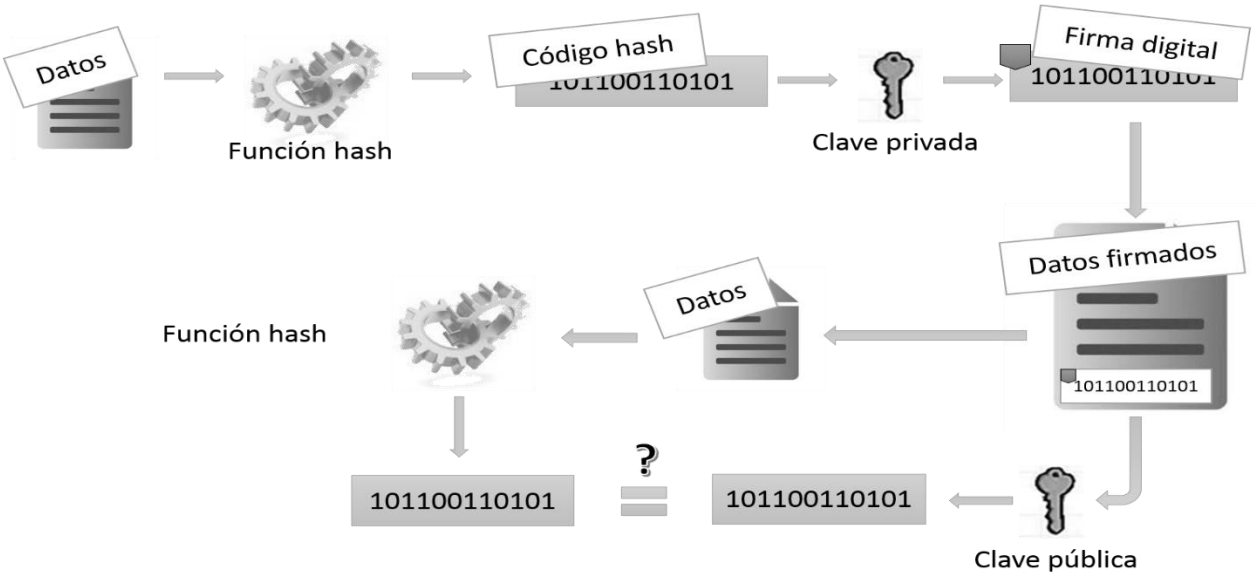


Fig. 1 Proceso para verificar la firma digital de un documento.

Según el Diccionario de la Real Academia, la palabra criptografía es el *"*Arte de escribir con clave secreta o de un modo enigmático*"*. Obviamente la criptografía hace tiempo ha dejado de ser un arte para convertirse en técnicas con el objetivo de lograr seguridad en la información.

La criptografía asimétrica o de clave pública consiste en que cada usuario posee dos claves, una privada y otra pública. La clave privada debe estar en secreto, mientras que la pública es conocida por todos. El sistema funciona de tal forma que la información cifrada con una de las claves puede ser descifrada solamente con la otra. De este modo si un usuario cifra una información determinada con su clave privada, cualquier persona que conozca su clave pública podrá descifrarla. Los criptosistemas de clave pública son los más adecuados para firmar digitalmente documentos, pues bastaría cifrarlo con la clave privada para obtener una firma digital segura, debido a que nadie excepto la persona a quien se le ha atribuido esa clave puede hacerlo. Luego, cualquier persona podría descifrarlo con su clave pública, demostrando su identidad (Días, y otros, 2012).

Actualmente existen varios algoritmos asimétricos para realizar la firma digital a un documento. A continuación se muestra una breve explicación de algunos de ellos:

- **Algoritmo Diffie-Hellman (DH):** Diffie-Hellman fue el punto de partida de los algoritmos asimétricos, basados en dos claves, una privada y otra pública. Este permite el intercambio secreto de claves entre dos entes que no han tenido un contacto directo, utilizando un canal inseguro y sin previa autenticación. Su seguridad radica en la dificultad de computarizar logaritmos discretos en un campo finito. Actualmente este algoritmo sólo es válido para el intercambio de claves simétricas, de esta forma es muy usado en los diferentes sistemas seguros implementados en Internet, como SSL (*Secure Socket Layer*, en español, Capa de Conexión Segura) y VPN (*Virtual Private Network*, en español, Red Privada Virtual). Este algoritmo no permite realizar firma digital a un documento. (López, 2004).

- **Algoritmo Rivest Shamir Adleman (RSA):** RSA es el algoritmos asimétrico más usado y también quizás el más sencillo de entender e implementar. Una peculiaridad de este algoritmo, es que sus dos claves sirven indistintamente tanto para cifrar como para autenticar. En cuanto a las longitudes de claves, el sistema RSA permite longitudes variables, siendo aconsejable actualmente el uso de claves de no menos de 1024 bits. RSA, se basa en la dificultad que presenta la factorización de números grandes. Las claves pública y privada se calculan a partir de un número que se obtiene como producto de dos primos grandes. Un atacante que quiera recuperar un texto claro a partir del criptograma y de la clave pública, tiene que enfrentarse a dicho problema de factorización o tendrá que resolver un logaritmo discreto (López, 2004).

- **Algoritmo ElGamal:** Este algoritmo está basado en Diffie-Hellman y fue creado primeramente para realizar firmas digitales pero luego se extendió para el cifrado de mensajes. Está basado en el problema de los logaritmos discretos que está relacionado con la factorización de números

enteros. Con este algoritmo se puede realizar también el cifrado eligiendo un número aleatorio y realizando el algoritmo extendido de Euler. Su seguridad reside en la utilización de logaritmos discretos en un campo finito (López, 2004).

- **Algoritmo de Rabin:** El sistema de clave asimétrica de Rabin es similar al RSA, aunque tiene un proceso de codificación muy distinto. Este algoritmo se basa en el problema de calcular raíces cuadradas módulo un número compuesto. Este problema se ha demostrado que es equivalente al de la factorización de dicho número. Funciona escogiendo dos números primos, siendo estos dos la clave privada y la clave pública es su producto (López, 2004).

Luego de hacer un estudio de varios algoritmos asimétricos, se determinó que el algoritmo RSA es el más adecuado para la implementación de la solución, debido a que el mismo es válido tanto para cifrar como para firmar digitalmente un documento, permitiendo además la detección de alteraciones y errores en la transmisión de los documentos y se basa en una función computacionalmente segura.

1.1.3 Funciones Hash

La función resumen o *hash* es un algoritmo que permite generar, a partir de un mensaje de entrada, resúmenes que representen de manera casi unívoca a este mensaje, aplicando operaciones aritméticas y lógicas, el cual posee un tamaño fijo, generalmente menor que el tamaño de entrada. En una función *hash* es necesario que exista un considerable cambio en el mensaje de salida y que sea imposible conocer el mensaje original a partir del mensaje resumen. El tamaño del resultado de la función resumen debe ser al menos de 128 bits para reducir la posibilidad de que dos mensajes posean el mismo valor resumen, además deben ser de poco costo tanto computacional como de uso de memoria. Una de las propiedades fundamentales de las funciones *hash* es que, si dos resultados de una misma función son distintas, entonces las entradas que generaron esos resultados también lo son. Dependiendo del tipo de función, estos pueden tener una longitud variable de 128,160, 224, 256, 384 o 512 bits (Estrada, 2011).

A continuación se describen varios tipos de algoritmos *hash* que se utilizan en el proceso de firma digital de un documento:

MD5: El MD5 (*Message-Digest Algorithm 5*, en español, Algoritmo de Resumen 5) es un algoritmo de reducción criptográfico, resultado de una serie de mejoras sobre el algoritmo MD4 (*Message-Digest Algorithm 4*, en español, Algoritmo de Resumen 4), procesa los mensajes de entrada en bloques de 512 bits, y produce una salida de 128 bits de longitud. Este algoritmo ha mostrado ciertas debilidades, por lo que ha disminuido su uso. Funciona utilizando un mensaje de n bits de longitud, este mensaje se alarga hasta que su longitud sea exactamente 64 bits inferior a un múltiplo de 512, de forma que va añadiendo

un uno seguido de tantos ceros como sea necesario. Luego se añaden 64 bits con el valor n, empezando por el byte menos significativo. De esta forma se obtiene el mensaje como un número entero de bloques de 512 bits. Este algoritmo emplea cuatro registros de 32 bits donde el byte menos significativo queda en la dirección de memoria más baja (López, 2004).

SHA-1: El algoritmo SHA-1 (*Secure Hash Algorithm 1,* en español Algoritmo de Resumen Seguro) se considera seguro pues es de una sola vía por lo que resulta difícil obtener el mensaje original a partir de su *hash* al utilizar fuerza bruta. SHA-1 genera una salida de 160 bits de longitud a partir de bloques de 512 bits del mensaje original. El algoritmo es similar a MD5, donde inicializa añadiendo al final del mensaje un uno seguido de tantos ceros como sea necesario hasta completar 448 bits en el último bloque, para luego aproximar la longitud en bits del propio mensaje, siendo el primer byte de la secuencia el más significativo. A diferencia de MD5, SHA-1 emplea cinco registros de 32 bits en lugar de cuatro. Su uso es recomendable para razones de interoperabilidad, aunque SHA-1 puede considerarse aceptable para ser utilizado como función *hash*. (López, 2004).

RIPEMD-160: El RIPEMD-160 (*RACE Integrity Primitives Evaluation Message Digest*, en español, Primitivas de Integridad del Resumen del Mensaje) es un algoritmo resumen diseñado para reemplazar a MD4 y MD5. Este algoritmo tiene una estructura muy similar a la de SHA-1. Se recomienda utilizar este algoritmo solo para compatibilidad con aplicaciones heredadas y de datos. Este toma como entrada un mensaje de longitud arbitraria produciendo como salida un resumen de mensaje de 160 bits, se representa típicamente como números en código hexadecimal de 40 dígitos y procesa los mensajes de entrada en bloques de 512 bits (Bosselaers, 2012).

SHA-2: Es un conjunto de funciones criptográficas de resúmenes (SHA-224, SHA-256, SHA-384, SHA-512). A continuación se muestra una descripción de estas variantes del algoritmo que presentan un tamaño de salida diferente (Mu, y otros, 2008):

- SHA-256: Debe ser elegido en la mayoría de los casos donde se desea una función de dispersión de alta velocidad. Se considera seguro, tiene un tamaño razonable de 32 bytes y 64 de rondas. SHA-256 se utiliza en un gran número de herramientas de seguridad y protocolos.
- SHA-224: Utiliza el mismo algoritmo que SHA-256. Fue creada porque su tamaño resumen tiene la misma longitud que dos claves Triple DES claves que pueden ser útiles.
- SHA-512: Este algoritmo es diferente con 64 números de bits y con 80 rondas. Su tamaño resumen es de 64 bytes, es muy grande y se considera algo excesivo para la mayoría de su uso.

- SHA-384: Este algoritmo es parecido al SHA-512, posee 64 números de bits y con 80 rondas. Genera una salida de 384 bits de longitud.

El algoritmo *hash* SHA-1 es más lento que MD5 pero más seguro y resistente a ataques por fuerza bruta, al tener su salida mayor longitud. El algoritmo RIPEMD-160 tiene una estructura semejante al de SHA-1 pero es recomendable emplearlo solo para compatibilidad con aplicaciones heredadas y de datos por lo que no es adecuado utilizarlo para desarrollar la solución. Por otra parte se encuentra el algoritmo SHA-2 el cual también es un algoritmo seguro pero resulta complicado su implementación. En todos los procesos de cifrado y *hash*, cuanto más seguros, más lento es su procesamiento y uso, por lo que se debe tener un equilibrio entre seguridad y velocidad. El SHA-2 es más seguro que el SHA-1, pero en este caso no es necesario tanta seguridad, pues estos algoritmos solo se utilizan para calcular la suma de un fichero y no hace falta un algoritmo tan lento y seguro como el SHA-2.

Por estas razones los autores de la investigación definen como algoritmo *hash* para implementar en la solución el SHA-1, además que el estándar a utilizar para firmar un documento es el PKCS7 (se explicará posteriormente) y este solo admite la función resumen SHA-1.

1.1.4 Infraestructura de Clave Pública

La PKI es definida por un grupo de trabajo en octubre de 1995 llamado PKIX *Working Group* como “el conjunto de hardware, software, personas, políticas y procedimientos necesarios para crear, gestionar, almacenar, distribuir y revocar certificados digitales basados en criptografía de clave pública”. Una PKI bien construida debe proporcionar autenticidad, integridad y no repudio de los documentos que se firmen digitalmente.

Componentes de una Infraestructura de Clave Pública

Los componentes necesarios en una PKI son los siguientes (España Boquera, 2003):

- La autoridad de certificación (CA) es la encargada de emitir y revocar certificados.
- La autoridad de registro (RA) es la responsable de verificar el enlace entre los certificados.
- Los repositorios de certificados son las estructuras encargadas de almacenar la información relativa a la PKI.
- Sistema de revocación de certificados y una lista de certificados revocados que permita a los usuarios comprobar si un certificado determinado todavía es válido o a ha sido revocado.
- Copias de seguridad de las claves, para recuperarlas en caso de pérdida u olvido.

- Soporte de no repudio de firmas digitales para que la clave empleada para crear firmas digitales se genere y almacene de forma segura bajo el control exclusivo del usuario propietario de la misma.
- Mecanismos de actualización automática de pares de claves y de certificados para cuando estos hayan caducado.

1.10.1 Certificados digitales

El Certificado Digital es un medio digital que permite garantizar técnica y legalmente la identidad de una persona. Se utilizan para realizar la firma digital de documentos, donde el receptor de un documento firmado puede tener la seguridad de que éste es el original y no ha sido manipulado y el autor de la firma digital no podrá negar su autoría (Valencia, 2012).

Los estados de un Certificado Digital pueden ser los siguientes:

- Válido: cuando está acreditado por una Autoridad Certificadora y está en uso.
- Revocado: cuando ha sido rechazado por el propio usuario o por una Autoridad Certificadora que lo emite.
- Caducado: cuando ha superado el tiempo de validez tanto para usuarios como para la Autoridad Certificadora.
- Suspendido: cuando se cancela la validez del certificado durante un tiempo determinado, pudiendo activarlo dentro del período de validez del certificado.

Para establecer el contenido y la estructura que debe tener un certificado digital se utiliza el estándar internacional ITU-T X.509, aunque su nombre completo es *"Internet X.509 Public Key Infrastructure Certificate and CRL* (Certificate Revocation List, en español, Lista de Revocación de Certificados) *Profile"*, definido por el *"Network Working Group"* y definido en la RFC 5280. De esta forma, los certificados pueden ser leídos o escritos por cualquier aplicación que cumpla con el mencionado estándar (Cutanda, 2014).

Estándar X.509:

El estándar X.509 se utiliza para construir una PKI. En este estándar se establecen parámetros para identificar al propietario del certificado, a la autoridad de certificación, las fechas de caducidad, entre otros. Además puede usarse para convertir los certificados a otros tipos, cambiar sus opciones de aceptabilidad o confianza, así como ver su información y firmar peticiones.

Del X.509 se han creado tres versiones, que en su medida fueron mejorando, en la versión 2 se añadieron dos campos, para identificar de forma única al emisor y usuario del certificado para manejar la posibilidad de reutilización de ambos en el tiempo, en su última versión se amplió, aún más, sus funcionalidades, este soporta la notación de extensiones: se puede definir una extensión personalizada e incluirla en el certificado, además de marcarla como crítica para indicar que debe ser validada de forma obligatoria y debe ser rechazado si no se cumplen las características indicadas (Moliner López, 2005). Todos los certificados X.509, sin importar su versión, contienen los campos mostrados en la figura 2.

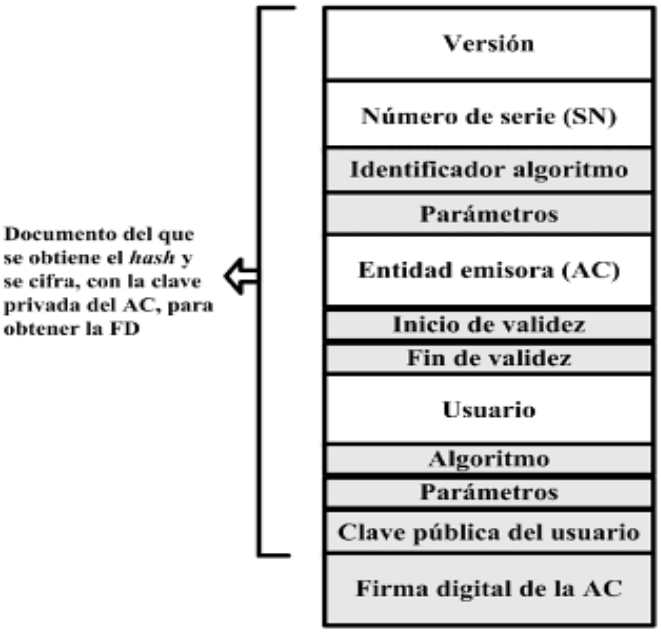


Fig. 2 Formato X.509 de un certificado digital.

Fuente: (Días, y otros, 2004).

La descripción de cada uno de los campos de un certificado digital X.509 es la siguiente (Días, y otros, 2004):

- Versión del formato del certificado, normalmente la versión X.509v3.
- Número de serie del certificado, SN. Es un identificador numérico único dentro del dominio de certificación de la CA (Autoridad Certificadora). Cuando se revoca un certificado, es este número el que aparece en la CRL, que contiene la lista de los certificados que no sirven para autenticar, al estar denegados como fiables para la autenticación.
- Algoritmo de firma y parámetros, que identifican el algoritmo asimétrico y la función de una sola vía usados para firmar el propio certificado.
- Emisor del certificado, que es el nombre de la CA, en formato X.500.
- Fechas de inicio y final de validez, que determinan el período de validez del certificado.

