Universidad de las Ciencias Informáticas

Facultad 3

Título: Herramienta de firma digital.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Rafael Alejandro Rodríguez Fernández

Tutor: Ing. Vlamir Rodríguez Fernández

Declaración de Autoría:

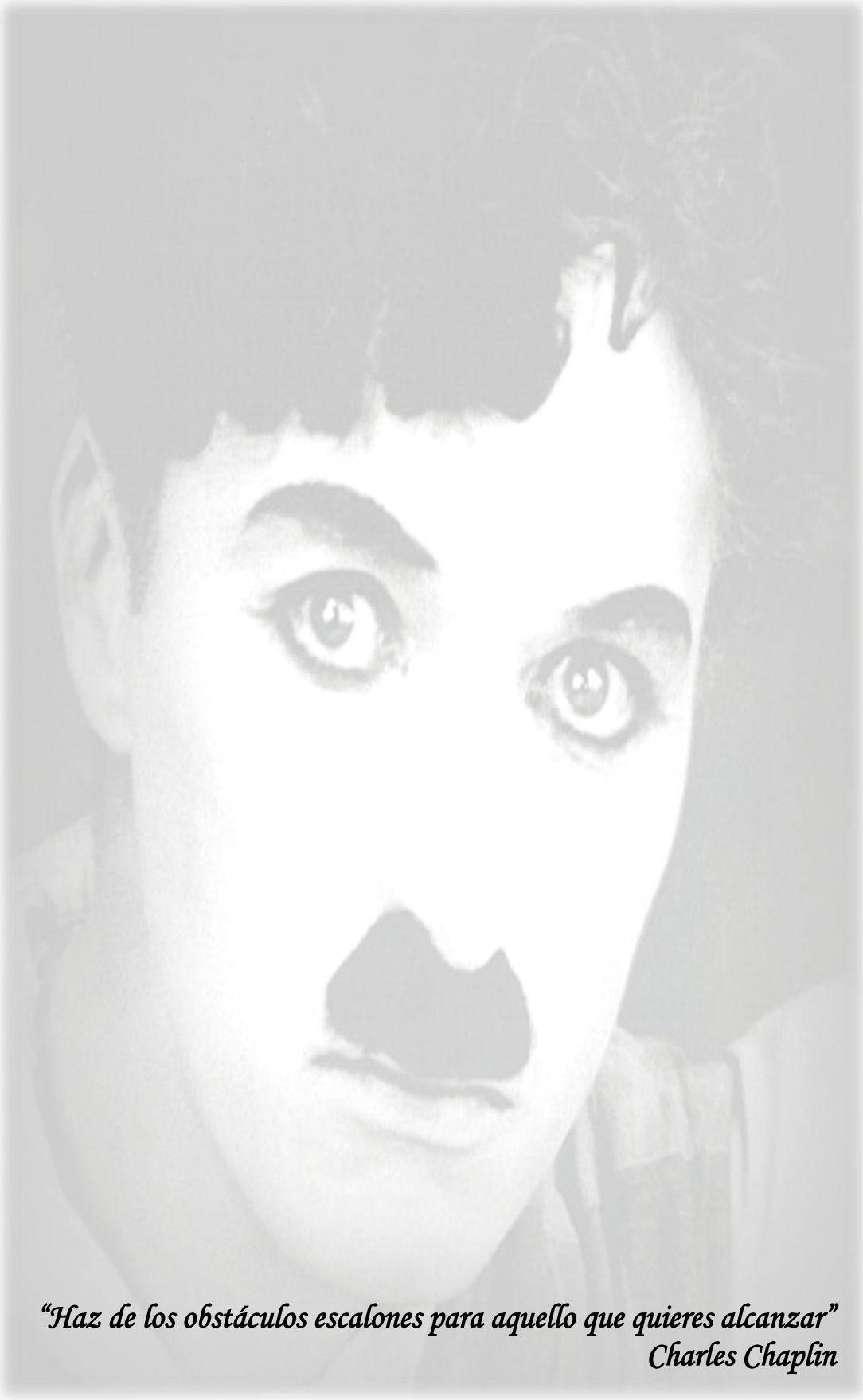
Declaro ser el único autor del presente trabajo de diploma y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

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

Rafael Alejandro Rodríguez Fernández *Vlamir Rodríguez Fernández*

_____ _____

Autor *Tutor*



"Haz de los obstáculos escalones para aquello que quieres alcanzar"
Charles Chaplin

Agradecimientos:

A mis padres por ser los mejores del mundo y estar siempre que los necesité, por brindarme su apoyo y su confianza, por preocuparse siempre por mí y porque sin ustedes no estuviera aquí.

A mi hermano por ser un ejemplo a seguir, por ser mi hermano y mi tutor, por ser mi guía y mi apoyo desde que entré a la universidad, por estar para mí siempre que lo necesité, por aguantarme todos estos años, por estar siempre dispuesto a ayudarme y porque sin su ayuda no hubiese sido posible terminar este trabajo.

A toda mi familia, mis abuelos, mis tíos, mis primos por su preocupación constante y por el apoyo brindado.

A mi cuñada Lianet que me ha ayudado mucho en estos últimos 3 años de la carrera.

A los otros dos mosqueteros Dayron y Dannier, y a D´Artagnan (Jose), que han sido mis hermanos en estos cinco años, han estado en las buenas y malas.

A todos mis compañeros de aula por apoyarme durante este tiempo en la universidad.

A mis amigos, los que comenzamos desde el primer año, los que vinieron después, los de siempre, gracias por estar ahí.

A todos los profesores que contribuyeron en mi formación como profesional.

A todos muchas gracias.

Dedicatoria:

A mis padres que son mi inspiración y mis guías durante toda mi vida, les dedico este sueño hecho realidad.

A mi hermano por ser sencillamente eso "mi hermano".

A todos los que confiaron en mí de una forma u otra y sabían que era posible que llegara hasta aquí.

Resumen

El presente trabajo tiene como objetivo desarrollar una herramienta para el Centro de Gobierno Electrónico (CEGEL) que permita 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. Para esto se hace un estudio de la firma digital como un mecanismo de seguridad para brindar valor probatorio a los documentos. Se describen las herramientas, tecnologías y artefactos generados en el desarrollo de la solución.

Índice

Índice de tablas

Índice de figuras

INTRODUCCIÓN

Desde la antigüedad los individuos se vieron en la necesidad de tener algún método, forma o mecanismo para reconocer que eran ellos los autores de una obra o identificarse como propietarios sobre algún objeto, material, artefacto e incluso lugar; ejemplo de ello son las pinturas utilizadas por nuestros primeros antepasados en las paredes de las cuevas para dejar constancia de su paso por dicho sitios, o en la antigua Roma con la manufirmatio[1]. En la Edad Media, se asentaba una cruz a la que se le sumaban diferentes letras y rasgos, siendo esto utilizado como firma. Posteriormente la implantación del sello real constituyó la vía para garantizar la autenticidad de los documentos, años más tarde las clases nobles y privilegiadas comenzaron a utilizar los sellos como mecanismo de firma.

A partir de esta necesidad es que surge la firma como un mecanismo para dar fe de la autenticidad y validez a cualquier documento. “Una firma es todo nombre o símbolo utilizado por una parte con la intención de que constituya su firma”(Comisión de las Naciones Unidas para el Derecho Mercantil Internacional, 2009). Cabe entender que la finalidad de las normas legales que prescriben que un documento concreto sea firmado por una persona concreta es confirmar la autenticidad del documento. El paradigma de la firma es el nombre del firmante, escrito de su propio puño y letra, en un documento de papel (una firma autógrafa[2] o manuscrita[3]). De esta forma la firma autógrafa constituye el mecanismo a través del cual la sociedad ha validado la voluntad del firmante con respecto a un documento, reflejando que está de acuerdo con los términos que en el mismo aparecen y asume todos los derechos y obligaciones que de este se derivan.