- El nombre del propietario de la clave pública que aparece firmada en el certificado, en formato X.500.
- La información de identificación del algoritmo utilizado, la clave pública del usuario y una serie de parámetros opcionales. Estos parámetros llamados extensiones del certificado permiten definir sus propios datos organizativos.
- La firma digital de la CA, es decir, el resultado de cifrar, mediante el algoritmo asimétrico y la clave privada del CA, el *hash* obtenido del documento X.509.

La versión 3 del estándar X.509 es la utilizada en el desarrollo de la aplicación ya que tiene incluida una serie de extensiones que permiten mejorar la información disponible. Entre la información añadida a esta nueva versión se encuentra:

- Identificación de la clave de la autoridad.
- Tributos de la clase.
- Políticas de certificación.
- Restricciones de uso de la clave.
- Alias del titular.
- Alias de la Empresa.
- Atributos del titular.
- Identidad de la autoridad certificadora.

Las extensiones de archivos de certificados del estándar X.509 son (Neira Vilalba, 2011):

- .CER - Certificado codificado en CER (*Canonical Encoding Rules*, en español, Reglas de Codificación Canónica).
- .DER - Certificado codificado en DER (*Distinguished Encoding Rules*, en español, Reglas de Codificación Distinguida).
- .PEM - Certificado codificado en Base64 (*Private Enhanced Mail*), puede contener certificados o claves privadas.
- .P7C/.P7B - Estructura PKCS#7 SignedData sin datos, solo certificado(s) o CRL(s).
- .P12/.PFX - PKCS#12, puede contener certificado(s) (público) y claves privadas (protegido con clave).

La extensión de archivo de certificado del estándar X.509 versión 3 utilizado en la solución es el formato .PEM ya que este permite generar y gestionar claves mediante el algoritmo RSA. Este formato es utilizado para autenticar un sitio web seguro y es compatible con las aplicaciones OpenSSL. Además

estos archivos de certificado se generan automáticamente y no pueden ser editados manualmente. Por último al utilizar un fichero con extensión .PEM se puede exportar el certificado sin la clave privada del usuario permitiendo distribuir mediante ese certificado su clave pública.

PKCS:

Los PKCS (Public Key Criptography Standards, en español, Estándares Criptográficos de Claves Públicas) son utilizados ampliamente en el mundo de las PKI. Existen muchos estándares de este tipo, a continuación se muestran algunos de ellos (Díaz Orueta, y otros, 2014):

PKCS # 1: Describe un método de cifrado y descifrado, mediante RSA, empleado en la construcción de las firmas digitales y de los formatos de petición descritos en el PKCS # 7.

PKCS # 3: Describe un método de implementación del algoritmo de Diffie-Hellman.

PKCS # 5: Describe un método para cifrar mensajes mediante una clave secreta derivada de una palabra de contraseña. Usa MD2 o MD5 para derivar la clave partir de la contraseña y usa DES para cifrar los mensajes. Está diseñado, principalmente, para cifrar claves privadas que viajan de un ordenador a otro, pero puede usarse igualmente para cifrar mensajes.

PKCS # 7: Define la sintaxis general de los datos que pueden estar cifrados o firmados. Tal sintaxis es recursiva, pudiéndose obtener mensajes dentro de mensajes o pudiéndose firmar datos que, previamente, estaban cifrados.

PKCS # 10: Describe las normas sintácticas de creación de las peticiones de certificados. Se pide un nombre del tomador del certificado, que pude utilizar algunos de los parámetros opcionales del certificado, una clave pública y, de manera opcional, otra serie de atributos. Todos estos datos deben ir firmados mediante la clave pública que se envía, siendo esta la información que usará la AC para validar la petición de certificado.

PKCS # 11: Especifica una interface de programación de aplicaciones criptográficas, la *Cryptographic Token* API, para dispositivos criptográficos de cualquier clase.

PKCS # 12: Describe la sintaxis para almacenar mediante software las claves públicas y privadas del participante, sus certificados y toda una serie de información criptográfica relacionada. Su objetivo es tratar de normalizar en un único fichero toda la información criptográfica que puedan necesitar diferentes aplicaciones.

El PKCS # 7 es el estándar utilizado en la realización del módulo debido a que es el más adecuado para firmar o cifrar documentos empleando algoritmos asimétricos. Además es compatible con el estándar X.509 y puede utilizar el formato de archivo .PEM, permitiendo convertirse en formatos de este tipo sin necesidad de realizar alguna operación criptográfica intermedia, y viceversa. Estas características hacen a los mensajes PKCS # 7 especialmente apropiado para la administración de claves basada en certificados digitales.

1.2 Herramientas de Firma Digital

En la actualidad se hace muy necesario brindar seguridad a la información que transita en la red ya que diariamente se realizan muchos negocios y trámites de forma electrónica, siendo importante la utilización de herramientas que permitan certificar la información dentro de la misma Web. A continuación se muestra un breve resumen de algunas de estas herramientas que firman documentos digitales. Luego de analizarlas se determinará si es conveniente integrar una de ellas al OpenERP o es más factible realizar un nuevo módulo que permita realizar esta funcionalidad.

XolidoSign: Con esta herramienta se pueden realizar firmas digitales en documentos con DNIe (Documento de Identidad Electrónico) o con un certificado digital, permitiendo verificar la autoría del archivo, que el documento no haya sido modificado y adjudica la autoría innegable del mismo. Es una herramienta gratuita disponible solo bajo la plataforma Windows, incluyendo las plataformas de 64bits. Entre sus características destaca (López Tallón, 2011):

- Soporta muchos tipos de certificado digital.
- Permite firmar cualquier documento o archivo sin límite de tamaño: PDF (*Portable Document Format*, en español, Formato de Documento Portátil), Excel, Word, Powerpoint, archivos txt, HTML (*HyperText Markup Language*, en español, lenguaje de marcas de hipertexto), PHP, bases de datos, entre otros.
- Soporta firma PDF integrada o externa. Permite hacer visible o invisible el campo de firma en el propio documento PDF.
- Soporta funcionalidades de portafirmas electrónico al poder firmar lotes de documento**s** a la vez.
- La interfaz de usuario es muy clara y fácil de utilizar.
- Comprueba la validez del certificado electrónico y el estado de revocación con la entidad emisora.

ESecure: Esta herramienta soporta tanto firma electrónica como el cifrado con contraseña o certificado para cualquier formato de fichero y varios algoritmos como RC2 (*Rivest Cipher* 4) y el AES (*Advanced Encryption Standard*, en español, Estándar de Encriptación Avanzada). La interfaz de usuario de ESecure está inspirada en el explorador de fichero de la plataforma Windows, soporta solamente esta

plataforma. ESecure dispone de varios modelos de licenciamiento, la licencia USER gratuita para ciudadanos**,** la licencia EDU establecida para la realización de un curso *online* gratuito y las versiones de pago PRO y PYME**.** Estas licencias se describen a continuación:

- *USER* soporta los formatos de firma PKCS#7***/***CMS (*Cryptographic Message Syntax*, en español, Sintaxis de Mensajes Criptográficos)***/***CAdES (CMS *Advanced Electronic Signatures,* en español, CMS Firmas Electrónicas Avanzadas).

- EDU soporta PDF/PAdES(PDF *Advanced Electronic Signatures*, en español, PDF Firmas Electrónicas Avanzadas).

- PRO y PYME permiten firmar en formato XMLDsig(Firma para Lenguaje de Marcado Extensible) ***/***XAdES (XML *Advanced Electronic Signatures*, en español, Firma electrónica avanzada para Lenguaje de Marcado Extensible) junto con funcionalidades adicionales como una funcionalidad de portafirmas electrónico o correo electrónico seguro S/MIME (*Secure* / *Multipurpose Internet Mail Extensions*, en español, Extensiones de Correo de Internet de Propósitos Múltiples / Seguro), en estas versiones solo está disponible un conjunto completo de funcionalidades. Permite importar certificados, soporta tanto ficheros PKCS#12, como librerías PKCS#11(*Cryptographic Token Interface***,** en español, Interfaz de Dispositivo Criptográfico) para el uso de tarjetas criptográficas como el DNIe. Además limita la validación de los certificados vía OCSP (*Online Certificate Status Protocol*, en español, Protocolo de estado de certificados en línea) (López Tallón, 2011).

CEGELFIR: Es una herramienta creada en la UCI específicamente en el Centro de Gobierno Electrónico (CEGEL) que permite realizar la firma digital de los documentos generados por los sistemas desarrollados en dicho centro, utilizando tarjetas inteligentes, *token usb* y certificados contenidos en dispositivos de almacenamiento. Algunas de sus funcionalidades son firmar PDF, permitir que un documento pueda ser firmado por más de un usuario y firmar varios documentos de forma simultánea. El sistema es multiplataforma (Linux/Windows) y está implementada en java (Rodríguez Fernández, 2012).

Sinadura: Es una aplicación de escritorio multiplataforma que permite firmar y comprobar la firma en archivos PDF. Sinadura es software libre que soporta los formatos PKCS11 utilizando tarjetas inteligentes(Izempe, DNIe), el estándar PKCS12, XAdES-X-L, XAdES-T y XAdES-BES. Se puede incluir el sello visible de la firma, soporta el idioma español e inglés, firma múltiples ficheros a la vez. Además permite firmar y validar por interfaz gráfica y por línea de comandos. Presenta un visor de información

de las firmas y un almacén de certificados de confianza para el módulo de validación permitiendo la validación de firmas PKCS7 (Asociación de Empresas de Software Libre , 2011).

Al estudiar estas herramientas descritas se observó que todas realizan y verifican la firma digital de documentos, pero las mismas no son adecuadas para ser integradas al OpenERP. En el caso de las aplicaciones XolidoSign y ESecure a pesar de ser herramientas a las que se pueden acceder de forma gratuita, solo trabajan bajo plataforma Windows, inconveniente que ya impide su integración pues se desea que el resultado del trabajo sea multiplataforma. La herramienta CEGELFIR tampoco puede ser utilizada debido a que está desarrollada en java y su integración al OpenERP podría traer errores de compatibilidad, además realizar este proceso de integración implicaría mayor esfuerzo y tiempo de los desarrolladores. Por otra parte, Sinadura es una aplicación de escritorio y su integración en el OpenERP podría ser engorroso, debido a que los procesos *online* y los procesos locales no son fáciles de intercomunicar si no se crearon desde su inicio con ese fin.

Al no encontrar una herramienta que se ajuste a los requisitos necesarios para trabajar integrada al OpenERP, se puede concluir que es necesario implementar un módulo que realice la firma digital. Si se incluye la firma digital dentro de la herramienta OpenERP, varias personas podrán aprobar de forma legal un documento sin que este viaje físicamente, sino moviéndose a través del flujo de trabajo que está dentro de la propia herramienta.

1.3 Metodología de desarrollo

Para asegurar el éxito de un software, es necesario utilizar una metodología que permita guiar el proceso de desarrollo de dicho software para obtener un producto que cumpla con sus especificaciones, sea elaborado en el tiempo previsto, con un costo adecuado y la calidad esperada por el cliente. Actualmente no existe una metodología global que se pueda aplicar a cualquier proyecto y su selección debe ajustarse a las necesidades específicas y características de cada proyecto.

Las metodologías se clasifican en ágiles y pesadas. Las metodologías pesadas han demostrado ser efectivas y necesarias en proyectos de gran tamaño con respecto a tiempo y recursos. Sin embargo, este enfoque no resulta ser el más adecuado para muchos de los proyectos actuales donde el entorno del sistema es muy cambiante, y se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad. Ante las dificultades para utilizar metodologías pesadas, muchos equipos de desarrollo prefieren utilizar metodologías ágiles por estar especialmente orientadas para proyectos pequeños, además constituyen una solución a medida para ese entorno, aportando una elevada simplificación que a pesar de ello no renuncia a las prácticas esenciales para asegurar la calidad del

producto (Calderón, y otros, 2007). Por esta razón se realizó un estudio de algunas metodologías ágiles permitiendo elegir la más conveniente para el desarrollo del módulo. A continuación se hace un breve resumen de las mismas.

1.3.1 XP

La Programación Extrema (XP) es una metodología basada en el intercambio continuo entre el equipo de desarrollo y el cliente pues es uno de los requisitos para llegar al éxito del proyecto, permite una comunicación fluida entre los desarrolladores, promoviendo el trabajo en equipo, además de simplicidad al desarrollar y codificar las soluciones. XP permite una programación organizada y es utilizada principalmente para proyectos de corto plazo y equipos pequeños donde los requerimientos cambian rápidamente. La metodología XP define Historias de Usuario para especificar los requisitos del software, estas historias las escribe el cliente y describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales (Canós, y otros, 2007).

Características:

- Desarrollo iterativo e incremental: Se realizan pequeñas mejoras, unas tras otras.
- Pruebas unitarias: frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión.
- Corrección de errores: Se realiza antes de añadir una nueva funcionalidad, además de hacerse entregas frecuentes.
- Refactorización del código: Se puede reescribir ciertas partes del código para aumentar su legibilidad y mantenimiento pero sin modificar su comportamiento.
- Propiedad del código compartida: Todos los desarrolladores pueden corregir y extender cualquier parte del proyecto.

1.3.2 FDD

El Desarrollo Manejado por Funcionalidades; como otras metodologías adaptables, realiza iteraciones cortas que producen un software funcional donde el cliente y los desarrolladores pueden ver y monitorear.

Esta metodología está centrada en el usuario, no en el programador; su objetivo es sintetizar un programa conforme a las funcionalidades requeridas. A diferencia de otras metodologías ágiles no cubre todo el ciclo de vida sino sólo las fases de diseño y construcción. Se considera adecuado para proyectos cortos y equipos más pequeños (Calderón, y otros, 2007).

Características:

- Se preocupa por la calidad, por lo que incluye un monitoreo constante del proyecto.
- Ayuda a contrarrestar situaciones como el exceso en el presupuesto, fallas en el programa o el hecho de entregar menos de lo deseado.
- Propone tener etapas de cierre cada dos semanas. Se obtienen resultados periódicos y tangibles.
- Define claramente entregas tangibles y formas de evaluación del progreso del proyecto.
- No hace énfasis en la obtención de los requerimientos sino en cómo se realizan las fases de diseño y construcción.
- La falta de documentación hacen difícil que pueda reutilizarse el código fuente.

1.3.3 Scrum

Scrum es una metodología que define un proceso empírico para asegurar que el conocimiento procede de la experiencia y de tomar decisiones basándose en lo que se conoce. Emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control del riesgo. El equipo de Scrum trata de elegir la mejor forma de llevar a cabo su trabajo y no son dirigidos por personas externas al equipo. Al inicio del proyecto se define la lista del producto durante reuniones de planeamiento con los clientes, estos son los requisitos funcionales y no funcionales que debe cumplir el sistema. Los mismos estarán especificados según las convenciones de la organización ya sea mediante: features, casos de uso, diagramas de flujo de datos, tareas, entre otros. Luego se definen las iteraciones, conocidas como sprint, donde cada una tendrá su propia lista de iteraciones con los requisitos a ser construidos en el sprint correspondiente. La duración recomendada de las iteraciones es de un mes (Schwaber, y otros, 2013).

1.3.4 Metodología de desarrollo seleccionada

Luego del estudio realizado sobre las características principales de estas metodologías de desarrollo y comparando cada una de ellas, se determina que la más indicada a utilizar es XP. Esta metodología es la más conveniente en la solución a implementar debido a que el tamaño del equipo de desarrollo es pequeño, se puede definir el estado del proyecto pues al realizar versiones funcionales de la solución a corto plazo es bastante sencillo hacer un seguimiento de su avance. Además se realiza solamente lo que el cliente desea y de la forma más sencilla, desde un diseño simple hasta la refactorización del código.

1.4 Ambiente de desarrollo

A raíz de un estudio de herramientas y tecnologías en las diferentes fuentes bibliográficas, se seleccionan algunas de ellas por ser las más apropiadas para el desarrollo de la solución. Además teniendo en cuenta que en Cuba es de vital importancia utilizar aplicaciones libres para impulsar el

desarrollo informático, orientado a alcanzar la seguridad, invulnerabilidad e independencia tecnológica. A continuación se muestran las descripciones de las herramientas y tecnologías elegidas.

1.4.1 OpenERP 7.0

Es una herramienta bajo licencia libre utilizada para la gestión empresarial que automatiza los procesos de negocio. OpenERP soporta varias plataformas como Windows, Linux, Unix y MacOS con posibilidad de buen uso mediante interfaz web o aplicación de escritorio. Añade herramientas de análisis y generación de informes, simplificándose la gestión y visualización de la información. Incorpora la gestión de documentos y permite la generación de impresos vía PDF, HTML, y permite exportar datos a otros programas como OpenOffice o Microsoft Office (Excel, Word).

La tecnología de OpenERP cuenta con un potente CRM (*Customer Relationship Management*, en español, Software para la Administración de la Relación con los Clientes) totalmente integrado en el sistema, esto significa poder disponer con facilidad de cualquier información relacionada con el cliente. La arquitectura del sistema es cliente/servidor, lo que permite que todos los usuarios trabajen sobre el mismo repositorio de datos, esto tiene la ventaja de que toda la información esté disponible y sincronizada en todo momento. Esta herramienta está desarrollado íntegramente en Python que se apoya sobre el gestor de base de datos PostgreSQL (Izquierdo, 2009).

1.4.2 IDE Eclipse 4.3.8

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores.

El entorno de desarrollo integrado (IDE) de Eclipse emplea módulos para proporcionar toda su funcionalidad al frente de la plataforma de cliente enriquecido, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software (Eclipse, 2012).

Adicionalmente le permite a Eclipse extenderse usando otros lenguajes de programación como son C/C++ y Python.

La arquitectura *plugin* permite escribir cualquier extensión deseada en el ambiente, como sería Gestión de la configuración. Por las facilidades que brinda para el depurado, porque dispone de un editor de texto con resaltado de sintaxis, así como por lo amigable que es su entorno visual, porque permite añadir el control de versiones con Subversión (Galli, 2005).

1.4.3 Python 2.7

Se utiliza el lenguaje de programación Python, ya que el OpenERP está implementado en este lenguaje y es la herramienta sobre la cual se trabaja para la implementación de la solución. Python es un lenguaje donde su sintaxis es sencilla y tiene incorporada características como (Knowlton, 2009):

- Es sencillo aprender ya que ofrece una sintaxis extraordinariamente simple.
- Es un software de código abierto.
- Es un lenguaje de alto nivel.
- Se puede usar en las plataformas Linux, Windows, Macintosh, Solaris, entre otras.
- Es un lenguaje interpretado que incluye un modo interactivo en el cual se escriben las instrucciones en una especie de intérprete de comandos.
- Es orientado a objetos, el programa es construido sobre objetos los cuales combinan datos y funcionalidad.
- Presenta extensas librerías que ayudan a realizar muchas tareas comunes sin necesidad de tener que programarlas desde cero.

1.4.4 PostgreSQL 8.4

Es un sistema gestor de base de datos relacional orientado a objetos, multiplataforma y con su código fuente disponible libremente. PostgreSQL mediante un sistema de Acceso Concurrente Multiversión (MVCC) permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Incluye además herencia entre tablas y tiene la capacidad de comprobar la integridad referencial, así como la de guardar procedimientos en la propia base de datos (PostgreSQL-es, 2010).

La selección de PostgreSQL como sistema gestor de base de datos estuvo determinada por el hecho de que es también el sistema gestor presente en el OpenERP, de ahí la importancia de que exista compatibilidad con la solución a desarrollar.

1.4.5 UML 2.0

UML (*Unified Modeling Language*, en español, Lenguaje Unificado de Modelado) es un lenguaje gráfico para definir, documentar, construir y detallar los artefactos de un sistema. Este lenguaje permite dar soporte a una metodología de desarrollo de software, pero no especifica cual metodología o proceso utilizar. UML cuenta con un conjunto de diagramas que muestran los aspectos de cada entidad representada, además permite modelar sistemas orientados a objetos (Sierra, 2006).

La utilización de UML está dada por las bondades que brinda a los desarrolladores, al proveer un vocabulario y reglas para permitir una comunicación estándar y robusta aumentando las posibilidades de obtener un producto exitoso.

1.4.6 Visual Paradigm for UML 8.0

Es una herramienta CASE (*Computer Aided Software Engineering*, en español, Ingeniería de Software Asistida por Computadora) que utiliza UML como lenguaje de modelado y está basado en un enfoque Orientado a Objetos, soporta el ciclo de vida completo del proceso de desarrollo de software a través de la representación de todo tipo de diagramas. Esta herramienta es de gran ayuda para la creación de un software pues permite analizar, diseñar y hasta incluso generar documentación y código fuente desde los diagramas. Se caracteriza además por estar disponible en múltiples plataformas, su diseño está centrado en casos de uso y enfocado al negocio para obtener un software de mayor calidad. Usa un lenguaje estándar común para cualquier equipo de desarrollo que facilita la comunicación (Visual_Paradigm, 2012).

1.5 Conclusiones Parciales

El estudio realizado sobre las herramientas que realizan la firma digital evidenció que ninguna soluciona el problema científico expuesto a lo largo de la investigación. Las condiciones de trabajo sobre la que está respaldada la presente investigación posibilitó el uso de la metodología XP para guiar el proceso de desarrollo de software. Se decide utilizar RSA como algoritmo asimétrico, SHA-1 como algoritmo para calcular el resumen *hash* de los documentos, Eclipse como entorno de desarrollo integrado y el lenguaje de programación Python, además de Visual Paradigm como herramienta CASE.

Capítulo 2: Análisis y diseño del módulo de OpenERP para firmas digitales de documentos.

En el presente capítulo se exponen las principales características y cualidades del sistema a implementar, mediante la identificación de los requisitos funcionales y no funcionales con los que debe cumplir la aplicación que dará solución a la problemática de la presente investigación. También se describen los conceptos relacionados con las definiciones de patrones de diseño de software empleados. Además se detallarán los pasos de la metodología que se propone en el capítulo anterior (XP), incluyendo la elaboración de las tarjetas CRC como artefactos fundamentales de la metodología. Por último se muestra el diagrama de clases del diseño para un mejor entendimiento de la solución a realizar.

2.1 Propuesta de solución

La herramienta OpenERP cuenta con un módulo de gestión documental donde potencia una buena utilización de la información y almacena sus documentos electrónicos en el gestor de base de datos PostgreSQL. Actualmente este módulo no garantiza la autenticidad, el no repudio del origen e integridad en los documentos que se gestionan por parte de los usuarios.

La firma digital es una de las alternativas para lograr la integridad en los documentos digitales y garantizar la identidad del firmante. Luego de realizar un estudio sobre la misma se pretende desarrollar un módulo que permita realizar la firma digital a documentos almacenados en la herramienta OpenERP.

Como propuesta de solución a la problemática existente en la herramienta OpenERP se pretende extender el módulo de gestión documental que tiene dicha herramienta, heredando todas las funcionalidades implementadas anteriormente. Para crear un módulo se necesita crear una carpeta con el nombre de este en el directorio donde se encuentran los módulos desarrollados de OpenERP, dentro de esta carpeta se aloja la lógica del negocio y está compuesto por cuatro archivos básicos. El primer archivo init.py permite cargar el módulo creado, el segundo openerp.py contiene un diccionario que describe todos los archivos que se utilizan en la implementación del módulo, el tercero refleja el nombre del módulo.py (mydocument.py) el cual define los objetos que componen un módulo en la vista y en la base de datos siendo la clase controladora de este módulo y por último un archivo que refleja el nombre del módulo.xml (mydocument.xml) que muestra las vistas, para describir cómo y dónde es dibujado cada campo de los objetos, finalmente añadiéndole la firma digital.

A través de la clase controladora el código es interpretado en XML (*eXtensible Markup Language*, en español, Lenguaje de Marcado Extensible) para obtener los datos y presentarlos al usuario mediante un

navegador web para que pueda firmar los documentos. El módulo permite firmar un documento utilizando un certificado digital autogenerado, es decir gestionado por la aplicación, debido a que no se tenía un certificado propio de alguna entidad certificadora, o el usuario puede cargar su certificado .p12 con la contraseña, el cual debe estar guardado en un dispositivo de almacenamiento.

Para generar la firma digital de un documento, se le aplica el algoritmo *hash* sha-1 al documento para obtener un resumen del mismo, luego se cifra este resumen con la clave privada mediante el algoritmo RSA y de esta forma se obtiene un documento firmado digitalmente. En la figura 3 se muestra la propuesta de solución para una mejor comprensión de la misma:

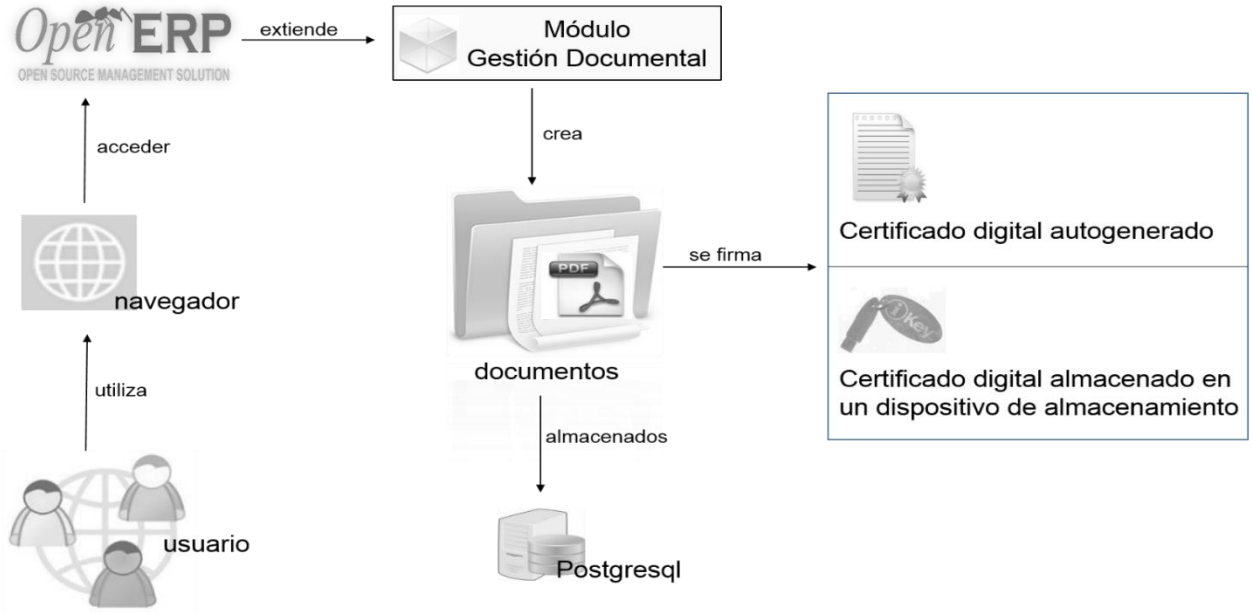


Fig. 3 Propuesta de solución.

2.2 Modelo de dominio

A pesar de que el modelo de dominio no es uno de los artefactos que se generan en la metodología XP, se decide realizar el mismo para una mejor comprensión del negocio. Un modelo de dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés que pueden mostrar asociaciones entre las clases conceptuales y sus atributos. Utilizando la notación UML, un modelo de dominio se representa con un conjunto de diagramas de clases en los que no se define ninguna operación (Larman, 2004).

El modelo de dominio de la solución propuesta se muestra en la figura 4:

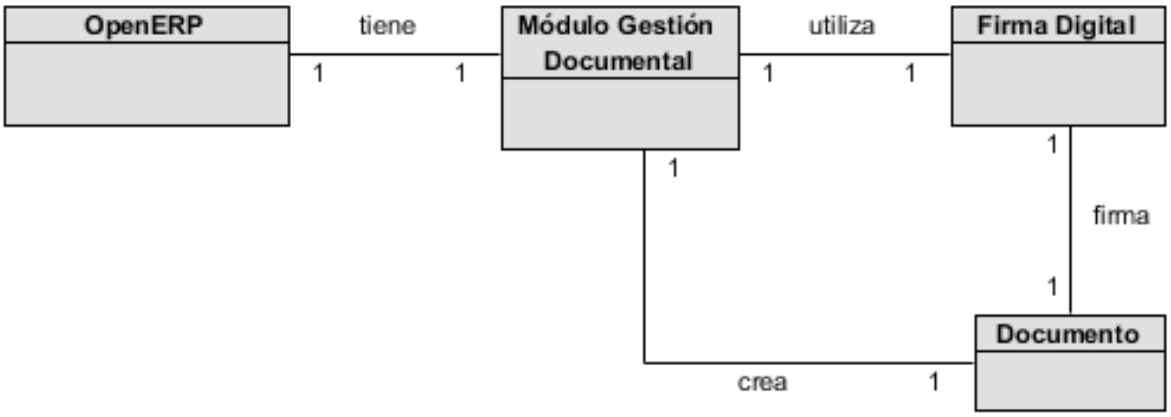


Fig. 4 Modelo de dominio.

OpenERP: Herramienta utilizada para la gestión empresarial.

Módulo Gestión Documental: Módulo que permite gestionar documentos.

Firma Digital: Mecanismo para lograr la integridad, no repudio del origen y autenticidad de un documento.

Documento: Elemento que contiene una información determinada.

2.3 Requisitos de Software

El IEEE *Standard Glossary of Software Engineering Terminology* (IEEE 610.12 1990) define un requisito como la "condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente". Es de gran importancia una buena identificación de los requisitos como parte del proceso de desarrollo de software pues contribuye considerablemente a la calidad del producto final repercutiendo en las demás fases de desarrollo del mismo. Además permite tomar mejores decisiones de diseño y de arquitectura garantizando un mejor soporte a la gestión de cambios, para reducir los riesgos y los problemas de mantenimiento.

2.3.1 Requisitos funcionales

Los requisitos funcionales "son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares." (Sommerville, 2005). La especificación de los requisitos funcionales describe lo que el sistema debe hacer, además de describir con detalle su función, entradas y salidas.

A continuación se muestran los requisitos funcionales definidos en esta investigación:

RF1. Firmar documento PDF.

RF1.1. Generar nuevo PDF.

RF1.2 Modificar esquema XML del PDF.

RF1.3 Calcular función resumen.

RF2. Generar certificado de estándar X.509.

RF3. Cargar un certificado digital.

2.3.2 Requisitos no funcionales

Según Sommerville los requisitos no funcionales son “restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares.” (Sommerville, 2005). Existen diferentes tipos de requisitos no funcionales, a continuación se muestran los identificados en el sistema a desarrollar:

Software:

- Sistema Operativo Windows XP o superior.
- Sistema Operativo Ubuntu 10.1 o superior.
- Los navegadores web Mozilla Firefox o Chrome.

Hardware:

- Memoria RAM de 1Gb.
- Espacio de disco duro de 4Gb para la base de datos.

Diseño e implementación:

- La herramienta OpenERP 7.0.
- Lenguaje de programación Python 2.7.
- Gestor de base de datos PostgreSQL 8.3.

Apariencia o interfaz externa:

- La interfaz debe ser sencilla de manera que permita la fácil interacción del usuario con el sistema.

Seguridad:

- Confiabilidad: La información manejada por el sistema debe estar protegida de acceso no autorizado.

- Integridad: La información manejada por el sistema debe estar protegida contra la corrupción y estados de inconsistencia.

- Disponibilidad: La información manejada por el sistema debe estar accesible para los usuarios y los mecanismos que se utilicen para lograr la seguridad no deben ocultar o retrasar la obtención de los datos deseados en un momento dado.

2.4 Historias de usuario

En la primera fase de la metodología XP se realizan las historias de usuario donde se describen los requisitos del software a desarrollar. Se tratan de una serie de tarjetas donde el cliente hace una breve descripción de las características que el sistema debe poseer, sin emplear un lenguaje técnico, realizándose una por cada funcionalidad principal del sistema. Cada historia de usuario debe estar descrita claramente para que los programadores puedan hacerla e implementarla en poco tiempo (Toro, 2013). Las historias de usuario realizadas en la presente investigación se muestran a continuación:

Tabla 1: HU Generar nuevo PDF.

Historia de Usuario	
Número: 1	**Nombre:** Generar nuevo PDF.
Cantidad de modificaciones: 1	
Iteración Asignada: 1	
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Medio	**Puntos Reales:** 2
Descripción: El sistema debe permitir tomar el PDF original y guardar su contenido en un documento nuevo haciéndolo página por página. Además adiciona en el documento nuevo un cuño como prueba de que ha sido firmado.	
Prototipo:	

Tabla 2: HU Modificar esquema XML del PDF.

Historia de Usuario	
Número: 2	**Nombre:** Modificar esquema XML del PDF.
Cantidad de modificaciones: 0	
Iteración Asignada: 1	
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1
Descripción: El sistema debe permitir agregar los diccionarios de formularios interactivos que es donde se guarda toda la información de firma. Además se agregan los campos de firma en los documentos PDF.	
Prototipo:	

Tabla 3: HU Calcular función resumen.

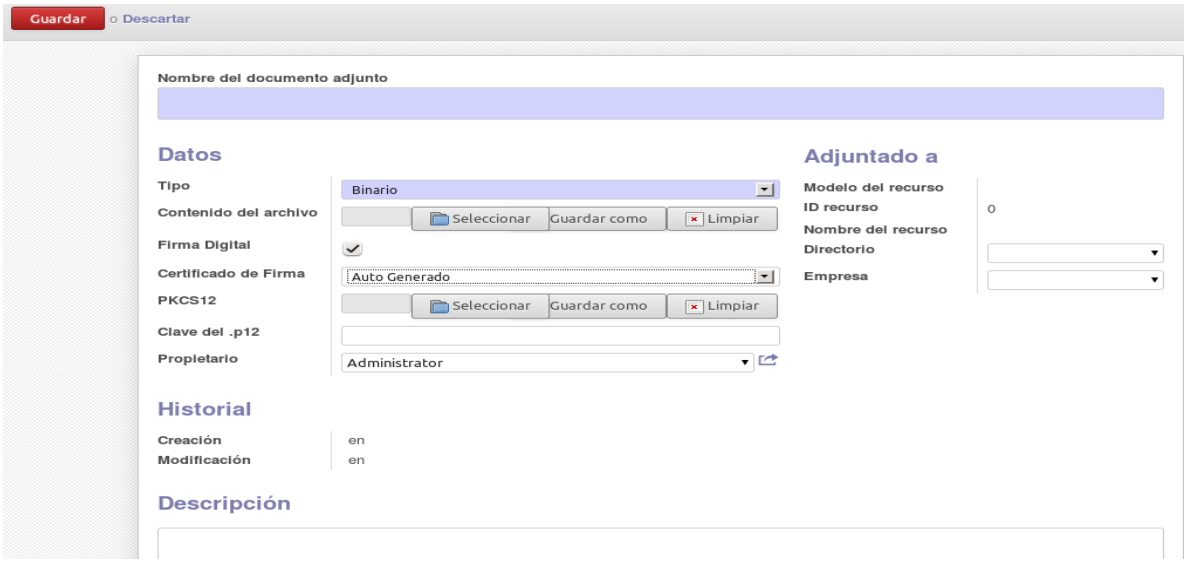
Historia de Usuario	
Número: 3	**Nombre:** Calcular función resumen.
Cantidad de modificaciones: 0	
Iteración Asignada: 1	
Prioridad en Negocio: Muy Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1
Descripción: El sistema debe leer un documento, calcular la función resumen a través del algoritmo sha-1, para obtener un resumen de ese documento.	
Prototipo:	

Tabla 4: HU Generar certificado de estándar X.509.