La firma autógrafa, a pesar de su aceptación a nivel mundial ha presentado varios desafíos como el de asegurar la autenticidad de la misma, que el documento firmado no ha sido modificado en su contenido (integridad) y el no repudio del autor en relación a la firma y el contenido de dicho documento. Con respecto al primer reto planteado se han creado algunos mecanismos de validación de autoría de la misma para poder

[1] Ceremonia practicada en la Antigua Roma, en la que leído un manuscrito por su autor o discípulo se disponía desenrollado y extendido sobre la mesa del escribano, y se le estampaba nombre, signo, y una o tres cruces por el autor, notario y escribano, después de haber pasado la mano abierta sobre él a modo de juramento, aceptación o reconocimiento. A continuación testigos presenciales hacían lo mismo.

[2] Se aplica al texto que está escrito de la mano de su propio autor.

[3] Que está escrito a mano.

garantizar la autenticidad de estos documentos, como es el uso de peritos calígrafos[4] certificados. En relación a los otros desafíos a los que se enfrenta por la utilización de la firma autógrafa, se torna un poco difícil probar la validez de los mismos, debido a que no existen métodos que permitan demostrar que estos documentos no han sido alterados y que otorguen certeza jurídica respecto a los documentos firmados.

La firma es uno de los requisitos de forma que exigen las legislaciones nacionales a la hora de considerar al documento. Cuando se refiere a documento electrónico, no se está ante una institución jurídica autónoma y diferente, sino que constituye una nueva forma de expresión o exteriorización del pensamiento que se materializa en soporte distinto al tradicional, sólo que el continente[5] ha cambiado manteniéndose intacto el contenido. Por ello el documento electrónico, consta de los mismos elementos que básicamente tiene el tradicional: soporte, medio y contenido. También lleva en sí otros que caracterizan al tradicional, dentro de los cuales puede ubicarse la firma como elemento esencial, que tiene como función fundamental la de identificar al autor del mismo.

El desarrollo continuo de las Tecnologías de la Información y las Comunicaciones (TIC) y su inclusión en sectores de la sociedad, como la economía y el ámbito legal unido a los elementos anteriormente planteados ha dado paso a la búsqueda de nuevas formas de darle valor probatorio a los trámites que se realizan de forma electrónica. En las tecnologías de la información y la informática se han puesto a punto diversos medios de vincular la información en forma electrónica a personas o entidades concretas, con objeto de garantizar la integridad de dicha información o de permitir que las personas demuestren su derecho o autorización para obtener acceso a un determinado servicio o depósito de información. Estas funciones suelen denominarse genéricamente métodos de firma electrónica.

En el transcurso del tiempo se han creado distintas técnicas de firma electrónica. Cada una de ellas tiene por objetivo atender a distintas necesidades y proporcionar distintos niveles de seguridad y también diferentes requisitos técnicos. Los métodos de autenticación utilizados para la firma electrónica pueden clasificarse en tres categorías: los que se basan en lo que el usuario o el receptor sabe (por ejemplo, contraseñas, números de identificación personal), los basados en las características

[4] Es un especialista encargado de asesorar a un juez sobre la autenticidad de un documento determinado.

[5] Cosa que contiene en sí a otra.(Real Academia Española, 2001)

físicas del usuario (por ejemplo, biométrica[6]) y los que se fundamentan en la posesión de un objeto por el usuario (por ejemplo, códigos u otra información almacenados en una tarjeta magnética).

Dentro de la firma electrónica se encuentra la firma electrónica avanzada o firma digital, que es el resultado de aplicar a un documento digital un procedimiento matemático que requiere información de exclusivo conocimiento del firmante, encontrándose ésta bajo su control. La firma digital debe ser susceptible de verificación por terceras partes, tal que dicha verificación simultáneamente permita identificar al firmante y detectar cualquier modificación del documento digital, después de haberse realizado el acto de firma.

En la actualidad los medios digitales están expuestos a cambios, reemplazo y replicación, a menos que estén especialmente protegidos con el objetivo de que se pueda confiar en las comunicaciones a través de la red. Es por ello que uno de los sistemas definidos para brindar seguridad a los documentos electrónicos es la Infraestructura de Clave Pública (Public Key Infrastructure, por sus siglas en inglés PKI), que es un sistema complejo para gestionar los certificados digitales y las aplicaciones de firma digital, basado en dos claves: una pública y una privada. La primera puede ser del conocimiento de cualquier persona, mientras que la segunda solo puede saberla el propietario. PKI tiene como objetivos lograr la autenticación de los usuarios, el no repudio por parte de los autores, la integridad de la información, la auditabilidad y el acuerdo de claves secretas. Este sistema engloba una serie de componentes como son la autoridad certificadora, publicación de certificados, soporte de clave privada, política de certificación y aplicaciones de PKI-Enabled[7].

En la Universidad de las Ciencias Informáticas (UCI) existen soluciones para brindar valor probatorio a los documentos digitales, como es el caso de un plugin de Alfresco, que permite realizar la firma digital de los documentos gestionados en esa entidad, también en la UCI se han desarrollado sistemas de identificación y seguridad con el uso de la tecnología de tarjetas inteligentes en pasaportes electrónicos. En algunos de los proyectos que pertenecen al Centro de Gobierno Electrónico (CEGEL) se han desarrollado soluciones para dar valor probatorio a los documentos electrónicos, entre ellas se pueden citar un componente de software para la firma de documentos jurídicos en formato electrónico desarrollado en el proyecto Tribunales y un

[6] La biometría es el estudio de métodos automáticos para el reconocimiento único de humanos basados en uno o más rasgos conductuales o físicos intrínsecos.
[7] Aplicaciones software capaces de operar con certificados digitales.

componente de firma digital para garantizar la autenticidad e integridad de la información generada en la Fiscalía General de la República de Cuba, implementado en el proyecto Sistema de Informatización de la Gestión de la Fiscalía. La solución para las oficinas de Registro y Notarías de la República Bolivariana de Venezuela realizada por el proyecto Registro y Notarías y la solución desarrollada por el proyecto Registro de Antecedentes Penales (RAP). Cabe resaltar que todos son componentes o módulos de proyectos específicos y no una herramienta independiente capaz de ser utilizada en cualquier proyecto.

Precisamente el no poder asegurar la autenticidad, no repudio e integridad de los documentos electrónicos, está afectando los sistemas que se desarrollan en el centro CEGEL, que por el entorno de Gobierno Electrónico al cual tributan, es muy necesario poder dar valor probatorio a los documentos digitales generados.

Partiendo de lo expuesto anteriormente la investigación se propone resolver el siguiente **problema:** en la generación de los documentos digitales por los diferentes sistemas desarrollados en el centro CEGEL, no se brinda la posibilidad de proporcionar valor probatorio a los mismos.

Teniendo como **objeto de estudio** el proceso de desarrollo de software.

Para resolver el problema planteado en el trabajo de diploma, se propone como **objetivo general:** desarrollar una herramienta de firma digital, que contribuya a proporcionar valor probatorio a documentos digitales generados por los sistemas desarrollados en el centro CEGEL.

Enfocado en el **campo de acción:** proceso de desarrollo de software de los sistemas de firma digital para los documentos digitales.

Planteando como **idea a defender:** si se utiliza una herramienta de firma digital entonces se podrá facilitar dar valor probatorio a los documentos digitales generados por los sistemas que se desarrollan en el centro CEGEL.

Con la finalidad de dar cumplimiento al objetivo trazado se proponen los siguientes **objetivos específicos:**

- Elaborar el marco teórico de la investigación.
- Elaborar la propuesta de solución.
- Validar los artefactos generados en la propuesta.