Historia de Usuario	
Número: 4	**Nombre:** Generar certificado de estándar X.509.
Cantidad de modificaciones: 1	
Iteración Asignada: 2	

Prioridad en Negocio: Muy Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Medio	**Puntos Reales:** 2
Descripción: El sistema debe permitir generar un certificado X.509, donde guardará la información de las claves del algoritmo de cifrado RSA en un fichero .PEM, además de calcular la tabla *hash* del estándar X.509.	
Prototipo:	

Tabla 5: HU Firmar documento PDF.

	Historia de Usuario
Número: 5	**Nombre:** Firmar documento PDF.
Cantidad de modificaciones: 2	
Iteración Asignada: 3	
Prioridad en Negocio: Muy Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Muy Alto	**Puntos Reales:** 2
Descripción: El sistema debe permitir al usuario firmar digitalmente un documento en formato PDF utilizando un certificado digital autogenerado o un .P12 guardado en un dispositivo de almacenamiento.	
Prototipo: 	

2.5 Esfuerzos requeridos por historias de usuario

En la fase de planificación es necesario que los programadores estimen cuánto esfuerzo requiere cada historia de usuario para a partir de esto definir el cronograma y programar de acuerdo a esta estimación. Cada una de las historias deben programarse de una a tres semanas de desarrollo, si las estimaciones son mayores a este tiempo son indicios de que la historia de usuario es muy compleja y debe ser dividida en diferentes historias de usuario (Beck, 2002). Los resultados estimados en esta investigación son los siguientes:

Tabla 6: Estimación de esfuerzo por historias de usuario.

Historia de Usuario	Estimación por semanas
Firmar documento PDF.	2
Generar nuevo PDF.	1
Modificar esquema XML del PDF	2
Calcular función resumen.	1
Generar certificados de estándar X.509.	3
Cargar certificado digital	2

2.6 Plan de iteraciones

Luego de ser definidas las historias de usuario y señalar las de mayor importancia por su dificultad o por su punto crítico es necesario establecer las iteraciones con las que se contarán. Estas iteraciones deben ser relativamente cortas pues mientras más entregas se hagan, más retroalimentación se obtendrá y esto va a representar una mayor calidad del sistema a largo plazo (Beck, 2002). La implementación del sistema a desarrollar tendrá una duración de 11 semanas, donde se realizarán tres iteraciones. La tabla que se muestra a continuación presenta las historias de usuario organizadas por iteraciones:

Tabla 7: Plan de iteraciones.

Iteraciones	Orden de las Historia de Usuario a implementar	Duración total
1	Generar nuevo PDF.	1

1	Modificar esquema XML del PDF	2
1	Calcular función resumen.	1
2	Generar certificado de estándar X.509.	3
2	Cargar certificado digital.	2
3	Firmar documento PDF.	2

2.7 Plan de entrega

El plan de entregas es una estrategia que vincula a los programadores y los clientes, siendo estos últimos los que tienen la obligación de manejar cuando debe ser entregado el producto y los programadores de cumplir ese tiempo de desarrollo. Deben asegurarse que el resultado final tenga el mayor valor de negocio posible con cada iteración, estableciendo la prioridad de cada historia de usuario, apostando por resolver las de más valor y riesgo primeramente (Joskowicz, 2008). A continuación se muestra el plan de entrega:

Tabla 8: Plan de entrega.

Iteraciones	**Primera entrega (27-3-2014)**	**Segunda entrega (01-5-2014)**	**Entrega final (15-5-2014)**
1	X	X	X
2		X	X
3			X

2.8 Tarjetas CRC

El uso de las tarjetas CRC permite al equipo de desarrollo realizar el diseño del software, representando en cada tarjeta una clase orientada a objetos, definiendo las funciones que debe implementar y las clases con las que debe trabajar para cumplir con sus responsabilidades. Con estas tarjetas se pretende facilitar el análisis y discusión entre los integrantes del equipo, donde permita verificar las especificaciones del sistema mediante un diseño lo más sencillo posible (Wells, 2013). Las tarjetas que se muestran en las tablas siguientes fueron las identificadas para guiar la implementación de las historias de usuario:

Tabla 9: Tarjeta CRC para la clase pdfSigner.

Clase: pdfSigner	
Responsabilidades	**Colaboradores**
Realiza la firma de documentos PDF, calculando la función resumen y cifrando este resumen con la clave privada.	cms pyPdf haslib os datetime

Tabla 10: Tarjeta CRC para la clase cms.

Clase: cms	
Responsabilidades	**Colaboradores**
Realiza la sintaxis de mensajes criptográficos. Inserta la estructura PKCS #7.	x509Generator x509Parser pkcs7 pyasn1 tlslite math hashlib datetime

Tabla 11: Tarjeta CRC para la clase pkcs7.

Clase: pkcs7	
Responsabilidades	**Colaboradores**
Define la estructura del estándar pkcs7 para firmar un documento.	x509 pyasn1

Tabla 12: Tarjeta CRC para la clase x509.

Clase: x509	
Responsabilidades	**Colaboradores**
Define la estructura del estándar de certificado digital X.509 versión 3.	pyasn1

Tabla 13: Tarjeta CRC para la clase x509Generator.

Clase: x509Generator	
Responsabilidades	**Colaboradores**
Genera un certificado X.509.	x509 pyasn1 tlslite OpenSSL Hashlib x509Parser datetime

Tabla 14: Tarjeta CRC para la clase x509Parser.

Clase: x509Parser	
Responsabilidades	**Colaboradores**
Transforma la estructura del certificado X.509 convirtiéndolo en un archivo de tipo .PEM. Devuelve la información contenida en ese archivo.	x509

Tabla 15: Tarjeta CRC para la clase tests.

Clase: tests	
Responsabilidades	**Colaboradores**
Comprueba el proceso de firma, cifrado de documentos y generación de un certificado X.509.	x509Generator cms pdfSigner tlslite

Tabla 16: Tarjeta CRC para la clase myDocument.

Clase: myDocument	
Responsabilidades	**Colaboradores**
Define los objetos que componen el módulo en la vista y en la base de datos. Controla el flujo de proceso de firmar un documento a través de un certificado autogenerado o un .p12 cargado desde un dispositivo de almacenamiento.	osv.osv pdfSigner OpenSSL.crypto os

2.9 Arquitectura del sistema

Autores como (Clements, y otros, 2003) definen la arquitectura de software como: “la estructura o estructuras del sistema, lo cual abarca componentes de software, las propiedades visibles externamente de esos componentes, y las relaciones entre ellas”.

Al diseñar una arquitectura de software se debe crear y representar componentes que interactúen entre ellos y realicen tareas específicas, además de organizarlos de tal manera que se logren los requisitos establecidos. A continuación se describen las arquitecturas utilizadas para el desarrollo del módulo:

2.9.1 Cliente-Servidor

La arquitectura cliente-servidor define una relación entre dos aplicaciones donde el cliente envía peticiones al servidor y este último le envía las respuestas. El servidor se ejecuta independientemente del cliente, este maneja la lógica de negocio y se comunica con la base de datos, siendo el cliente quien muestra la información a los usuarios permitiendo que interactúe con el servidor (Desongles, y otros, 2006).

Esta arquitectura que utiliza el módulo de Gestión Documental del OpenERP se basa en que el cliente accede al servidor del OpenERP mediante un navegador web a través del protocolo HTTPS, este servidor contiene la mayoría de los elementos de configuración relacionados con esta herramienta. En el módulo implementado el usuario puede crear documentos solicitando la firma digital, estos documentos son almacenados en el servidor OpenERP y este último a su vez se comunica con la base de datos PostgreSQL. En la figura 5 se muestra la arquitectura cliente- servidor del módulo implementado.

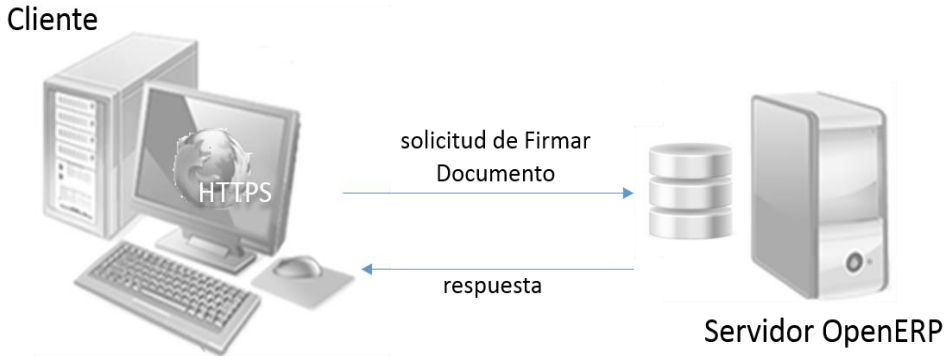


Fig. 5 Arquitectura Cliente-Servidor.

2.9.2 Modelo-Vista-Controlador

La arquitectura Modelo Vista Controlador (MVC) trabaja con tres componentes por separado, los datos de una aplicación, la interfaz de usuario y la lógica de control. El controlador se encarga de interpretar y dar sentido a las instrucciones que realiza el usuario, enviando el mensaje de acción al modelo y a la vista para luego realizar los cambios que sean necesarios (Eslava, 2013).

En el módulo desarrollado se accede a través de un navegador web a la interfaz crear documento para luego permitir la firma digital de este. La vista es definida en un XML que interactúa con la clase controladora del módulo Gestión Documental del OpenERP, la cual se encarga de procesar las interacciones del usuario y realizar los cambios apropiados en el modelo o en la vista. En la figura 6 se muestra la arquitectura Modelo-Vista-Controlador del módulo de firma digital de OpenERP.

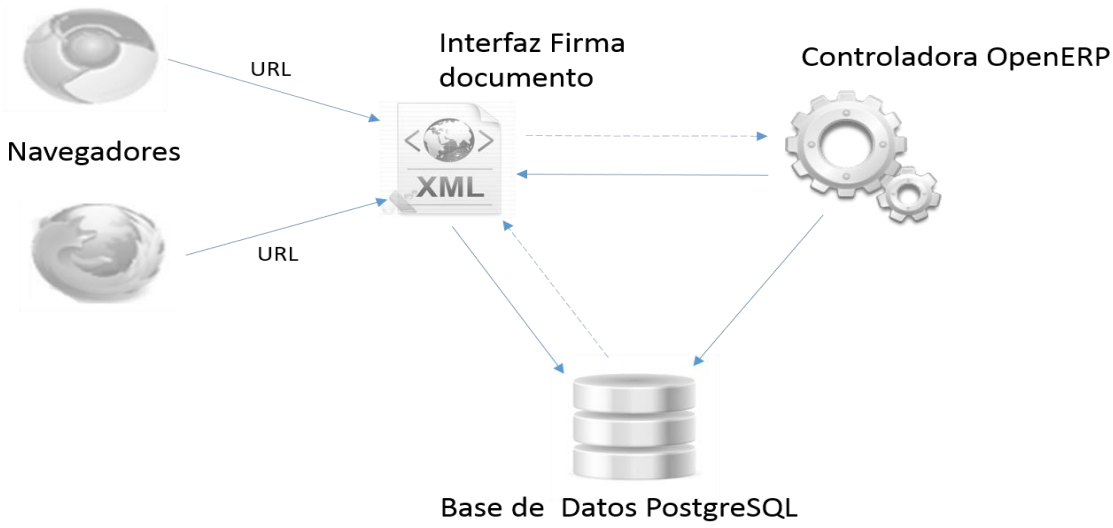


Fig. 6 Arquitectura Modelo-Vista-Controlador.

2.10 Patrones de diseño

Los patrones de diseño son "combinaciones de componentes, casi siempre clases y objetos que por experiencia se sabe que resuelven ciertos problemas de diseño comunes", así lo define Eric Braude en el 2003. Dentro de los patrones de diseño se encuentran los GRASP (*General Responsibility Assignment Software Patterns*, en español, Patrones Generales de Asignación de Responsabilidades de Software) y los GOF (*Gang of Four*, en español, Grupo de los cuatro). A continuación se muestra cada uno de los patrones utilizados para la implementación de la solución.

2.10.1 Patrones GRASP

Los patrones GRASP describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades (Grosso, 2011). Se muestran a continuación los patrones utilizados en la solución y un ejemplo de su uso:

- Experto: Asigna una responsabilidad al experto en información, es decir a la clase que tiene la información necesaria para realizar la responsabilidad (Grosso, 2011). En la clase *pdfSigner* se realizan todas las operaciones para el firmado de un documento. El método *Sign()* se ocupa de firmar un documento y retornar el documento firmado.

- Creador: Guía la asignación de responsabilidades relacionadas con la creación de objetos para identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases (Grosso, 2011). El método *createPkcs7()* es utilizado para crear objetos del estándar *PKCS7*. La clase cms genera objetos de este tipo de estándar.

- Controlador: Asigna la responsabilidad de recibir o manejar un mensaje de evento del sistema donde define el método para la operación del sistema y es un objeto que no pertenece a la interfaz de usuario (Grosso, 2011). La clase controladora *myDocument* cuenta con el método *pdfSign()* donde interviene en el flujo de firmar documentos implementado en la clase *pdfSigner*, es decir accede específicamente al método sign() para realizar este evento.

- Alta Cohesión: La cohesión funcional de las responsabilidades de una clase deben estar altamente relacionadas para que la información que almacena una clase sea coherente y esté (en la medida de lo posible) relacionada con la clase (Grosso, 2011). La clase *x509Parser* transforma la estructura de un certificado de la clase *x509*, donde cada clase realiza sus tareas afines con la utilidad.

2.11 Diagrama de clases del diseño

El diagrama de clases se utiliza principalmente para el modelado de sistemas orientados a objetos. Es el diagrama principal para el análisis y diseño de una solución, compuesto por clases, interfaces y relaciones, estas relaciones pueden ser de dependencia, de asociación y de generalización. Mediante un diagrama de clases se puede modelar el esquema de una base de datos (Marcos, y otros, 2005).

A continuación se muestra en la figura 7 el diagrama de clases del diseño del módulo desarrollado para evidenciar la relación entre las tarjetas CRC.

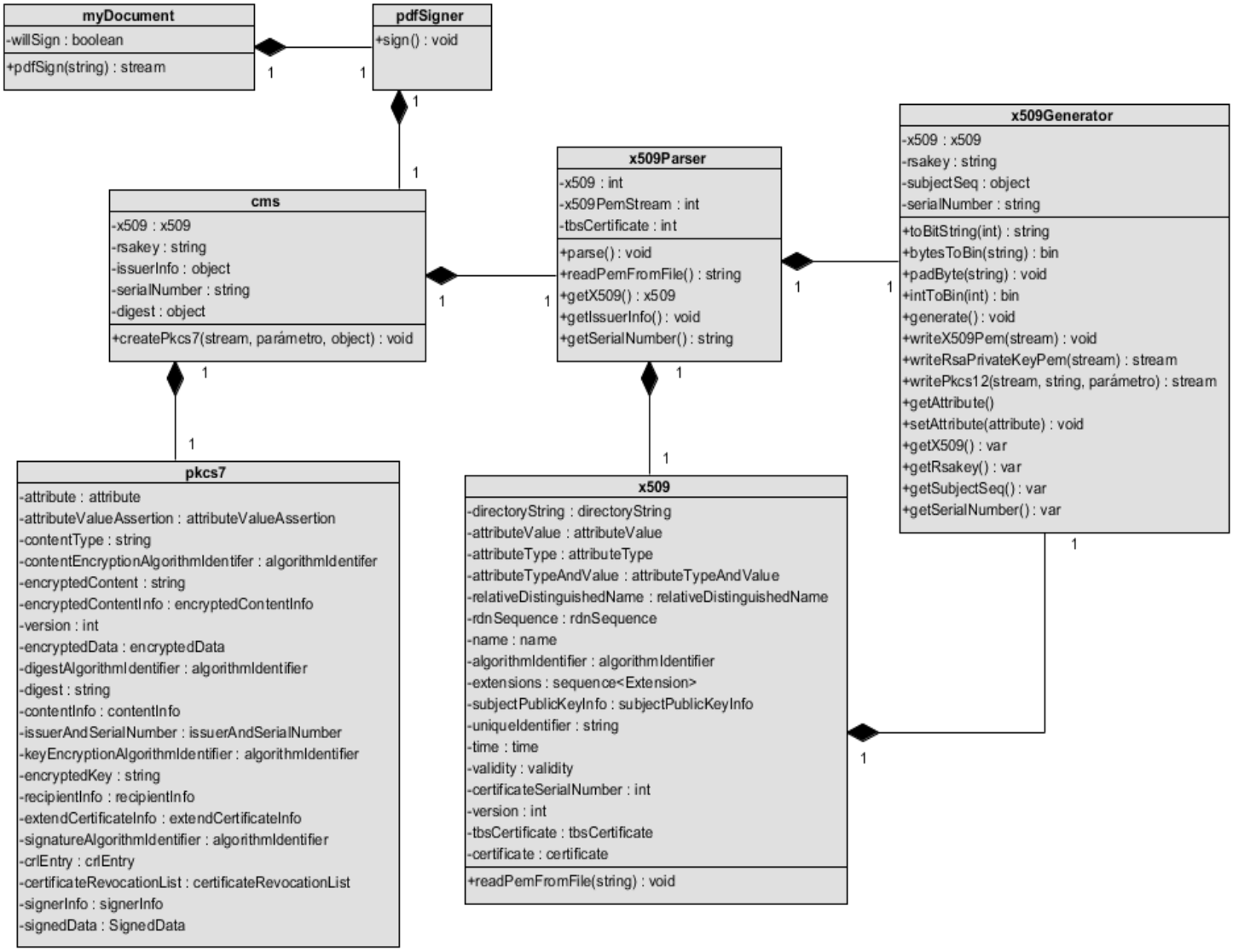


Fig. 7 Diagrama de clases del diseño del módulo de OpenERP para firmas digitales de documentos.

2.12 Conclusiones parciales

En este capítulo se identificaron los requisitos funcionales y no funcionales garantizando el desarrollo del módulo para dar cumplimiento a las necesidades del centro CISED. Las funcionalidades fueron descritas sin emplear un lenguaje técnico a través de las historias de usuario para su posterior implementación, además se realizaron las tarjetas CRC para representar las clases del sistema con sus responsabilidades y clases colaboradoras. La descripción de las arquitecturas a utilizar y los patrones de diseño seleccionados proporcionaron un mejor entendimiento de la solución. La realización del diagrama de clases del diseño, posibilitó una clara visión del problema en cuestión, de manera que las clases que fueron identificadas y sus relaciones facilitan el trabajo para su posterior implementación.

Capítulo 3: Implementación y Prueba del módulo de OpenERP para firmas digitales de documentos.

En el presente capítulo se abordan dos fases fundamentales en ciclo de desarrollo propuesto por la metodología de desarrollo de software XP, las fases de implementación y prueba. Se describen los estándares de codificación empleados, así como las tareas de ingeniería para descomponer las historias de usuario en tareas de programación. La calidad de la solución se muestra mediante las pruebas realizadas al módulo implementado.

3.1 Implementación

La implementación es fundamental en el proceso de desarrollo de un software porque en esta etapa los programadores empiezan a definir el código fuente realizando cada una de las funcionalidades especificadas en las historias de usuario.

3.1.1 Bibliotecas de Python utilizadas

Las bibliotecas que ofrece Python permiten a los programadores una mejor implementación garantizando su calidad. En esta investigación se utilizaron algunas de las bibliotecas de Python, estas se muestran a continuación:

- **PyDev:** Permite a los usuarios utilizar Eclipse para Python y el desarrollo Jython.
- **Pypdf:** Se utiliza para manipular páginas y contenido de documentos con formato PDF.
- **Pyasn1:** Almacena e intercambia datos estructurados entre programas y sistemas. Esta biblioteca hace que sea más fácil para los programadores desarrollar, depurar y experimentar con protocolos ASN.1 (Abstract Syntax Notation One, en español, notación sintáctica abstracta 1).
- **Tlslite:** Implementa SSL y el protocolo criptográfico TLS (Transport Laver Security, en español, seguridad en la capa de transporte).
- **Hashlib:** Permite realizar cifrados de los algoritmos SHA1, SHA224, SHA256, SHA384, SHA512 y MD5.
- **Crypto:** Biblioteca contenida en el paquete OpenSSL que implementa los algoritmos criptográficos RSA, DH y DSA (Digital Signature Algorithm, en español, Algoritmo de Firma

digital). Además crea certificados X.509, CSR (Certificate Signing Request, en español, Solicitud de Firmar un Certificado) y CRL.

- **Datetime:** Biblioteca que contiene funciones y clases para trabajar con fechas y tiempos tanto junto como por separados.
- **Os:** Permite acceder a funcionalidades dependientes del Sistema Operativo como aquellas que refieren información sobre el entorno del mismo y permite manipular la estructura de directorios.
- **Math:** Provee el acceso a funciones y constantes matemáticas.
- **Python- unittest:** Se utilizada para realizar pruebas unitarias, para probar las funciones y todos los casos posibles para cada una de ellas.

3.1.2 Tareas de ingeniería

En la fase de implementación de la metodología XP uno de los artefactos que se generan son las tareas de ingeniería. Estas se realizan para descomponer las historias de usuario en tareas de programación y asignar a los desarrolladores lo que debe programar en cada iteración (Sommerville, 2005). Las tablas siguientes muestran las tareas a desarrollar por cada HU en la primera iteración.

Tabla 17: Descripción de la tarea de ingeniería 1.

Tarea de Ingeniería	
Número Tarea: 1	**Nombre de la Historia de Usuario:** Generar nuevo PDF.
Nombre Tarea: Permitir generar un nuevo PDF.	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados (días): 3**
Fecha Inicio: 27/02/2014	**Fecha Fin:** 02/03/2014
Programador Responsable: Yander Morfa González.	
Descripción: Esta tarea permite tomar el PDF original y guardar su contenido en un documento nuevo haciéndolo página por página.	

Tabla 18: Descripción de la tarea de ingeniería 2.

Tarea de Ingeniería	
Número Tarea: 2	**Nombre de la Historia de Usuario:** Generar nuevo PDF.
Nombre Tarea: Insertar cuño de identificación.	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados (días):** 4
Fecha Inicio: 02/03/2014	**Fecha Fin:** 6/03/2014
Programador Responsable: Yander Morfa González.	
Descripción: Esta tarea adiciona en el documento nuevo un cuño como prueba de que ha sido firmado.	

Tabla 19: Descripción de la tarea de ingeniería 3.

Tarea de Ingeniería	
Número Tarea: 3	**Nombre de la Historia de Usuario:** Modificar esquema XML del PDF.
Nombre Tarea: Agregar diccionarios.	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados (días):** 7
Fecha Inicio: 06/03/2014	**Fecha Fin:** 13/03/2014
Programador Responsable: Yander Morfa González.	
Descripción: Esta tarea permite agregar los diccionarios de formularios interactivos que es donde se guarda toda la información de firma.	

Tabla 20: Descripción de la tarea de ingeniería 4.

Tarea de Ingeniería	
Número Tarea: 4	**Nombre de la Historia de Usuario:** Modificar esquema XML del PDF.
Nombre Tarea: Agregar campos de firma.	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados (días):** 7
Fecha Inicio: 13/03/2014	**Fecha Fin:** 20/03/2014
Programador Responsable: Yander Morfa González.	

Descripción: Esta tarea agrega los campos de firma en los documentos PDF.

Tabla 21: Descripción de la tarea de ingeniería 5.

Tarea de Ingeniería	
Número Tarea: 5	**Nombre de la Historia de Usuario:** Calcular función resumen.
Nombre Tarea: Calcular función resumen.	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados (días):** 7
Fecha Inicio: 20/03/2014	**Fecha Fin:** 27/03/2014
Programador Responsable: Lissette Valdés García.	
Descripción: Esta tarea calcula la función resumen a través del algoritmo sha-1.	

Tabla 22: Descripción de la tarea de ingeniería 6.

Tarea de Ingeniería	
Número Tarea: 6	**Nombre de la Historia de Usuario:** Generar un certificado de estándar X.509.
Nombre Tarea: Generar claves privadas.	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados (días):** 7
Fecha Inicio: 27/03/2014	**Fecha Fin:** 03/04/2014
Programador Responsable: Yander Morfa González.	
Descripción: Esta tarea genera las claves privadas a través del algoritmo RSA.	

Tabla 23: Descripción de la tarea de ingeniería 7.

Tarea de Ingeniería	
Número Tarea: 7	**Nombre de la Historia de Usuario:** Generar un certificado de estándar X.509.
Nombre Tarea: Calcular tabla *hash*.	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados (días):** 5
Fecha Inicio: 03/04/2014	**Fecha Fin:** 08/04/2014

Programador Responsable: Lissette Valdés García.
Descripción: Esta tarea calcula la tabla *hash* del estándar X.509.

Tabla 24: Descripción de la tarea de ingeniería 8.

Tarea de Ingeniería	
Número Tarea: 8	**Nombre de la Historia de Usuario:** Generar un certificado de estándar X.509.
Nombre Tarea: Guardar clave privada.	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados (días):** 5
Fecha Inicio: 08/04/2014	**Fecha Fin:** 13/04/2014
Programador Responsable: Yander Morfa González.	
Descripción: Esta tarea escribe la clave privada RSA en un fichero PEM.	

Tabla 25: Descripción de la tarea de ingeniería 9.

Tarea de Ingeniería	
Número Tarea: 9	**Nombre de la Historia de Usuario:** Generar un certificado de estándar X.509.
Nombre Tarea: Guardar estándar X.509.	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados (días):** 4
Fecha Inicio: 13/04/2014	**Fecha Fin:** 17/04/2014
Programador Responsable: Yander Morfa González.	
Descripción: Esta tarea guarda el X.509 dentro de un fichero .PEM.	

Tabla 26: Descripción de la tarea de ingeniería 10.

Tarea de Ingeniería	
Número Tarea: 10	**Nombre de la Historia de Usuario:** Cargar un certificado digital.
Nombre Tarea: Cargar el certificado .p12.	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados (días):** 12

Fecha Inicio: 17/04/2014	**Fecha Fin:** 01/05/2014
Programador Responsable: Lissette Valdés García.	
Descripción: Esta tarea permite al usuario cargar su propio certificado .p12.	

Tabla 27: Descripción de la tarea de ingeniería 11.

Tarea de Ingeniería	
Número Tarea: 11	**Nombre de la Historia de Usuario:** Firmar documento.
Nombre Tarea: Elaborar prototipo de interfaz.	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados(días):** 7
Fecha Inicio: 01/05/2014	**Fecha Fin:** 08/05/2014
Programador Responsable: Lissette Valdés García.	
Descripción: Esta tarea sirve como guía para su posterior implementación ya que muestra cómo se visualizará la funcionalidad de firmar documento.	

Tabla 28: Descripción de la tarea de ingeniería 12.

Tarea de Ingeniería	
Número Tarea: 12	**Nombre de la Historia de Usuario:** Firmar documentos PDF.
Nombre Tarea: Permitir firma de un documento.	
Tipo de Tarea : Desarrollo Desarrollo / Corrección / Mejora / Otra (especificar)	**Puntos Estimados(días):** 7
Fecha Inicio: 08/03/2014	**Fecha Fin:** 15/03/2014
Programador Responsable: Lissette Valdés García.	
Descripción: Esta tarea permitirá firmar el documento PDF utilizando el certificado autogenerado.	

3.1.3 Estándares de codificación

Los estándares de codificación se usan durante la fase de implementación de un software para darle una estructura al código facilitando su comprensión y mantenimiento. La metodología XP exige el uso de estos estándares de programación para que el código sea legible y exista una buena comunicación entre programadores, de tal forma que se logre disminuir los riesgos de introducir errores que no son detectados por lo compiladores, reduciendo el tiempo y coste de las actividades de refinación y pruebas

necesarias para la detección y corrección de los mismos (Beck, 2002). A continuación el estándar que se utiliza en la implementación y un ejemplo de su aplicación:

Tabla 29: Estándares de codificación.

Comentarios, separadores, líneas y espacios en blancos		
Ubicación de comentarios.	Inicio de una clase y de cada función.	Describir brevemente el objetivo de una clase y de cada función. Ejemplos: ' ' ' Esto es un comentario al inicio de una clase ' ' ' class x509Generator(object): cuerpo de la clase

Líneas en blanco.	Antes y después de cada función.	Dejar una línea en blanco entre las declaraciones de cada función. Ejemplo:
		def getObjectsCount(self):
		cuerpo de la función
		// espacio en blanco
		def isSigned(self):
		cuerpo de la función

Espacios en blanco.	Entre operadores aritméticos y lógicos.	Usar un espacio en blanco entre los miembros de operaciones aritméticas y lógicas. Ejemplos: self.x509 = xp.getX509() self._newobjectscount += 1
Clases, objetos, funciones, atributos		

Apariencia de clases y funciones.	Simples y descriptivos.	Las clases y funciones comenzarán con letra inicial minúscula, en el caso de ser un nombre compuesto por más de una palabra no serán separadas, y el resto de las palabras comenzarán con letra inicial mayúscula. Ejemplos:
		class x509Generator(object):
		def toBitString(self, num):

Nombre de clases y objetos.	Relacionados con el propósito de su función.	Nombrar las clases de manera tal que con solo leerla se note su objetivo. Ejemplo: class pdfSigner(object):
Apariencia de atributos.	Letras minúsculas.	Los atributos deben ser escritos en minúsculas y su nombre debe guardar relación con el valor que almacena. Ejemplo: self.x509 = None self.rsakey = None

3.1.4 Diagrama de despliegue

El diagrama de despliegue describe la arquitectura física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. Entre los nodos se establecen relaciones que significan que existe comunicación entre ellos. Un nodo es una unidad material capaz de recibir y ejecutar software, la mayoría de las veces son ordenadores. Este diagrama es utilizado en el diseño e implementación de la solución (Debrauwer, y otros, 2013). A continuación el diagrama de despliegue del módulo de OpenERP para firmas digitales de documentos.

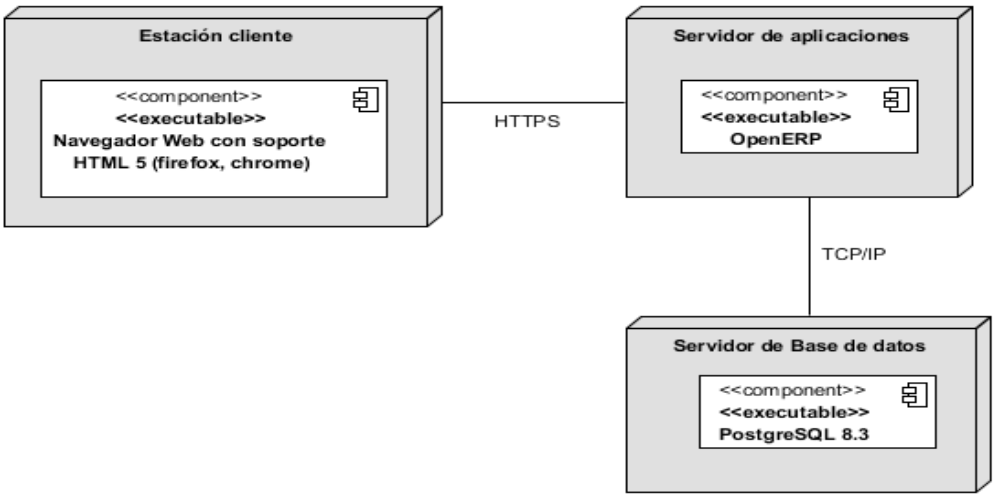


Fig. 8 Diagrama de despliegue del módulo de OpenERP para firmas digitales de documentos.

3.2 Pruebas

Las pruebas del software son un elemento crítico para la garantía de calidad del software y representan una revisión final de las especificaciones, del diseño y de la codificación. En cada fase del ciclo de vida

de desarrollo del software se plantea un conjunto de pruebas que permiten verificar que el software desarrollado satisface las especificaciones de esa fase (Alonso, y otros, 2005).

Las pruebas principales que se realizan en un proceso de desarrollo basado en la metodología XP son las pruebas de unidad, desarrolladas por los programadores encargadas de verificar el código de forma automática y las pruebas de aceptación, destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente (Beck, 2002).

3.2.1 Pruebas unitarias

La realización de pruebas unitarias antes de comenzar la codificación es fundamental en la metodología de desarrollo XP. Estas pruebas deben tener una estructura que permita ejecutarlas repetidas veces y con facilidad. Los requisitos funcionales en XP se expresan como historias de usuario, donde el equipo de desarrollo evalúa cada historia y las divide en tareas. Cada tarea representa una característica diferente del sistema y se puede diseñar entonces una prueba de unidad para verificar la implementación descrita en esa tarea (Sommerville, 2005).

Para realizar las pruebas unitarias del módulo implementado, se utilizó una librería llamada “unittest”, desarrollada para automatizar estas pruebas en Python.

En las siguientes tablas se muestran los casos de prueba aplicados a algunas de las funcionalidades más importantes del módulo.

Tabla 30: Caso de prueba unitaria “test_isSigned”.