Teniendo como **tareas de la investigación:**

- Recopilación de la información necesaria para la investigación.
- Selección de la información relevante para la investigación.
- Estudio de las metodologías de desarrollo, las herramientas case y lenguajes del modelado.
- Identificación y especificación de requisitos.
- Realización de las historias de usuario.
- Confección del plan de iteraciones.
- Confección del plan de entregas.
- Realización de las tarjetas CRC.
- Refinamiento del diseño utilizando patrones.
- Definición del estándar de codificación para la implementación.
- Descripción de las tareas de programación por iteraciones.
- Implementación de los componentes necesarios que dan solución al objetivo propuesto.
- Confección de los casos de pruebas de aceptación.
- Confección de los casos de pruebas unitarias.
- Validación de los resultados obtenidos.

Métodos

Para el desarrollo de la investigación se utilizaron diferentes métodos con el objetivo de proveer a la investigación de bases fundamentadas para su exitosa evolución.

Los **métodos de investigación teóricos** permiten estudiar las características del objeto de investigación que no son observables directamente, crean todas las condiciones para descubrir las características del objeto de estudio mucho más allá de la superficialidad de la realidad.

Histórico-lógico: permite estudiar de forma analítica la trayectoria histórica real de los fenómenos, su evolución y desarrollo.

- Usado con el objetivo de estudiar la evolución de la firma digital, dispositivos criptográficos, estándares de criptografía y documentos electrónicos.

Analítico-sintético: facilita el entendimiento del fenómeno en el que se trabaja, es más útil la división de este en diferentes fases y de esta forma descubrir sus características generales, lo que ayuda a seguir una correcta investigación.

- A partir de la información obtenida se hace necesario organizarla y sintetizarla para elaborar una estructura adecuada.

Por otra parte, los **métodos empíricos:** describen y explican las características fenomenológicas del objeto, representan un nivel de la investigación cuyo contenido procede de la experiencia y es sometido a cierta elaboración racional.

Entrevista: es una conversación planificada entre el investigador y el entrevistado para obtener información.

- Tiene como objetivo obtener la mayor información posible sobre los requisitos que el cliente desea.

La investigación está estructurada de la siguiente forma:

Capítulo 1. Fundamentación Teórica: se realiza un estudio de los aspectos fundamentales relacionados con la firma digital, así como un estudio de otras herramientas existentes para la firma digital. Además se seleccionan las herramientas y tecnologías para desarrollar la aplicación.

Capítulo 2. Propuesta de Solución: 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 con los que debe cumplir el sistema.

Capítulo 3. Diseño y Desarrollo: se elaboran las tarjetas CRC para guiar la construcción de la solución propuesta, así como otros artefactos propuestos para las fases de diseño y desarrollo.

Capítulo 4. Validación de la propuesta de solución: se muestra el conjunto de pruebas realizadas a la herramienta que demuestran la calidad obtenida en su desarrollo.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1. INTRODUCCIÓN

El presente capítulo tiene como objetivo brindar el marco teórico en el cual se desarrolla el presente trabajo. Se establecen conceptos y definiciones que dan fundamento al uso de las herramientas, metodología y lenguaje que se emplean en la implementación de la solución que se desarrolla en los capítulos siguientes, así como ayudan en la compresión de las bases teóricas de la firma digital como mecanismo para brindar autenticidad a los documentos electrónicos. Por último se hace un estudio sobre otras herramientas de firma digital existentes.

1.2. FIRMA DIGITAL

1.2.1. DEFINICIÓN DE FIRMA DIGITAL

El concepto de firma digital fue introducido por Diffie y Hellman en 1976. Este es descrito por Maricarmen Pascale en el artículo que presentó en el VIII Congreso Iberoamericano de Derecho e Informática de Ciudad de México en el año 2000: “firma digital es una secuencia de caracteres alfanuméricos que contiene los elementos que identifican al remitente aplicando una clave privada”(Pascale, 2000), garantizando así la integridad, autenticidad y el no repudio del mensaje.

Durante la investigación se asumirá que los términos firma electrónica avanzada y firma digital son equivalentes.

1.2.2. ORÍGENES

El origen de la firma digital data de mediados de la década del setenta, cuando Whitfield Diffie y Martin Hellman, encontraron el modo de asegurar la autenticidad y confidencialidad de la información digital mediante la aplicación de fórmulas matemáticas. Dicha técnica es conocida con el nombre de criptografía asimétrica.

1.2.3. CARACTERÍSTICAS DE LA FIRMA DIGITAL

Dentro de las características fundamentales de la firma digital se encuentran las descritas en el artículo “Tipología legal de la firma electrónica en la Unión Europea” del autor Ignacio Alamillo Domingo (Domingo, 2002), entre ellas se encuentran:

- La utilización de algoritmos criptográficos asimétricos[8], con claves diferentes para el proceso de cifrado y descifrado.
- Integridad: si un documento ha sido firmado y su contenido resulta modificado por un tercero, las verificaciones de seguridad corroborarán que el documento resultante no es idéntico al original. De esta forma se evitan posibles suplantaciones o manipulaciones.
- No repudio: un documento firmado identifica al emisor y el contenido, con lo que se evita la posibilidad de rechazar la propiedad de un documento o el haberlo enviado. El emisor no puede negar bajo ninguna circunstancia que ha generado dicho mensaje.
- Autenticidad: garantiza que el mensaje ha sido generado por la parte identificada en el documento como emisor del mismo, no pudiendo alguna otra entidad suplantar a un usuario del sistema.

1.2.4. IMPORTANCIA DE LA FIRMA DIGITAL

El desarrollo de las comunicaciones y las tecnologías ha aumentado el número de transacciones en línea, además de permitir el ingreso a sistemas abiertos de comunicación como Internet, sumando a esto la tendencia mundial a la agilización de trámites y operaciones, la firma digital ocupa un papel protagónico innegable, siendo el principal mecanismo para brindar seguridad y confianza en las redes abiertas. Además la firma digital constituye una solución que goza de la aceptación mundial ante la demanda de un mundo informatizado.

Gracias a la capacidad de adaptarse con gran facilidad y rapidez, la firma digital actualmente es utilizada para todo tipo de información, tanto texto, sonido como imágenes, mostrando sus beneficios tanto en el área del comercio electrónico como en el ámbito legal. Cabe destacar también que la firma digital es el único medio tecnológico que permite con total seguridad emplear los medios de comunicación para el intercambio de información con fines sociales.

El uso de la firma digital reviste gran importancia ya que ofrece la seguridad de que el autor no puede negar que un documento fue hecho por él. También puede ser detectada cualquier modificación realizada a un documento que ha sido firmado. La firma digital es de gran utilidad para lograr validar la identidad del firmante y asegurar

[8] Es un algoritmo que modifica los datos de un documento con el objeto de alcanzar algunas características de seguridad como autenticación, integridad y confidencialidad, que utiliza una clave para encriptar y otra para desencriptar.

que es quien dice ser. Además es muy improbable que sea falsificada, a diferencia del caso de la firma autógrafa.

En la tabla se muestra una comparación entre los tipos de firma existentes, en relación a los elementos de seguridad con los que puede contar un documento.

	Firma Autógrafa	Firma Electrónica Simple	Firma Electrónica Avanzada
Elementos formales			
La firma como signo personal.	**x**	**x**	**x**
El animus signandi, voluntad de asumir el contenido de un documento.	**x**	**x**	**x**
Elementos funcionales			
Identificación	**x**	**x**	**x**
Autentificación	**x**	**x**	**x**
Confidencialidad		**x**	**x**
Integridad			**x**
No repudio			**x**

Tabla 1. Tabla comparativa entre los distintos tipos de firmas. **(Zúñiga, 2011)**

1.3. Infraestructura de Clave Pública

Una infraestructura de clave pública es un sistema complejo para gestionar los certificados digitales y las aplicaciones de firma digital. Tiene como objetivo certificar la autenticidad de las claves públicas, confirmar la identidad de los usuarios y comprobar la validez de los certificados. Las PKI proporcionan a una organización los mecanismos, tecnología, procedimientos y políticas que otorgan autenticación, confidencialidad, integridad y no repudio para los trámites que realizan de forma electrónica.

1.3.1. CARACTERÍSTICAS DE UNA PKI

Entre las principales características de una PKI se encuentran:

- Establece los mecanismos de generación, distribución y revocación de claves criptográficas asimétricas[9].
- Generación de certificados: se habilita la posibilidad de que los dos extremos de una comunicación se identifiquen y evitar así el repudio.
- Registro de certificados: los certificados emitidos deben estar registrados para poder comprobar que un firmante es quien dice ser.
- Almacenamiento y distribución de certificados: los certificados deben estar almacenados en un repositorio.
- Revocación y renovación de certificados.

1.3.2. ENTIDADES DE UNA PKI

Entidades que conforman la estructura de una PKI:

- Las Autoridades Certificadoras (CA) son las encargadas de certificar la autenticidad de las claves públicas, emiten los certificados e interoperan con otras CA.
- Las Autoridades Registradoras (RA) son las encargadas de verificar la identidad del usuario, generan las solicitudes de certificados para las CA, verifican las revocaciones de los certificados y las relaciones entre claves privadas y claves públicas.
- Las Autoridades de Validación (VA) verifican los certificados online y mantienen la lista de certificados revocados (CRL).
- Los Servicios de Directorios almacenan los certificados emitidos por las CA y permiten el acceso a los certificados a cualquiera que los necesite.

1.4. ARQUITECTURA A UTILIZAR

La arquitectura propuesta para desarrollar la herramienta de firma digital es la arquitectura basada en N-capas, debido a que se ajusta a las necesidades de la solución que se desea construir. Esta arquitectura tiene como objetivo principal separar los diferentes aspectos del desarrollo, tales como las cuestiones de

[9] Clave púbica que interviene en el proceso de criptografía asimétrica, es conocida por todos aquellos que les incumbe el mensaje.

presentación, lógica de negocio, mecanismos de almacenamiento. Se caracteriza fundamentalmente porque cada capa es completamente independiente de las otras capas, excepto aquella que está inmediatamente debajo de ella. La capa n solo necesita saber cómo manejar una solicitud de la capa n+1, cómo hacer la solicitud a la capa n-1 (si existe) y cómo manejar el resultado de la petición. La arquitectura de N-capas tiene al menos tres capas separadas o partes, cada una de ellas con su responsabilidad. Mejora escalabilidad, disponibilidad, administración y utilización de recursos. Dentro de los principales beneficios que aporta esta arquitectura están:

• Mejoras en las posibilidades de mantenimiento: debido a que cada capa es independiente de la otra, los cambios o actualizaciones pueden ser realizados sin afectar la aplicación como un todo.

• Flexibilidad: como cada capa puede ser manejada y escalada de forma independiente, la flexibilidad se incrementa.

• Disponibilidad: las aplicaciones pueden aprovechar la arquitectura modular de los sistemas habilitados usando componentes que escalan fácilmente, lo que incrementa la disponibilidad.

1.5. METODOLOGÍAS DE DESARROLLO DE SOFTWARE

A la hora de desarrollar un producto de software es necesario aplicar una metodología que guie el proceso de desarrollo de dicho software para que reúna todos los requisitos y calidad esperados por el cliente.

Según Mario Piattini una metodología de desarrollo de software es “un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software”(Piattini, 2007). Una metodología es la que define Quién debe hacer Qué, Cuándo y Cómo.

Las necesidades principales que persiguen las metodologías son:

- Mejores aplicaciones, tendientes a una mejor calidad, aunque a veces no es suficiente.
- Un proceso de desarrollo controlado, que asegure uso de recursos apropiados y costo adecuado.
- Un proceso estándar en la organización, que no sienta los cambios del personal.

Objetivos de las metodologías de software:

- Brindar un método sistemático, de modo que se controle el progreso del desarrollo.
- Especificar los requerimientos de un software en forma apropiada.
- Construir productos bien documentados y de fácil mantenimiento.
- Ayudar a identificar las necesidades de cambio lo más pronto posible.
- Proporcionar un sistema ágil que satisfaga a todas las personas involucradas.

Seleccionar una metodología apropiada, resulta en ocasiones un factor esencial para obtener un software con la calidad esperada. Actualmente no existe una metodología global que se le pueda aplicar a todos los proyectos y su selección depende de las características del proyecto.

Existen dos grandes grupos o corrientes que agrupan las metodologías de desarrollo del software, las metodologías tradicionales o pesadas y las metodologías ágiles o ligeras. “Las primeras están pensadas para el uso exhaustivo de documentación durante todo el ciclo del proyecto mientras que las segundas ponen vital importancia en la capacidad de respuesta a los cambios, la confianza en las habilidades del equipo y al mantener una buena relación con el cliente” (Figueroa, y otros, 2009).

A continuación se muestran las características de las principales metodologías como parte del estudio realizado para seleccionar la metodología que más se ajusta a las necesidades específicas de este producto.

1.5.1. METODOLOGÍAS PESADAS

Según Karenny Brito en su artículo Selección de Metodologías de Desarrollo para Aplicaciones Web en la Facultad de Informática de la Universidad de Cienfuegos del año 2009 plantea que “las metodologías pesadas son las más tradicionales, se centran en la definición detallada de los procesos y tareas a realizar, herramientas a utilizar, y requiere una extensa documentación, ya que pretende prever todo de antemano.” (Acuña, 2009).

Las metodologías pesadas están orientadas al control de los procesos, determinando estrictamente las actividades que se van a desarrollar, así como las herramientas y notaciones que se utilizarán durante todo el proceso de desarrollo del software. Estas metodologías dan mucha importancia al plan de proyecto, en tener todo bien

especificado antes de empezar el proceso de desarrollo del software, en seguir exactamente el camino trazado y en documentar íntegramente todo lo realizado.

1.5.1.1. RUP

Rational Unified Process (RUP) “es una infraestructura flexible de desarrollo de software que proporciona prácticas recomendadas probadas y una arquitectura configurable. Es un Proceso Práctico”(Grupo Soluciones Innnova S.A., 2007).

RUP permite seleccionar de una manera muy sencilla el grupo de componentes de procesos que concuerdan con las necesidades específicas del proyecto. Al unificar el equipo de desarrollo con los procesos comunes para crear un entendimiento común para todas las tareas, responsabilidades y artefactos, permitirá lograr los resultados esperados, o sea, obtener un software de calidad, en el tiempo esperado y dentro del coste planificado.

Rational Unified Process, es una plataforma flexible de procesos de desarrollo de software que ayuda brindando guías consistentes y personalizadas de procesos para todo el equipo de proyecto (Acuña, 2009).

RUP estructura los proyectos en términos de disciplinas y fases, estando conformada a su vez cada una de estas por una o más iteraciones. Con esta aproximación iterativa, el énfasis de cada flujo de trabajo (workflow) se modificará a través del ciclo de vida. La aproximación iterativa ayuda a minimizar los riesgos en forma prematura y continua, con un progreso que se puede observar a través de continuas liberaciones (releases) ejecutables.

Características del ciclo de vida de RUP:

- Dirigido por casos de uso.
- Centrado en la arquitectura.
- Iterativo e incremental.

Elementos fundamentales que componen RUP:

Trabajadores (quién): define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.

Actividades (cómo): es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.

Artefactos (qué): productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.

Flujo de actividades (cuándo): secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

El ciclo de vida de RUP está dividido en cuatro fases:

- Inicio: en esta fase se define la visión del proyecto, objetivos y alcance del proyecto. Tiene como hito los **Objetivos (visión)**.
- Elaboración: determina la arquitectura óptima del proyecto, así como se concluye el análisis de los casos de uso. Presenta como hito la **Arquitectura**.
- Construcción: obtiene la capacidad operacional inicial. Posee como hito la **Funcionalidad operativa.**
- Transición: obtiene el producto acabado y definido. Tiene como hito un **Release del sistema.**

Dentro de cada iteración se agrupan las actividades o tareas en nueve flujos de trabajo o disciplinas.