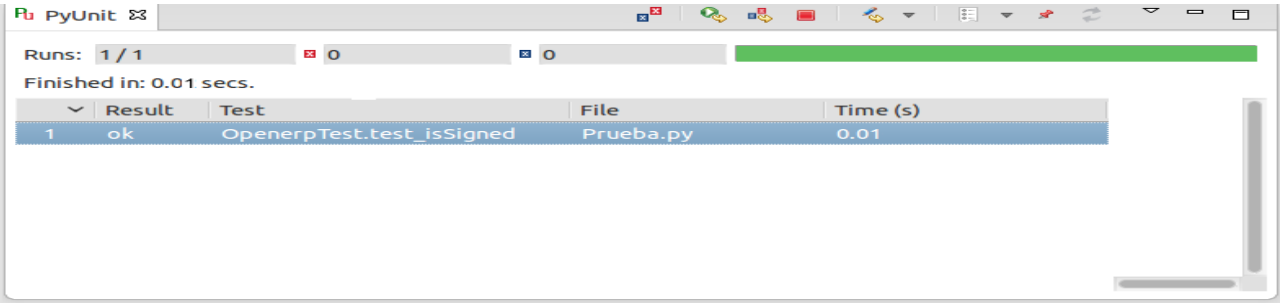
Caso de prueba unitaria		
Nombre de prueba: test_isSigned		**Id. de prueba:** 1.
Descripción del caso de prueba: Verifica si el documento contiene las banderas de firma.		
Entradas: -		
Criterio de aceptación: Comprueba que el resultado sea el esperado.		
Resultado: 		

Tabla 31: Caso de prueba unitaria “test_insertPkcs7”.

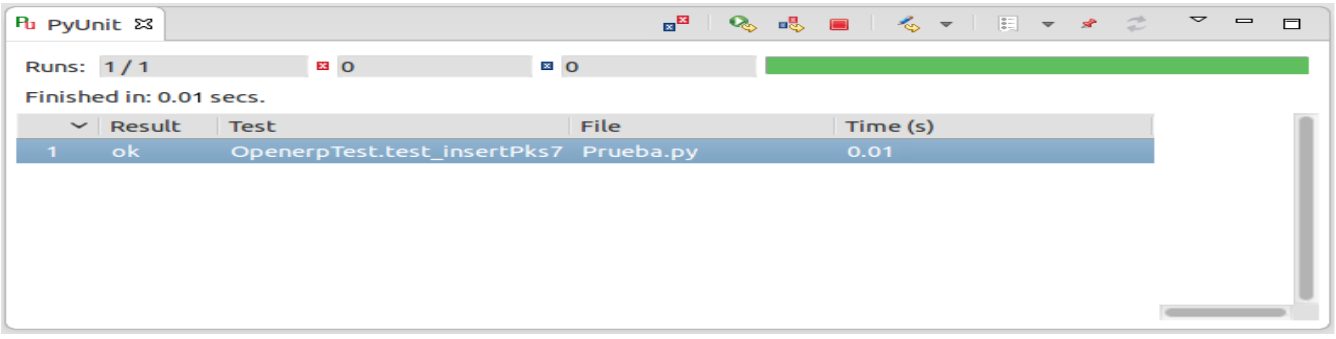
Caso de prueba unitaria	
Nombre de prueba: test_insertPkcs7	**Id. de prueba:** 2
Descripción del caso de prueba: Inserta la estructura pkcs7 para realizar el proceso de firma.	
Entradas: resumen	
Criterio de aceptación: Comprueba que el resultado sea el esperado.	
Resultado: 	

Tabla 32: Caso de prueba unitaria “test_calcSha1”.

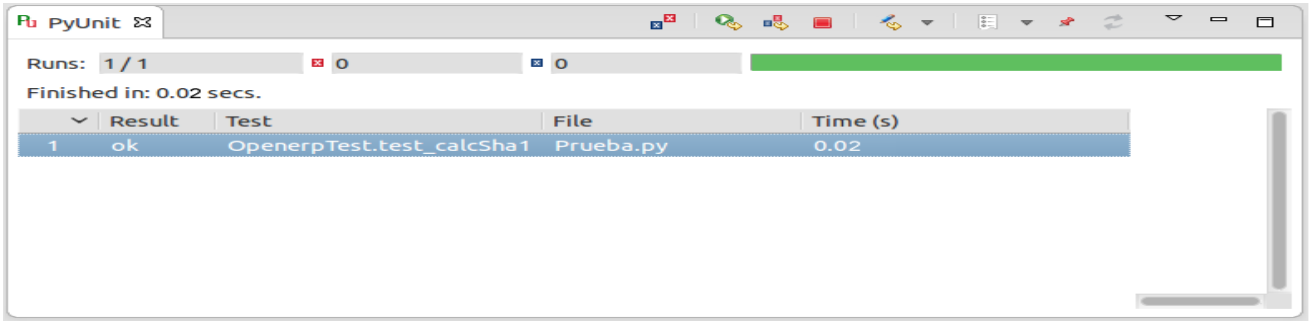
Caso de prueba unitaria	
Nombre de prueba: test_calcSha1	Id. de prueba: 3
Descripción del caso de prueba: Calcula el resumen del documento con el algoritmo sha-1.	
Entradas: Tamaño del bloque.	
Criterio de aceptación: Comprueba que el resultado sea el esperado.	
Resultado: 	

Tabla 33: Caso de prueba unitaria “testSing”.

Caso de prueba unitaria		
Nombre de prueba: testSign		**Id. de prueba:** 4
Descripción del caso de prueba: Firma un documento y devuelve un documento firmado.		
Entradas: -		

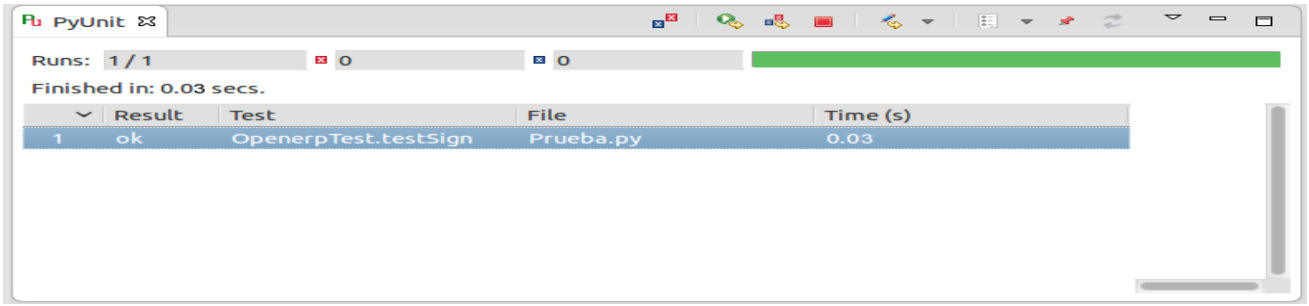
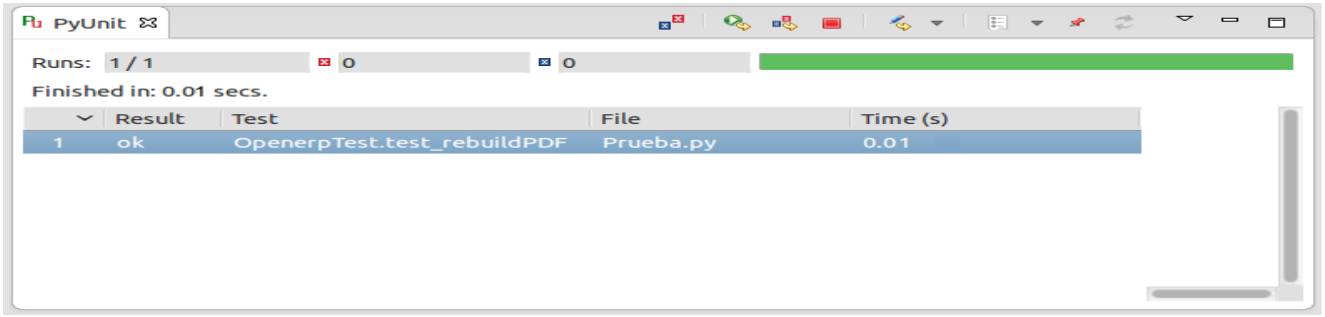
Criterio de aceptación: Comprueba que el resultado sea el esperado.
Resultado: 

Tabla 34: Caso de prueba unitaria “test_rebuildPDF”.

Caso de prueba unitaria	
Nombre de prueba: test_rebuildPDF	**Id. de prueba:** 5
Descripción del caso de prueba: Genera un nuevo PDF, página por página y agrega el cuño en la primera página del documento.	
Entradas: -	
Criterio de aceptación: Comprueba que el resultado sea el esperado.	
Resultado: 	

3.2.2 Pruebas de integración

Las pruebas de integración parten de los componentes individuales previamente probados y tienen como objetivo descubrir errores que se pueden producir en la interacción entre los módulos. En teoría, la combinación de componentes válidos debería dar como resultado software en el que no se detectan errores, pero en la práctica hay múltiples ocasiones en las que las pruebas de unidad no detectan errores que sí se desvelan al ejecutar pruebas sobre el software integrado.

En cuanto a la estrategia a seguir en las pruebas de integración, existen dos enfoques: descendente (*top-down*) y ascendente (*bottom-up*). En la integración descendente se comienza la prueba por el

componente principal, sustituyendo por resguardos los componentes que son llamados por éste. Sobre dicho conjunto se realiza una primera fase de pruebas. Luego, se integran los componentes subordinados al principal y se continúan realizando pruebas sucesivamente hasta completar la integración del software completo.

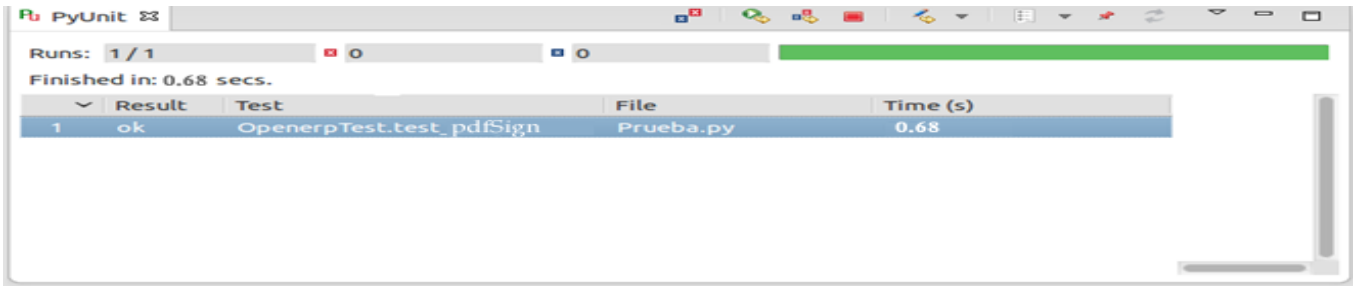
En la integración ascendente las pruebas se hacen en sentido inverso, comenzando por los componentes que no necesitan llamar a ningún otro y finalizando por el componente principal. Mientras que en la integración descendente los elementos auxiliares son, principalmente, resguardados, en la integración ascendente se utilizan conductores para sustituir a los componentes que llaman al software bajo prueba (Tuya, y otros, 2007).

El módulo extendido de Gestión Documental que permite la firma digital debe integrarse a la herramienta OpenERP, se utiliza el enfoque ascendente para realizar las pruebas de integración, permitiendo construir y verificar las funcionalidades desde el nivel más bajo y finalizar en la funcionalidad principal, en este caso es firmar un documento PDF.

A continuación se muestra el caso de prueba de integración correspondiente al proceso para firmar un documento PDF.

Tabla 35: Caso de prueba de integración “test_pdfSign”.

Caso de prueba de integración	
Nombre de prueba: test_pdfSign	Id. de prueba: 1.
Descripción del caso de prueba: Garantiza que se le aplique la firma digital a un documento PDF en la herramienta OpenERP.	
Entradas: Identificador de área en la estructura administrativa de la UCI.	

Criterio de aceptación: Se realiza el proceso de firma digital a un documento PDF.
Resultado: 

3.2.3 Pruebas de aceptación

Las pruebas de aceptación son realizadas para verificar que el sistema cumpla con las necesidades y especificaciones del cliente. Los clientes deben ayudar para desarrollar las pruebas de aceptación a las historias que se tienen que implementar en cada entrega del sistema. En XP, este tipo de prueba, así como su desarrollo, son incrementales.

Fundamentalmente, las pruebas se escriben como un componente ejecutable antes de que se implemente la tarea. Una vez que se ha implementado el software, se pueden ejecutar las pruebas inmediatamente. Este componente de pruebas debe ser una aplicación independiente, debe simular el envío de la entrada a probar y debe verificar que el resultado cumple la especificación de salida. Estas pruebas se realizan al final antes del despliegue del sistema y generalmente lo realizan los usuarios finales (Bruegge, y otros, 2002).

A continuación los casos de pruebas de aceptación correspondiente a Firmar documento con extensión PDF.

Tabla 36: Caso de prueba de aceptación "Firmar documento PDF utilizando un certificado digital autogenerado".

Caso de prueba de aceptación	
Nombre de HU: Firmar documento PDF.	**Id. de prueba:** 1
Nombre del caso de prueba: Firmar documento PDF utilizando un certificado digital autogenerado.	
Descripción del caso de prueba: Permite crear un documento añadiéndole la firma digital utilizando un certificado autogenerado.	
Condiciones de ejecución: Debe estar autenticado.	
Entrada / Pasos de ejecución: ▪ Entrar en el campo documento. ▪ Pulsar la opción “Crear documento”. ▪ Seleccionar la opción de utilizar certificado autogenerado. ▪ Subir el PDF. ▪ Marcar “Firma Digital”. ▪ Pulsar la opción “Guardar”.	
Resultado esperado: El sistema crea un documento añadiéndole la firma digital.	
Evaluación de la prueba: Satisfactoria.	

Tabla 37: Caso de prueba de aceptación "Firmar documento PDF utilizando un certificado digital guardado en un dispositivo de almacenamiento".

Caso de prueba de aceptación	
Nombre de HU: Firmar documento PDF.	**Id. de prueba:** 2
Nombre del caso de prueba: Firmar documento PDF utilizando un certificado digital guardado en un dispositivo de almacenamiento	
Descripción del caso de prueba: Permite crear un documento añadiéndole la firma digital utilizando un certificado que está guardado en un dispositivo de almacenamiento.	
Condiciones de ejecución: Debe estar autenticado.	
Entrada / Pasos de ejecución: ▪ Entrar en el campo documento. ▪ Pulsar la opción “Crear documento”. ▪ Seleccionar el certificado .p12. ▪ Cargar el certificado .p12. ▪ Escribir la contraseña del certificado .p12. ▪ Subir el PDF. ▪ Marcar “Firma Digital”. ▪ Pulsar la opción “Guardar”.	
Resultado esperado: El sistema crea un documento añadiéndole la firma digital.	
Evaluación de la prueba: Satisfactoria.	

3.2.4 Resultado de las pruebas

Las pruebas unitarias se realizaron a los métodos más importantes del módulo una vez concluida su implementación. Estas pruebas se fueron realizando a medida que se implementaban las historias de usuario para así detectar cualquier problema de funcionalidad. En total hubo 6 no conformidades siendo corregidas en el momento que fueron detectadas. En la siguiente figura se muestran la cantidad de no conformidades encontradas en cada iteración.

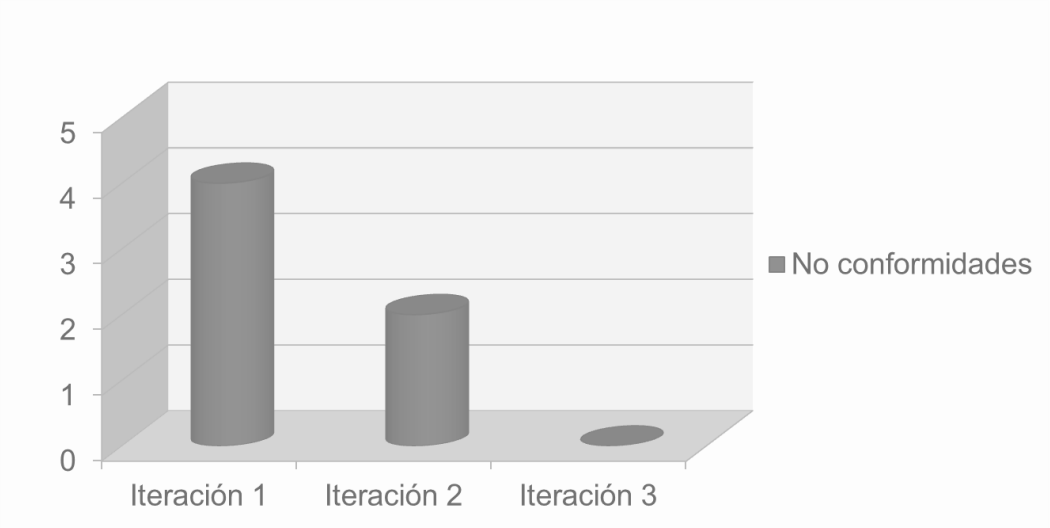


Fig. 9 Pruebas unitarias.

Las pruebas de integración aplicadas al módulo extendido de Gestión Documental en la herramienta OpenERP, requirieron de previas pruebas unitarias a los métodos que garantizan realizar la firma digital de documentos. Una vez resueltas las no conformidades encontradas en las pruebas unitarias, se realizó la prueba de integración a la funcionalidad principal "Firmar documento PDF" obteniendo una correcta vinculación al módulo de Gestión Documental de OpenERP.

Se realizaron dos casos de pruebas de aceptación a la funcionalidad principal del módulo implementado con el objetivo de probar que se cumple con las especificaciones del cliente. Las pruebas de aceptación se realizaron al finalizar la tercera iteración donde se obtuvo un módulo con la calidad requerida.