Flujos de trabajo de ingeniería:

- Modelamiento del negocio.
- Requisitos.
- Análisis y diseño.
- Implementación.
- Prueba.
- Despliegue.

Flujos de trabajo de apoyo:

- Administración de configuración y cambios.
- Administración del proyecto.
- Ambiente.

1.5.2. METODOLOGÍAS ÁGILES

Las metodologías ágiles hacen énfasis en desarrollar métodos de trabajo flexibles que permitan adaptarse al cambio, a ir trabajando sobre la marcha, precisando el camino según va avanzando el proyecto. También se encargan de valorar al individuo y a las iteraciones del equipo más que a las herramientas o procesos usados. Las metodologías ágiles tienen mucho más en cuenta crear un buen producto que generar mucha documentación, se centran en alcanzar resultados que satisfagan al cliente, adaptándose a sus cambiantes necesidades. Además promueve iteraciones a lo largo de todo el ciclo de vida de un proyecto y minimiza riesgos desarrollando productos en cortos plazos de tiempo.

1.5.2.1. SCRUM

Scrum es una metodología ágil de desarrollo de proyectos desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Esta metodología es de carácter adaptable, donde la gestión no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto. Esta metodología está dirigida principalmente a las personas y no a los procesos, además de explotar un desarrollo ágil, iterativo e incremental.

Scrum especifica un grupo de roles y prácticas que definen el proceso de desarrollo que se ejecutará en un proyecto. Los roles definidos por esta metodología son: propietario del producto, maestro Scrum y el equipo. Otros elementos de importancia en esta metodología son las reuniones y los documentos, dentro de las reuniones existentes están la reunión diaria de Scrum, reunión de planificación del sprint, reunión de revisión del sprint y seguimiento del sprint, mientras que entre los documentos que se generan se encuentran la pila del producto, pila del sprint y el incremento (Schwaber, y otros, 2011).

Dentro de los beneficios que aporta Scrum se tiene:

- Entrega mensual (o quincenal) de resultados.
- Productividad y calidad.
- Alineamiento entre el cliente y el equipo de desarrollo.
- Equipo motivado.

1.5.2.2. XP

Programación Extrema (eXtreme Programming, por sus siglas en inglés XP) es una metodología de desarrollo del software creada por Kent Beck. Esta metodología se basa en la comunicación, la claridad y la reutilización continua de código. Tiene como objetivos fundamentales la satisfacción del cliente y promover el trabajo en equipo.

Características de XP (Beck, 1999):

- XP es una metodología "liviana" que no tiene en cuenta la utilización de elaborados casos de uso, la exhaustiva definición de requerimientos y la generación de una extensa documentación.
- XP tiene asociado un ciclo de vida y es considerado a su vez un proceso.
- La tendencia de entregar software en espacios de tiempo cada vez más pequeños con exigencias de costos reducidos y altos estándares de calidad.
- XP define Historias de Usuario como base del software a desarrollar, estas historias las escribe el cliente y describen escenarios sobre el funcionamiento del programa, a partir de las historias de usuario y de la arquitectura perseguida se crea un plan de liberaciones entre el equipo de desarrollo y el cliente.

XP consta de cuatro fases:

- Planificación.
- Diseño.
- Desarrollo.
- Pruebas.

Los roles de XP propuestos por Kent Beck(Beck, 2002) son:

- Programador: el programador escribe las pruebas unitarias y produce el código del sistema.
- Cliente: escribe las historias de usuario y las pruebas funcionales para validar su implementación. Asigna la prioridad a las historias de usuario y determina cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.
- Encargado de pruebas (Tester): ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

- Encargado de seguimiento (Tracker): proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.
- Entrenador (Coach). es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
- Consultor: es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto en el que puedan surgir problemas.
- Gestor (Big boss): es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

El ciclo de desarrollo consiste en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Vuelve al paso 1.

Luego de un estudio sobre estas metodologías de desarrollo, analizando las características de cada una, se determina que la opción más factible es XP. Esta metodología se ajusta a las necesidades de la solución a implementar: el tamaño del equipo de desarrollo, en este caso una persona; necesidad de versiones funcionales de la solución a corto plazo; dificultad para un equipo de desarrollo pequeño para adoptar una metodología robusta a causa de la cantidad documentación generada.

1.6. HERRAMIENTAS DE DESARROLLO DEL SOFTWARE

1.6.1. VISUAL PARADIGM

Visual Paradigm es una herramienta que emplea UML (Unified Modeling Language) como lenguaje de modelado principal. Está diseñado para soportar el ciclo de vida completo del desarrollo de software permitiendo la captura de requisitos, análisis, diseño e implementación. Realiza una actualización automática del modelo de diseño y código permitiendo mantener la documentación de ambos modelos actualizados con

los cambios que ocurran en ambos sentidos, optimizando la descripción textual de elementos de código a partir de la descripción visual. Presenta una interfaz con grandes facilidades a la hora de modelar los diagramas de clases, así como generar código fuente a partir de estos diagramas mediante ingeniería inversa. También apoya los estándares más recientes de las notaciones de Java y de UML. Esta potente herramienta ayuda a construir mejores aplicaciones en un menor espacio de tiempo y con menores costos (Visual Paradigm International).

Otras características de Visual Paradigm (Visual Paradigm International):

- Fácil de instalar y actualizar.
- Varios idiomas.
- Generación de código para Java y exportación como HTML (HyperText Markup Language).
- Compatibilidad entre ediciones.
- Se integra con NetBeans IDE.

Debido a sus múltiples y notables prestaciones se escoge Visual Paradigm for UML como herramienta CASE (Computer Aided Software Engineering) para el desarrollo de la herramienta de firma digital, además de su capacidad para integrarse con el resto de las herramientas y lenguajes escogidos.

1.6.2. NETBEANS

"NetBeans IDE es un entorno de desarrollo integrado (IDE) modular y basado en estándares" (Oracle Corporation, 2010). Constituye una herramienta para programadores diseñada para escribir, compilar, depurar y ejecutar programas. Es un entorno de desarrollo disponible para varios sistemas operativos como Windows, Mac, Linux y Solaris (Oracle Corporation, 2010). NetBeans consta de un IDE de código abierto con gran diversidad de funciones escrito con el lenguaje de programación Java y una plataforma para aplicaciones de cliente enriquecido que se puede utilizar como marco genérico para crear cualquier tipo de aplicación, tanto empresariales como de escritorio y móviles.

NetBeans IDE 6.9 ofrece mejoras con respecto a versiones anteriores como son:

- JavaFX Composer, como interfaz gráfico para construir aplicaciones RIA[10] mediante la tecnología JavaFX. JavaFX Composer soportará drag&drop[11] de componentes y la posibilidad de establecer una conexión entre los componentes y el modelo de datos.
- Soporte de la plataforma OSGI[12].
- Mejoras en los editores y debuggers Java.
- Regeneración de entidades JPA[13] ante cambios de la base de datos.
- Soporte de Transferencia de Estado Representacional (REST por sus siglas en inglés) para servicios web.
- Corrector ortográfico en el editor.

1.7. LENGUAJES DE PROGRAMACIÓN

Muchas veces la elección de un lenguaje de programación para resolver un problema del mundo real se hace por facilidad, experiencia o simplemente por actualidad, que aunque son razones válidas no garantizan la solución más eficiente ni mucho menos la más barata (recursos). Es importante saber elegir el lenguaje en función de compatibilidad, portabilidad y facilidad para resolver una tarea determinada.

1.7.1. JAVA

Java es un lenguaje de programación desarrollado por Sun Microsystems Inc. a principios de los años 90. Este lenguaje orientado completamente a objetos fue desarrollado con el objetivo de que pudiera funcionar en redes heterogéneas de computadoras y que fuera independiente de la plataforma en la que se ejecutara, lo que representa una gran ventaja para los desarrolladores, ya que no se ven limitados a hacer un programa para cada sistema operativo (Oracle Corporation, 2010).

Dentro de las características fundamentales de Java se encuentran (Oracle Corporation, 2010):

- Simple: elimina la complejidad de otros lenguajes como C y se enfoca en el contexto de los lenguajes orientados a objetos.

[10] Aplicaciones de internet enriquecidas, son aplicaciones web que tienen la mayoría de las características de las aplicaciones de escritorio tradicionales.
[11] Expresión informática que se refiere a la acción de mover con el ratón objetos de una ventana a otra o entre partes de una misma ventana.
[12] Una plataforma modular y ligera para construir aplicaciones basadas en servicios.
[13] Es un framework del lenguaje de programación Java que maneja datos relacionales en aplicaciones usando la Plataforma Java.

- Robusto: maneja la memoria de la computadora para que el programador no se tenga que preocupar por ello, además de realizar verificaciones en busca de errores lo mismo en tiempo de compilación que en tiempo de ejecución.
- Portable: un programa compilado de Java puede ser utilizado por cualquier computadora que tenga implementado el intérprete de Java, ya que su código compilado es interpretado.
- Multiproceso: puede ejecutar diferentes líneas de código al mismo tiempo.
- Dinámico: no es necesario que compile todas las clases de un programa para que este funcione. Al efectuar al menos un cambio en alguna de las clases, Java se encarga de realizar un enlace dinámico o una carga dinámica para encontrar las clases.
- Interpretado: corre en máquina virtual.

Java se ha convertido en la tecnología apropiada para casi todo tipo de aplicaciones gracias a sus características: robustez, sencillez, seguridad, independencia de la arquitectura y otras anteriormente mencionadas. Por estas razones y además permitir desarrollar aplicaciones libres se ha escogido Java como lenguaje de programación para desarrollar esta herramienta de firma digital.

1.8. OTRAS HERRAMIENTAS DE FIRMA DIGITAL

Con el objetivo de brindar los elementos de seguridad necesarios a la información y documentos generados en el ámbito de la gestión documental, así como a los trámites que se ejecutan de forma electrónica, se han desarrollado algunas herramientas que permiten realizar la firma digital, muchas de ellas son propietarias, pero gracias al auge del software libre a nivel mundial también se han desarrollado algunas herramientas libres.

1.8.1. SINADURA DESTOKP

Sinadura es una aplicación de escritorio multiplataforma para firmar digital de archivos en cualquier formato. Está publicado bajo licencia GPL[14]. Entre las características más destacadas de esta herramienta se encuentran (Asociación de Empresas de Software de Euskadi, 2011):

[14] Es una licencia creada por la Free Software Foundation en 1989 (la primera versión, escrita por Richard Stallman), y está orientada principalmente a proteger la libre distribución, modificación y uso de software.

Sistemas operativos soportados:

- GNU/Linux.
- Mac OS X Snow Leopard.
- Microsoft Windows XP, Vista y 7.

Funcionalidades:

- Formatos soportados para firmar:
  - ✓ PKCS11 – Smartcard - Tarjetas (Izenpe, DNIe...).
  - ✓ PKCS12 – Software (Izenpe, DNIe, FNMT ...).
  - ✓ XAdES-X-L.
  - ✓ XAdES-T.
  - ✓ XAdES-BES.
- Firma básica.
- Se puede incluir el sello visible de la firma.
- Personalización de la imagen de fondo de la firma.
- Firma múltiple.
- Firma y validación de cualquier tipo de archivo.
- Módulo de validación para los documentos firmados.
- Almacén de certificados de confianza para el módulo de validación.
- Firma y validación por interfaz gráfica y por línea de comandos.
- Validación de firmas PKCS#7.
- Visor de información de las firmas.
- Envío de correos.
- Interfaz de línea de comandos.
- Idiomas soportados:
  - ✓ Español.
  - ✓ Inglés.

1.8.2. EPARAPHER

La herramienta eParapher para la firma digital soporta casi todo tipo de archivos, firma y convierte el archivo a tres posible estándares: PDF, PDF/A, CMS y XML.

Principales características (eParapher, 2008):

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

1.8.3. ADOBE ACROBAT 9.0

Es la aplicación más completa para edición y creación de documentos en formato PDF. Entre sus funcionalidades se encuentra la firma digital de documentos en formato PDF para brindarle seguridad a dichos documentos. Se caracteriza por ser una aplicación muy robusta y se integra con el almacén de llaves de Windows, a través del cual puede hacer uso de las tarjetas inteligentes. Al ser una aplicación propietaria su costo es elevado.

1.9. PRUEBAS

Pressman plantea que una prueba de software es la ejecución de programas de software con el objetivo de detectar defectos y fallas. Permiten comprobar y mostrar la calidad de un producto de software (Pressman, 2002).

Pensar en utilizar la metodología XP sin tener en cuenta las pruebas no es un camino aconsejable. Estas constituyen una parte imprescindible de esta metodología, que no se debe obviar bajo ningún concepto y que debe aplicarse de manera frecuente y temprana.

La metodología XP divide las pruebas en dos grupos: pruebas unitarias, 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, Kent. 2000).

1.9.1. PRUEBAS DE ACEPTACIÓN

Las pruebas de aceptación son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido implementada correctamente. Las

pruebas de aceptación se realizan con el objetivo de validar que un sistema cumple con el funcionamiento requerido y esperado por el cliente, permitiendo al mismo determinar su aprobación, teniendo en cuenta la funcionalidad y el rendimiento de la aplicación informática. Este tipo de pruebas es considerado como “pruebas de caja negra”, donde cada prueba representa una salida esperada del sistema.

1.9.2. PRUEBAS UNITARIAS

Las pruebas unitarias o también llamadas pruebas de unidad son una de las piedras angulares de la metodología XP. Las mismas aseguran que un único componente de la aplicación produce una salida correcta para una determinada entrada. Este tipo de pruebas validan la forma en la que las funciones y métodos trabajan en cada caso particular. XP aconseja que la realización de las pruebas unitarias al sistema se realice de forma automatizada y en etapas tempranas del desarrollo, permitiendo disminuir la ocurrencia de defectos y aprovechar las ventajas de la retroalimentación que se produce en el proceso (Beck, 2002).

Entre las ventajas que ofrecen este tipo de pruebas se encuentran:

- Al escribir primero los casos de prueba, se define de manera formal los requisitos que se espera que cumpla la aplicación.
- Al escribir una prueba de unidad, se piensa en la forma correcta de utilizar un módulo que aún no existe.
- Los casos de prueba permiten que los desarrolladores no sientan inseguridad a la hora de realizar modificaciones en el código.

Pressman en su libro Ingeniería del Software un enfoque práctico quinta edición, plantea que la prueba del camino básico (propuesto por Tom McCabe) es una técnica de prueba unitaria que permite obtener una medida de la complejidad de un diseño procedimental, y utilizar esta medida como guía para la definición de una serie de caminos básicos de ejecución, diseñando casos de prueba que garanticen que cada camino se ejecuta al menos una vez(Pressman, 2002).

“La complejidad ciclomática es una métrica de software que brinda una medición cuantitativa de la complejidad lógica de un programa”(Pressman, 2002). Define el número de caminos independientes del conjunto básico de un programa. Determina el número de casos de prueba que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez.

La complejidad ciclomática se puede calcular de tres formas:

1. El número de regiones del grafo de flujo (V(G) = R)
2. Aristas - Nodos + 2 (V(G) = A – N + 2)
3. Nodos predicado + 1 (un nodo predicado es el que representa una condicional if o case, es decir, que de él salen varios caminos) (V(G) = P + 1)

El valor de V(G) da el número de caminos linealmente independientes de la estructura de control del programa. Entonces se preparan los casos de prueba que forzarán la ejecución de cada camino del conjunto básico.