3.3 Conclusiones parciales

En la etapa de implementación se definieron los estándares de codificación para lograr uniformidad y legibilidad en el código, además se realizaron las tareas de ingeniería por cada una de las historias de usuario permitiendo a los programadores implementar con facilidad las funcionalidades. Se desarrollaron las pruebas unitarias, de integración y de aceptación, presentándose en estas, los casos de pruebas correspondientes y se analizó los resultados obtenidos permitiendo identificar y corregir las no conformidades encontradas, garantizando así la calidad del módulo construido.

Conclusiones Generales

Una vez culminada la investigación se puede afirmar que se le dio cumplimiento a los objetivos planteados, arribando a las siguientes conclusiones:

- El estudio de los principales conceptos relacionados a la firma digital de documentos permitió sentar las bases para el desarrollo del módulo de OpenERP para firmas digitales de documentos.
- A partir de la realización del análisis y diseño se obtuvo como resultado los artefactos necesarios para guiar el desarrollo del mismo.
- La implementación del módulo dio cumplimiento a los requisitos funcionales identificados en las fases de análisis y diseño.
- El diseño y ejecución de las pruebas unitarias, de integración y de aceptación permitieron comprobar el correcto funcionamiento del módulo de OpenERP para firmas digitales de documentos.
- Como resultado se obtuvo un módulo que realiza la firma digital de documentos en la herramienta OpenERP y garantiza la integridad, el no repudio del origen y autenticidad de los documentos.

Recomendaciones

Se recomienda a los interesados en la presente investigación:

- Implementar la firma digital a otros tipos de formatos de documentos (Word, XML).
- Utilizar en el centro CISED la herramienta OpenERP para la firma digital de documentos, en caso que un cliente de cualquier proyecto, necesite dar valor probatorio a los documentos digitales que se gestionan.

Referencias Bibliográficas

1. **Alonso, Fernando, Martínez Normand, Loic y Segovia Pérez, Francisco Javier. 2005.** *Introducción a la ingeniería de software. Modelos de desarrollo de programas.* 2005. ISBN: 84-96477-00-2.
2. **Asociación de Empresas de Software Libre . 2011.** Sinadura. [En línea] 2011. http://www.sinadura.net/es/.
3. **BA. 2013.** Firma Digital. [En línea] 2013. [Citado el: 16 de 12 de 2013.] http://www.firmadigital.gba.gov.ar/?q=funcionamiento.
4. **Beck, Kent. 2000.** *Extreme Programing.* Massachusetts : s.n., 2000. ISBN: 0201616416.
5. **Bosselaers, Antoon. 2012.** The hash function RIPEMD-160. [En línea] 2012. http://homes.esat.kuleuven.be/~bosselae/ripemd160.html.
6. **Bruegge, Bernd y Dutoit, Allen h. 2002.** *Ingeniería de Software Orientado a Objetos.* 2002.
7. **Calderón, Amaro , Dámaris, Sarah y Valverde Rebaza, Jorge Carlos. 2007.** *Metodologías Ágiles.* Trujillo : s.n., 2007.
8. **Canós, José H., Letelier, Patricio y Penadés, María Carmen. 2007.** Metodologías Ágiles en el Desarrollo de Software. [En línea] 2007. [Citado el: 8 de diciembre de 2013.] www.willydev.net/descargas/prev/TodoAgil.Pdf.
9. **Clements, P, Bass, L y Kazman, R. 2003.** *Software Architecture in Practice. .* s.l. : SEI Series in Software Engineering: Addison Wesley, 2003.
10. **Comunidad PostgreSQL.** PostgreSQL: About. [En línea] [Citado el: 12 de 10 de 2013.] http://www.postgresql.org/about/.
11. **Cutanda, David. 2014.** security A(r)TWORK. [En línea] 07 de 04 de 2014. http://www.securityartwork.es/2014/04/07/fundamentos-sobre-certificados-digitales-el-estandar-x-509-y-estructura-de-certificados/.
12. **Debrauwer, Laurent y Der Heyde, Fien Van. 2013.** *UML 2 Iniciación, ejemplos y ejercicios corregidos.* s.l. : ENI, 2013. ISBN:978-2-7460-7994-6.
13. **Desongles, Juan, y otros. 2006.** *Técnicos de soporte informático Grupo III.* s.l. : Mad, 2006. ISBN:84-665-5098-4.
14. **Días, Gabriel , y otros. 2004.** *Seguridad en las Comunicaciones y en la Información.* Madrid : s.n., 2004. 978-84-362-4789-3.
15. **Díaz Orueta, Gabriel, y otros. 2014.** *Procesos y herramientas para la seguridad de redes.* Madrid : s.n., 2014. ISBN: 978-84-362-6838-6.

16. **Eclipse, Fundación. 2012.** The Eclipse Foundation. *Eclipse.* [En línea] 2012. http://www.eclipse.org/.
17. **Eslava, Vicente Javier. 2013.** *El nuevo PHP. Conceptos avanzados.* 2013. ISBN:978-84-686-4434-9.
18. **España Boquera, Maria Carmen. 2003.** *Servicios Avanzados de Telecomunicación.* Madrid : s.n., 2003. ISBN 84-7978-607-8.
19. **Estrada, Alejandro Corletti. 2011.** *Seguridad por niveles.* Madrid : s.n., 2011.
20. **Fernández Domínguez, Jesus Ignacio. 2006.** *La firma electrónica.* Madrid : Reus SA, 2006.
21. **Galli, P. 2005.** *Moglen:GPL 3.0 Rewrite Drive Is No Democracy eWeek.com Linux & Open Source.* 2005.
22. **Grosso, Andrés . 2011.** Prácticas de Software. [En línea] 2011. http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.
23. **Izquierdo, Susana. 2009.** *Apostando por el mercado del ERP .* 2009.
24. **Joskowicz, José. 2008.** *Reglas y Prácticas en eXtreme Programming.* Vigo : s.n., 2008.
25. **Knowlton, Jim. 2009.** *Python.* [Multimedia] Madrid : s.n., 2009.
26. **Larman, Craig. 2004.** *UML Y PATRONES INTRODUCCION AL ANALISIS Y DISEÑO ORIENTADO A OBJETOS.* s.l. : Prentice Hall, 1999. 970-1 7-0261-1.
27. **López Tallón, Alberto. 2011.** Microlopez. [En línea] 14 de noviembre de 2011. [Citado el: 13 de 01 de 2014.] http://www.microlopez.org/2011/11/14/6-herramientas-gratuitas-para-firmar-electronicamente-documentos-pdf-parte-ii-esecure-y-xolidosign/.
28. **López, Manuel José Lucena. 2004.** *Criptografía y Seguridad en Computadoras.* 2004.
29. **Marcos, Esperanza, Vela, Belén y Vara, Juan. 2005.** *Diseño de bases de datos Objetos-Relacionales con UML.* Madrid : DYKINSON, 2005. ISBN:978-84-9982-346-1.
30. **Martín Suárez, Ernesto Raúl y Verano Escalona, Wilhem Manuel. 2011.** *Módulo de Conectores para el Subsistema de Aprovisionamiento de Usuarios del Sistema de Administración de Identidades.* La Habana : s.n., 2011.
31. **Martínez Equihua, Saúl. 2007.** *Biblioteca Digital: Conceptos, Recursos y Estándares.* Buenos Aires : s.n., 2007.
32. **Mehuron, William . 2000.** *Digital Signature Standard (DSS).* 2000.
33. **Microsoft Office. 2010.** *Manual de office.* 2010.
34. **Moliner López, Francisco Javier. 2005.** *INFORMÁTICOS GENERALITAT VALENCIANA GRUPOS A Y B. TEMARIO BLOQUE ESPECÍFICO VOLUMEN II.* s.l. : Mad,S-L, 2005. ISBN: 84-665-1829-0.

35. **Mu, Yi, Susilo, Willy y Seberry, Jennifer. 2008.** *Information Security and Privacy.* 2008. ISSN: 0302-9743.
36. **Neira Vilalba, Javier. 2011.** *Curso de Certificados Digitales.* 2011.
37. **Nicado, Miriam. 2014.** *RESOLUCION No. 179/14.* Habana : UCI, 2014.
38. **PostgreSQL-es. 2010.** Sobre PostgreSQL. [En línea] 2010. [Citado el: 20 de 01 de 2014.] http://www.postgresql.org.es/sobre_postgresql.
39. **Rodríguez Fernández, Rafael Alejandro . 2012.** *Herramienta de firma digital.* La Habana : s.n., 2012.
40. **Schwaber, Ken y Sutherland, Jeff . 2013.** *The Scrum Guide.* 2013.
41. **Sierra, Maria. 2006.** *Trabajando con Visual Paradigm for UML. .* Cantabria : s.n., 2006.
42. **Sommerville, Ian. 2005.** *INGENIERIA DE SOFTWARE. Séptima edición.* Madrid : PEARSON EDUCACION, 2005. 84-7829-074-5.
43. **Toro López, Francisco J. 2013.** *Administración de proyectos de informática.* Bogotá : s.n., 2013. ISBN:978-958-648-816-7.
44. **Tuya, Javier, Ramos Román, Isabel y Dolado Cosín, Javier. 2007.** *Técnicas cuantitativas para la Gestión en la Ingeniería del Software.* s.l. : Netbiblo, S. L, 2007. ISBN: 978-84-9745-204-5.
45. **Valencia, Universitat Politécnica . 2012.** UNIVERSITAT POLITÉCNICA DE VALENCIA. [En línea] 2012. [Citado el: 26 de 05 de 2014.] http://www.upv.es/contenidos/CD/info/711545normalc.html.
46. **Visual_Paradigm. 2012.** Visual Paradigm. [En línea] 2012. [Citado el: 9 de diciembre de 2013.] http://www.visualparadigm.com/product/vpuml/.
47. **Wells, Don. 2013.** www.extremeprogramming.org. [En línea] 13 de marzo de 2013. http://www.extremeprogramming.org/rules/crccards.html.

Bibliografía Consultada

1. **Alonso, Fernando, Martínez Normand, Loic y Segovia Pérez, Francisco Javier. 2005.** *Introducción a la ingeniería de software. Modelos de desarrollo de programas.* 2005. ISBN: 84-96477-00-2.
2. **Asociación de Empresas de Software Libre . 2011.** Sinadura. [En línea] 2011. http://www.sinadura.net/es/.
3. **Avison, D. y Fitzgerald, G. 1995.** *Information Systems Development: Methodologies, Techniques, and Tools.* s.l. : McGraw-Hill, 1995.
4. **BA. 2013.** Firma Digital. [En línea] 2013. [Citado el: 16 de 12 de 2013.] http://www.firmadigital.gba.gov.ar/?q=funcionamiento.
5. **Barranco de Areba, Jesus. 2001.** *Metodologías del análisis estructurado de sistemas.* 2001. ISBN: 84-8468-043-6.
6. **Beck, Kent. 2002.** *Extreme Programing.* Massachusetts : s.n., 2002. ISBN: 0201616416.
7. **Bosselaers, Antoon. 2012.** The hash function RIPEMD-160. [En línea] 2012. http://homes.esat.kuleuven.be/~bosselae/ripemd160.html.
8. **Bruegge, Bernd y Dutoit, Allen h. 2002.** *Ingeniería de Software Orientado a Objetos.* 2002.
9. **Buschmann, Frank , y otros. 1996.** *Pattern Oriented Software Architecture: A system of patterns.* 1996.
10. **Calderón, Amaro , Dámaris, Sarah y Valverde Rebaza, Jorge Carlos. 2007.** *Metodologías Ágiles.* Trujillo : s.n., 2007.
11. **Canós, José H., Letelier, Patricio y Penadés, María Carmen. 2007.** Metodologías Ágiles en el Desarrollo de Software. [En línea] 2007. [Citado el: 8 de diciembre de 2013.] www.willydev.net/descargas/prev/TodoAgil.Pdf.
12. **Comunidad PostgreSQL.** PostgreSQL: About. [En línea] [Citado el: 12 de 10 de 2013.] http://www.postgresql.org/about/.
13. **Cutanda, David. 2014.** security A(r)TWORK. [En línea] 07 de 04 de 2014. http://www.securityartwork.es/2014/04/07/fundamentos-sobre-certificados-digitales-el-estandar-x-509-y-estructura-de-certificados/.
14. **Desongles, Juan, y otros. 2006.** *Técnicos de soporte informático Grupo III.* s.l. : Mad, 2006. ISBN:84-665-5098-4.
15. **Días, Gabriel , y otros. 2004.** *Seguridad en las Comunicaciones y en la Información.* Madrid : s.n., 2004. 978-84-362-4789-3.
16. **Díaz Orueta, Gabriel, y otros. 2014.** *Procesos y herramientas para la seguridad de redes.* Madrid : s.n., 2014. ISBN: 978-84-362-6838-6.

17. **Domingo, Jesús Ignacio Fernández. 2006.** *La Firma Electrónica.* Madrid : Reus.S.A, 2006. 84-290-1436-5.
18. **Eclipse Foundation. 2013.** About the Eclipse Foundation. [En línea] 2013. [Citado el: 12 de 12 de 2013.] http://www.eclipse.org/org/.
19. **Eclipse, Fundación. 2012.** The Eclipse Foundation. *Eclipse.* [En línea] 2012. http://www.eclipse.org/.
20. **España Boquera, Maria Carmen. 2003.** *Servicios Avanzados de Telecomunicación.* Madrid : s.n., 2003. ISBN 84-7978-607-8.
21. **España. 2013.** *Ley de firma electrónica.* 2013. 5392048110.
22. **Estrada, Alejandro Corletti. 2011.** *Seguridad por niveles.* Madrid : s.n., 2011.
23. **Fernández Domínguez, Jesus Ignacio. 2006.** *La firma electrónica.* Madrid : Reus SA, 2006.
24. **Galli, P. 2005.** *Moglen:GPL 3.0 Rewrite Drive Is No Democracy eWeek.com Linux & Open Source.* 2005.
25. **Gamma, Erich , y otros. 1994.** *Design Patterns: Elements of Reusable Object-Oriented.* 1994. ISBN 0-201-63361-2.
26. **García-Cervigón Hurtado, Alfonso y Alegre Ramos, Maria del Pilar. 2011.** *SEGURIDAD INFORMÁTICA ED.11.* 2011. 978-84-9732-812-8.
27. **González Duque, Raúl.** *Python para todos.*
28. **Grosso, Andrés . 2011.** Prácticas de Software. [En línea] 2011. http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.
29. **Gutiérrez, Jaime y Tena, Juan. 2003.** *Protocolos Criptográficos y Seguridad en Redes.* 2003. 84-8102-345-0.
30. **INAP. 2003.** *Firma digital y Administraciones públicas.* Madrid : s.n., 2003.
31. **Infante, Jorge. 2012.** La Arquitectura de Software desde adentro. *Mejora en el proceso de autorización. Uso de XACML con el Identity Server de WSO2.* [En línea] 10 de 5 de 2012. [Citado el: 11 de 12 de 2013.]
32. **Izquierdo, Susana. 2009.** *Apostando por el mercado del ERP .* 2009.
33. **Joskowicz, José. 2008.** *Reglas y Prácticas en eXtreme Programming.* Vigo : s.n., 2008.
34. **Knowlton, Jim. 2009.** *Python.* [Multimedia] Madrid : s.n., 2009.
35. **Larman, Craig. 2004.** *UML Y PATRONES INTRODUCCION AL ANALISIS Y DISEÑO ORIENTADO A OBJETOS.* s.l. : Prentice Hall, 1999. 970-1 7-0261-1.
36. **Lasala Calleja, Pilar . 2013.** *DERECHO y tecnologías avanzadas.* Zaragoza : s.n., 2013. ISBN 978-84-15770-13-8.

37. **Letelier, Patricio y Penadés, María Carmen .** *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Valencia : Universidad Politécnica de Valencia.
38. **López Tallón, Alberto. 2011.** Microlopez. [En línea] 14 de noviembre de 2011. [Citado el: 13 de 01 de 2014.] http://www.microlopez.org/2011/11/14/6-herramientas-gratuitas-para-firmar-electronicamente-documentos-pdf-parte-ii-esecure-y-xolidosign/.
39. **López, Manuel José Lucena. 2004.** *Criptografía y Seguridad en Computadoras.* 2004.
40. **Marcos, Esperanza, Vela, Belén y Vara, Juan. 2005.** *Diseño de bases de datos Objetos-Relacionales con UML.* Madrid : DYKINSON, 2005. ISBN:978-84-9982-346-1.
41. **Martelli, Alex. 2008.** Python. Guía de referencia. [En línea] 2008. [Citado el: 3 de 11 de 2013.] http://dialnet.unirioja.es/servlet/libro?codigo=318613. ISBN: 978-84-415-2317-3 84-415-2317-7.
42. **Martín Suárez, Ernesto Raúl y Verano Escalona, Wilhem Manuel. 2011.** *Módulo de Conectores para el Subsistema de Aprovisionamiento de Usuarios del Sistema de Administración de Identidades.* La Habana : s.n., 2011.
43. **Martínez Equihua, Saúl. 2007.** *Biblioteca Digital: Conceptos, Recursos y Estándares.* Buenos Aires : s.n., 2007.
44. **Mehuron, William . 2000.** *Digital Signature Standard (DSS).* 2000.
45. **Microsoft Office. 2010.** *Manual de office.* 2010.
46. **Moliner López, Francisco Javier. 2005.** *INFORMÁTICOS GENERALITAT VALENCIANA GRUPOS A Y B. TEMARIO BLOQUE ESPECÍFICO VOLUMEN II.* s.l. : Mad,S-L, 2005. ISBN: 84-665-1829-0.
47. **Mu, Yi, Susilo, Willy y Seberry, Jennifer. 2008.** *Information Security and Privacy.* 2008. ISSN: 0302-9743.
48. **Mühlrad, Daniel . 2008.** *Patrones de diseño.* 2008.
49. **Neira Vilalba, Javier. 2011.** *Curso de Certificados Digitales.* 2011.
50. **Nicado, Miriam. 2014.** *RESOLUCION No. 179/14.* Habana : UCI, 2014.
51. **Pinckaers, Fabien y Gardiner, Geoff. 2009.** *Open ERP, a modern approach to.* 2009.
52. **PostgreSQL-es. 2010.** Sobre PostgreSQL. [En línea] 2010. [Citado el: 20 de 01 de 2014.] http://www.postgresql.org.es/sobre_postgresql.
53. **Pressman, Roger. 2002.** *INGENIERIA DE SOFTWARE. Un enfoque práctico (5ta edición).* Madrid : The McGraw-Hill Companies, 2002. 0-07-709677-0.
54. **Rodríguez Fernández, Rafael Alejandro . 2012.** *Herramienta de firma digital.* La Habana : s.n., 2012.
55. **Rumbaugh, James, Jacobson, Ivar y Booch, Grady. 2000.** *El Lenguaje Unificado de Modelado. Manual de referencia.* s.l. : Addison - Wesley, 2000. 8478290281.