Para la realización de las pruebas unitarias se propone utilizar JUnit, este es un marco de trabajo (framework) para automatizar las pruebas unitarias de aplicaciones Java(JUnit.org, 2012). Fue creado por Erich Gamma y Kent Beck en 1997. Es un software de código abierto. Consta de un conjunto de clases que el programador puede utilizar para construir sus casos de prueba y ejecutarlos automáticamente. Los casos de prueba son programas Java, quedan archivados y se pueden volver a ejecutar tantas veces como sea necesario. Es adecuado para el desarrollo dirigido por las pruebas. Permite evaluar el funcionamiento de los métodos de las clases, verificando si el valor obtenido es el esperado.

JUnit tiene muchas características que hacen que sea fácil de escribir y ejecutar pruebas:

- Separa instancias de la clase de prueba y cargadores de clases para cada prueba de unidad para evitar efectos secundarios.
- Anotaciones JUnit para proporcionar métodos de inicialización de los recursos y la recuperación: @Before, @BeforeClass, @After, y @AfterClass.
- Una variedad de métodos para afirmar que sea fácil de comprobar los resultados de sus pruebas.
- Integración con el IDE NetBeans.

1.10. CONCLUSIONES DEL CAPÍTULO

En este capítulo se expusieron los conceptos necesarios para la comprensión de la solución informática que da origen a esta investigación. También se realizó un estudio sobre la firma digital como mecanismo esencial para dar valor probatorio a los documentos electrónicos, demostrando la necesidad de contar con una herramienta de firma digital en el centro CEGEL. Se realizó un estudio de las tecnologías, herramientas y tendencias de desarrollo de software actuales, exponiendo sus

principales características y ventajas, lo que ayudó a seleccionar las herramientas para la construcción de la aplicación, quedando XP como metodología de desarrollo a utilizar, Visual Paradigm como herramienta CASE, Java como lenguaje de programación, NetBeans como entorno de desarrollo.

CAPÍTULO 2. PROPUESTA DE SOLUCIÓN

2.1. INTRODUCCIÓN

En el capítulo anterior se seleccionó XP como la metodología a utilizar para guiar el desarrollo de la solución propuesta, en este se realiza una descripción referente a la fase planificación que propone dicha metodología. Se escriben las historias de usuario, además de hacer referencia a otros artefactos generados en esta fase. Se describen los requisitos funcionales y no funcionales, así como las principales características del sistema.

2.2. REQUISITOS DE SOFTWARE

"Un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. En el otro extremo, es una definición detallada y formal de una función del sistema."(Sommerville, 2005), así define Ian Sommerville un requisito de software en su libro *Ingeniería del software*. La calidad con que se realiza la captura de los requisitos afecta todo el proceso de desarrollo del software repercutiendo en el resto de las fases de desarrollo del mismo. Además contribuye a tomar mejores decisiones de diseño y de arquitectura.

2.2.1. REQUISITOS FUNCIONALES

Sommerville plantea que 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).

Requisitos funcionales (RF) identificados:

RF1. Gestionar certificado digital.

RF1.1. Gestionar certificado para tarjeta inteligente.

RF1.2. Gestionar certificado para token usb.

RF1.3. Gestionar certificado para dispositivos de almacenamiento.

RF2. Firmar documentos con extensión PDF.

RF3. Validar documentos firmados.

RF4. Firmar varios documentos de forma simultánea.

RF5. Permitir multifirma del documento.

RF6. Visualizar documentos.

RF7. Personalizar firma para el documento.

RF7.1. Permitir firma visible.

RF7.2. Personalizar imagen de fondo de la firma.

RF7.3. Posicionar firma en el documento.

2.2.2. REQUISITOS NO FUNCIONALES

Sommerville define los requisitos no funcionales como “restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares.”(Sommerville, 2005).

Requisitos no funcionales (RNF) identificados:

Apariencia o interfaz

RNF1. Debe presentar una interfaz amigable que permita la fácil interacción con el sistema.

Usabilidad

RNF2. El software debe adaptarse al lenguaje y términos utilizados por los clientes, con vista a una mayor comprensión de la herramienta de trabajo.

Seguridad

RNF3. Confiabilidad: La información manejada por el sistema debe estar protegida de acceso no autorizado.

RNF4. Integridad: La información manejada por el sistema debe ser objeto de cuidadosa protección contra la corrupción y estados de inconsistencia.

Fiabilidad

RNF5. Garantizar capacidad para capturar excepciones.

Portabilidad

RFN6. El sistema será multiplataforma, podrá correr en Linux y en Windows.

Software

RNF7. El lenguaje de programación Java.

RNF8. La herramienta IDE de desarrollo utilizada será NetBeans 6.9.

RNF9. Tener instalada la máquina virtual de java.

2.3. Fase de Planificación

La metodología XP define como fase inicial la planificación. Durante esta etapa se lleva a cabo el proceso de identificación y confección de las historias de usuario, así como la familiarización del equipo de trabajo con las tecnologías y herramientas seleccionadas para el desarrollo del software. También el cliente especifica la prioridad en que se deben implementar las historias de usuario, así como una estimación del esfuerzo que costará implementar todas las historias de usuario. El resultado de la fase es un plan de entregas donde se realiza una estimación de las versiones que tendrá el producto en su realización, de manera tal que guíe el desarrollo del mismo (Beck, 2002).

2.3.1. Historias de Usuarios

XP se basa fundamentalmente en las historias de usuario para representar los requerimientos del sistema. Las historias de usuarios (HU) son escritas por el cliente, en su propio lenguaje, se realiza una por cada característica principal del sistema, se emplean para hacer estimaciones de tiempo y para el plan de lanzamientos, reemplazan un gran documento de requisitos y presiden la creación de las pruebas de aceptación. La idea es que sean sencillas y que satisfagan al cliente (Beck, 2002).

Historia de Usuario	
Número: 5	Nombre: Firmar documentos
Usuario: Desarrollador	
Prioridad en Negocio: Muy Alta	Riesgo en Desarrollo: Alto
Puntos Estimados: 2	Iteración Asignada:1
Programador responsable: Rafael	
Descripción: Firmará digitalmente un documento que podrá ser en formato PDF utilizando un certificado digital que estará almacenado en el contenedor seleccionado, como	

resultado de la operación se obtendrá el documento firmado, con las características de la firma definidas por el usuario previamente (firma visible, posición de la firma dentro del documento, imagen de fondo en la firma), el cual se guardará en el directorio seleccionado por el usuario en las preferencias generales.
Observaciones:

Tabla 2. Historia de usuario número 5: Firmar documentos.

Historia de Usuario	
Número: 6	Nombre: Validar documentos firmados
Usuario: Desarrollador	
Prioridad en Negocio: Muy Alta	Riesgo en Desarrollo: Medio
Puntos Estimados: 2	Iteración Asignada:1
Programador responsable: Rafael	
Descripción: Permitirá la validación del documento que fue firmado, comprobando quién ha firmado el documento y la validez de su certificado, a través de la comprobación con el repositorio de certificados brindado por la entidad certificadora.	
Observaciones:	

Tabla 3. Historia de usuario número 6: Validar documentos firmados.

2.3.2. ESTIMACIÓN DE ESFUERZOS POR HISTORIAS DE USUARIO

En el epígrafe se realiza la estimación del esfuerzo por historia de usuario para lo cual se hace necesario tener en cuenta que estas deben ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas, se divide en dos o más historias. Si es menor de una semana, se combina con otra historia. Estas estimaciones permiten tener una medida de la velocidad del proyecto y ofrecen una guía a la cual ajustarse. Los resultados estimados se muestran a continuación.