56. **Schwaber, Ken y Sutherland, Jeff . 2013.** *The Scrum Guide.* 2013.
57. **Sierra, Maria. 2006.** *Trabajando con Visual Paradigm for UML. .* Cantabria : s.n., 2006.
58. **Sommerville, Ian. 2005.** *INGENIERIA DE SOFTWARE. Séptima edición.* Madrid : PEARSON EDUCACION, 2005. 84-7829-074-5.
59. **Spain, OpenERP. 2013.** ¿Qué es OpenERP? [En línea] 2013. [Citado el: 14 de 01 de 2014.] http://openerpspain.com/.
60. **Toro López, Francisco J. 2013.** *Administración de proyectos de informática.* Bogotá : s.n., 2013. ISBN:978-958-648-816-7.
61. **Tuya, Javier, Ramos Román, Isabel y Dolado Cosín, Javier. 2007.** *Técnicas cuantitativas para la Gestión en la Ingeniería del Software.* s.l. : Netbiblo, S. L, 2007. ISBN: 978-84-9745-204-5.
62. **Valencia, Universitat Politécnica . 2012.** UNIVERSITAT POLITÉCNICA DE VALENCIA. [En línea] 2012. [Citado el: 26 de 05 de 2014.] http://www.upv.es/contenidos/CD/info/711545normalc.html.
63. **Visual_Paradigm. 2012.** Visual Paradigm. [En línea] 2012. [Citado el: 9 de diciembre de 2013.] http://www.visualparadigm.com/product/vpuml/.
64. **Wells, Don. 2013.** www.extremeprogramming.org. [En línea] 13 de marzo de 2013. http://www.extremeprogramming.org/rules/crccards.html.

Glosario de Términos

A

AES256: Es un esquema de cifrado por bloques adoptado como un estándar de cifrado. Tiene un tamaño de bloque fijo de 128 bits y tamaño de llave de 256 bits.

ASN.1: Es una norma para representar datos independientemente de la máquina que se esté usando y sus formas de representación internas. Es un protocolo de nivel de presentación en el modelo OSI.

C

CA: Es una entidad de confianza, responsable de emitir y revocar certificados digitales, utilizados en la firma electrónica, para lo cual se emplea la criptografía de clave pública.

CAdES: Es un conjunto de extensiones de sintaxis de mensajes criptográficos (CMS) es adecuada para la firma electrónica avanzada.

CMS: Es el estándar del IETF para mensajes protegidos criptográficamente. Formato binario de firma usado para la firma, autenticación, resumen y encriptación de documentos. Fue diseñado, principalmente, para el intercambio de información a partir de Correos Electrónicos. Usa el estándar PKCS#7.

CRL: Es una lista de certificados (más concretamente sus números de serie) que han sido revocados, ya no son válidos y en los que no debe confiar ningún usuario del sistema.

CRM: Sistemas informáticos de apoyo a la gestión de las relaciones con los clientes, a la venta y al marketing. Es un sistema que administra un data *warehouse* (almacén de datos) con la información de la gestión de ventas y de los clientes de la empresa.

CSR: Es un cuerpo de texto cifrado. Contiene información codificada específica para su compañía y el nombre de dominio.

D

DNIe: Es un documento emitido por una autoridad oficial para permitir la identificación de la población de forma personal o virtual.

DSA: Es un algoritmo utilizado para firmar digitalmente documentos.

H

HTML: Es un estándar que sirve de referencia para la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código para la definición de contenido de una página web, como texto, imágenes, etc. Es el lenguaje con el que se definen las páginas web.

O

OCSP: Es un método para determinar el estado de revocación de un certificado digital X.509 usando otros medios que no sean el uso de CRL.

ORM: Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional, utilizando un motor de persistencia.

OpenSSL: Es un paquete de herramientas de administración y bibliotecas relacionadas con la criptografía. Implementan los protocolos de seguridad SSL y TLS. También permite crear certificados digitales que pueden aplicarse a un servidor.

P

PAdES: Es un conjunto de restricciones y extensiones a PDF adecuada para la firma electrónica avanzada.

PDF: Es un formato de almacenamiento de documentos digitales independiente de plataformas de software o hardware. Este formato es de tipo compuesto (imagen vectorial, mapa de bits y texto).

PHP: Es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.

PKCS#11: Define una Interfaz de programación de aplicaciones (API) genérica de acceso a dispositivos criptográficos.

R

RC2: Es un cifrado de bloques de claves simétricas.

S

S/MIME: Es un estándar para criptografía de clave pública y firmado de correo electrónico encapsulado en MIME. Este provee los servicios de seguridad criptográfica de autenticación, integridad y no repudio (mediante el uso de firma digital), privacidad y seguridad de los datos (mediante el uso de cifrado) para aplicaciones de mensajería electrónica.

SSL: Es un protocolo criptográfico que proporciona comunicaciones seguras por una red, comúnmente Internet. Permite la autenticación y privacidad de la información entre extremos sobre Internet mediante el uso de criptografía.

T

TLS: Es un protocolo criptográfico empleado para permitir comunicaciones seguras por una red, comúnmente Internet.

V

VPN: Es una tecnología de red que permite una extensión segura de la red local (LAN) sobre una red pública o no controlada como Internet. Permite que la computadora en la red envíe y reciba datos sobre redes compartidas o públicas como si fuera una red privada con toda la funcionalidad, seguridad y políticas de gestión de una red privada.

X

XAdES: Es un conjunto de extensiones a las recomendaciones XMLDSig haciéndolas adecuadas para la firma electrónica avanzada permitiendo que los documentos firmados electrónicamente puedan seguir siendo válidos durante largos períodos.

XML: Es un lenguaje de marcas utilizado para almacenar datos en forma legible. Además da soporte a bases de datos, siendo útil cuando varias aplicaciones se deben comunicar entre sí o integrar información.

XMLDsig: Define una sintaxis XML para la firma digital. Funcionalmente, tiene mucho en común con PKCS#7, pero es más extensible y está orientada hacia la firma de documentos XML.

X500: Es un conjunto de estándares de redes de ordenadores de la ITU-T sobre servicios de directorio, entendidos estos como bases de datos de direcciones electrónicas (o de otros tipos).

Anexos

Anexo 1: Interfaces de usuario.

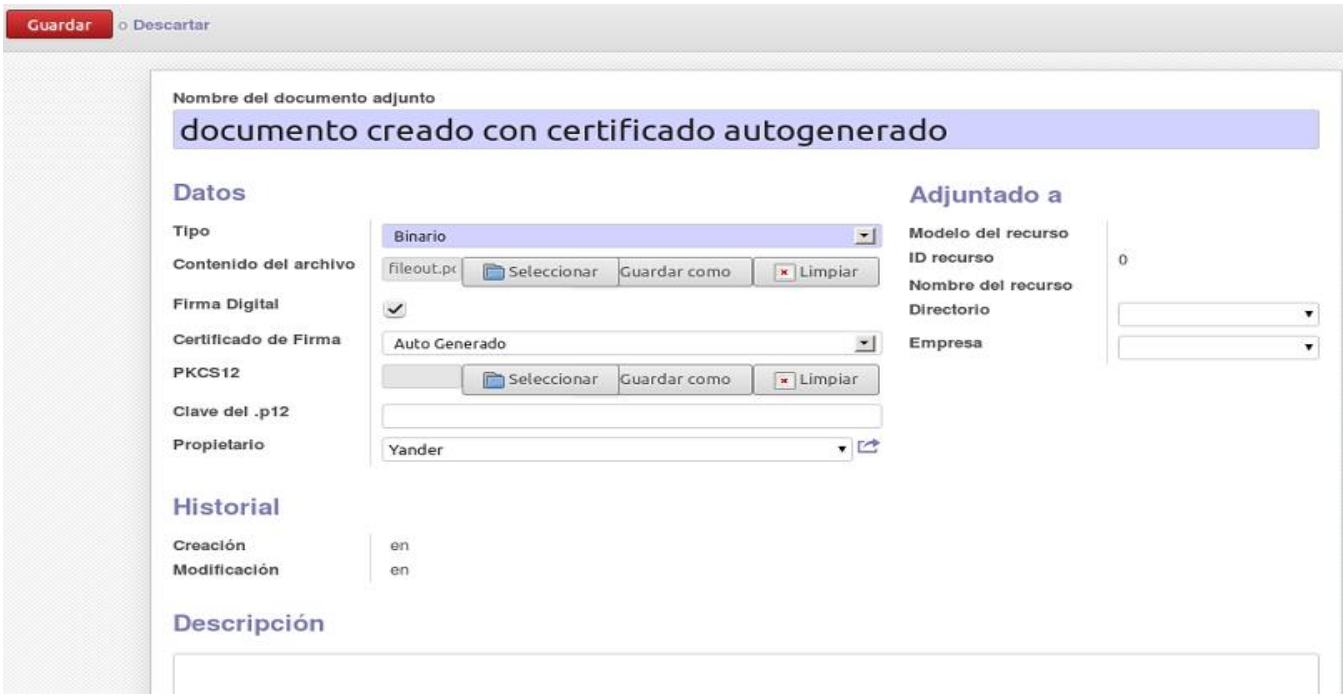


Fig. 10 Interfaz de usuario “Firmar documento con certificado autogenerado”.

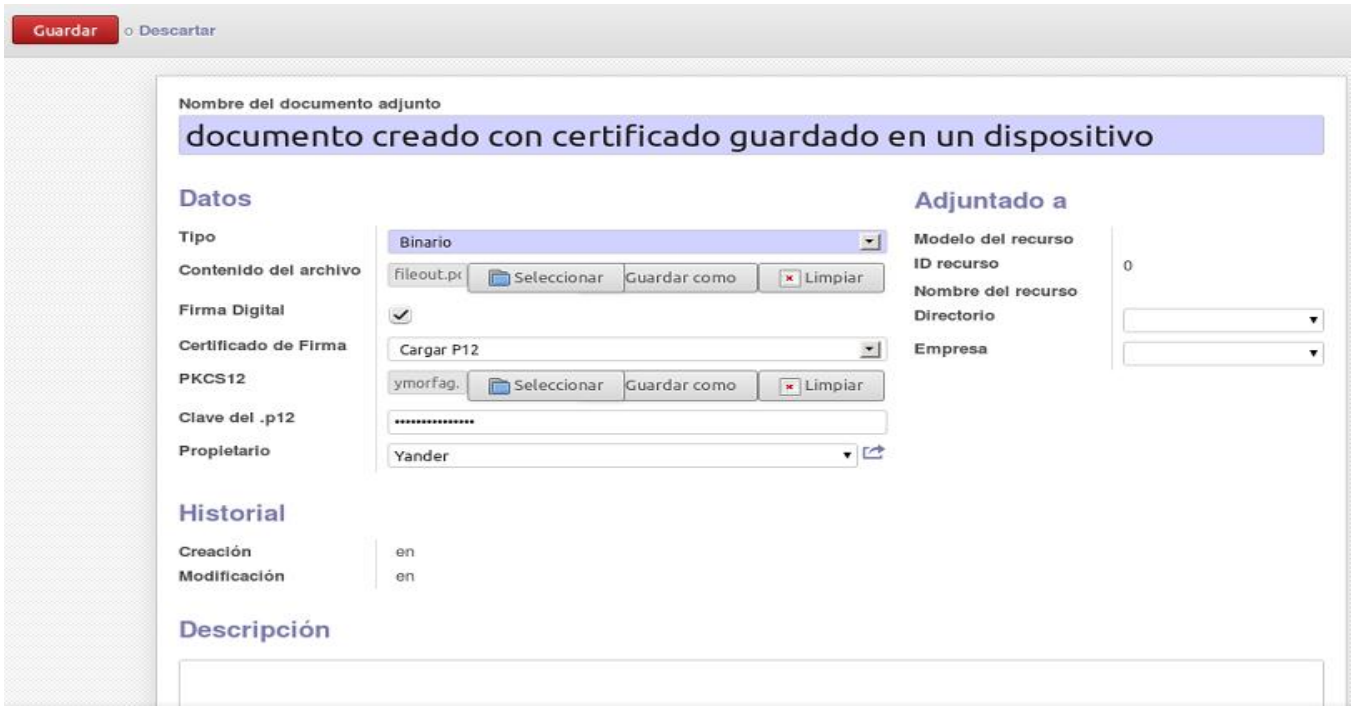


Fig. 11 Interfaz de usuario “Firmar documento con un certificado .P12 guardado en un dispositivo de almacenamiento”.

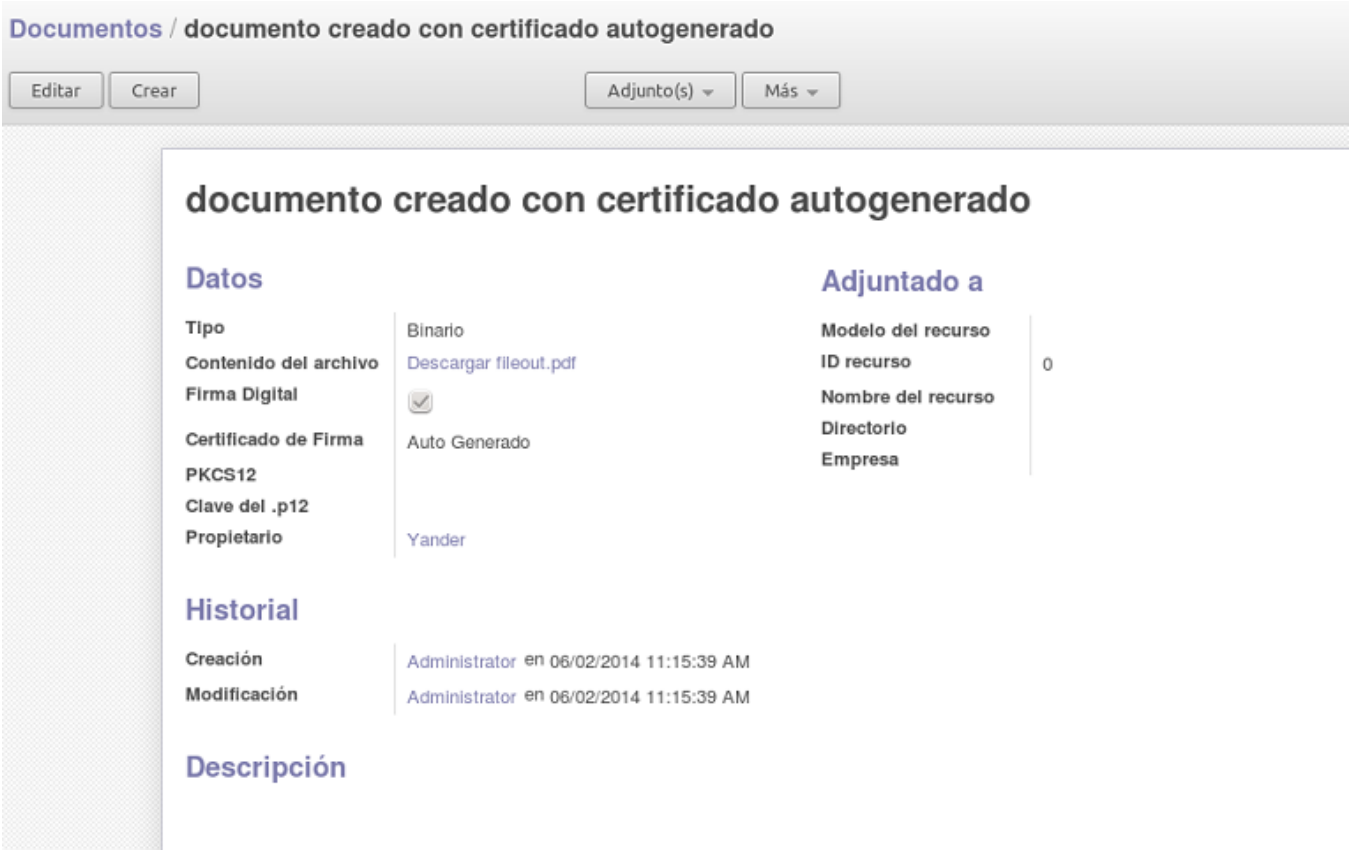


Fig. 12 Interfaz de usuario “Visualizar información de un documento creado”.