Historias de Usuario	**Puntos de Estimación (semanas)**
Gestionar certificados digitales	1
Gestionar certificados para tarjeta inteligente	2
Gestionar certificados para token usb	2
Gestionar certificados para dispositivos de almacenamiento	2

Firmar documentos	3
Validar documentos firmados	2
Firmar varios documentos de forma simultánea	1
Permitir multifirma del documento	1
Visualizar documentos	1
Personalizar firma para el documento	1
Permitir firma visible	1
Personalizar imagen de fondo de la firma	1
Posicionar firma en el documento	1

Tabla 4. Estimación de esfuerzos por historias de usuario.

2.3.3. PLAN DE ITERACIONES

Luego de identificar y definir las historias de usuario y estimar el esfuerzo propuesto para la realización de cada una de las HU, se pasa a la planificación de la etapa de implementación del sistema. Se seleccionan las historias de usuario que se implementan en cada iteración, de acuerdo al nivel de prioridad de las mismas, así como las posibles fechas de sus entregas.

2.3.3.1. PLAN DE DURACIÓN DE LAS ITERACIONES

Este plan tiene como objetivo mostrar la duración de cada iteración, así como el orden en que serán implementadas las historias de usuario en cada una de las mismas.

Iteraciones	**Orden de las historias usuario a implementar**	**Duración de las iteraciones**
Iteración 1	Gestionar certificados digitales	4
	Gestionar certificados para tarjeta inteligente	
	Gestionar certificados para token USB	
	Gestionar certificados para dispositivos de almacenamiento	
	Firmar documentos	
Iteración 2	Validar documentos firmados	2
	Firmar varios documentos de forma simultánea	
	Permitir multifirma del documento	
Iteración 3	Visualizar documentos	3
	Personalizar firma para el documento	

	Permitir firma visible	
	Personalizar imagen de fondo de la firma	
	Posicionar firma en el documento	
Total		9

Tabla 5. Plan de duración de las iteraciones.

2.3.4. PLAN DE ENTREGA

Plan de entregas (“Release Plan”): el cronograma de entregas establece qué historias de usuario serán agrupadas para conformar una entrega, y el orden de las mismas (Joskowicz, 2008). En este plan se concentran las funcionalidades referentes a un mismo tema en módulos, esto permite un mayor entendimiento en la fase de implementación. Tiene como objetivo definir el número de liberaciones que se realizarán en el transcurso del proyecto y las iteraciones que se requieren para desarrollar cada una. De esta forma se puede trazar el plan de entrega en función de estos dos parámetros: el tiempo de desarrollo ideal y el grado de importancia para el cliente.

Módulos	HU que abarca
Preferencias	Gestionar certificados digitales. Gestionar certificados para tarjeta inteligente. Gestionar certificados para token usb. Gestionar certificados para dispositivos de almacenamiento. Personalizar firma para el documento. Permitir firma visible. Personalizar imagen de fondo de la firma. Posicionar firma en el documento.
Firmar documento	Firmar documentos. Firmar varios documentos de forma simultánea. Permitir multifirma del documento. Permitir firma visible. Visualizar documentos.
Validación de documento	Validar documentos firmados. Visualizar documentos

Tabla 6. Funcionalidades por módulos.

	Final 1ra Iteración	**Final 2da Iteración**	**Final 3ra Iteración**

Módulos	2da semana de Marzo	3ra semana de abril	2da semana de mayo
Preferencias	v1.0	v1.1 Final	Finalizado
Firmar documento		v1.2	v1.4 Final
Validación de documento.		v1.3	V1.5 Final

Tabla 7. Plan de duración de entrega.

2.4. CONCLUSIONES DEL CAPÍTULO

En este capítulo se ha descrito la primera etapa del ciclo de vida de la solución propuesta, la fase de Planificación propuesta por la metodología XP, generando las historias de usuario que caracterizan al sistema, así como la estimación del esfuerzo necesario para la implementación de las mismas. Además se construyó el plan de iteraciones, se realizó la estimación del tiempo que requiere la implementación de cada iteración y se llevó a cabo el plan de entregas en el cual se define qué historias de usuario formarían parte de cada entrega, posibilitando el paso a la siguiente fase de desarrollo.

CAPÍTULO 3. DISEÑO Y DESARROLLO

3.1. INTRODUCCIÓN

En el presente capítulo se hace referencia a los elementos que conforman el diseño del sistema a construir, que acorde a lo planteado por la metodología XP debe ser un diseño simple, sencillo y de fácil interacción, basado principalmente en el desarrollo de las tarjetas Clases, Responsabilidad y Colaboración (CRC). En este capítulo también se realiza un acercamiento a los elementos de la implementación, se detallan las tres iteraciones llevadas a cabo durante la construcción de la herramienta, así como las tareas generadas por cada HU.

3.2. FASE DE DISEÑO

La metodología de desarrollo XP plantea prácticas especializadas que accionan directamente en la realización del diseño para lograr un sistema robusto y reutilizable, tratando en todo momento de conservar su simplicidad, es decir, crear un diseño evolutivo que va mejorando incrementalmente y que permite hacer entregas pequeñas y frecuentes de valor para el cliente, basado principalmente en el desarrollo de las tarjetas CRC.

3.2.1. TARJETAS CRC

Las tarjetas CRC (Clase, Responsabilidad y Colaboración) son utilizadas para representar las responsabilidades de las clases y sus interacciones. Estas tarjetas permiten trabajar con una metodología basada en objetos, permitiendo que el equipo de desarrollo completo contribuya en la tarea del diseño. El nombre de la clase se coloca a modo de título en la tarjeta, las responsabilidades se colocan a la izquierda y las clases que se implican en cada responsabilidad a la derecha, en la misma línea que su requerimiento correspondiente. Las tarjetas determinan el comportamiento de cada actividad.

Clase: ValidarDocumento	
Responsabilidad	Colaboración
Permite validar los documentos firmados.	Controladora

Tabla 8. Tarjeta CRC para la clase ValidarDocumento.

Clase: FirmarDocumento

Responsabilidad	Colaboración
Permite realizar la firma de los documentos.	Controladora

Tabla 9. Tarjeta CRC para la clase FirmarDocumento.

Clase: FicheroConfiguracion	
Responsabilidad	Colaboración
Permite escribir y cargar de un fichero las preferencias seleccionadas por el usuario.	Controladora

Tabla 10. Tarjeta CRC para la clase FicheroConfiguracion.

3.2.2. ESTÁNDARES DE CODIFICACIÓN.

XP resalta que la comunicación de los programadores es a través del código, por lo que es necesario que sigan ciertos estándares de programación para lograr un entendimiento entre los programadores y que cualquier persona del equipo de desarrollo pueda modificar cualquier parte del código, además que sea un código entendible por otros programadores que posteriormente puedan apoyarse en ese trabajo y poder desarrollar otras soluciones.

En el caso de la herramienta que se desarrolla, el estándar que se utiliza en la implementación es:

- Arriba de todas las funciones se pone un comentario especificando el objetivo de la misma.
- Los nombres de los paquetes serán escritos con minúscula, en el caso de ser un nombre compuesto por más de una palabra serán separadas por "_".
- Las funciones comenzarán con letra inicial mayúscula, en el caso de ser un nombre compuesto por más de una palabra serán separadas por "_", y el resto de las palabras serán escritas con minúscula.
- Las clases comenzarán con letra inicial mayúscula, 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.
- Las variables serán escritas con minúscula, en el caso de ser un nombre compuesto por más de una palabra serán separadas por "_".

- Código de la clase: va precedido por comentarios tipo javadoc con la siguiente información: prólogo explicativo de la clase (opcional), autor, versión, referencias a otros elementos de código si se considera que debe haberlas.
- Organización de ficheros: las clases se agrupan en paquetes. (interfaces, clases, recursos).
- Sentencias de paquete: la primera línea no comentada de un fichero fuente debe ser la sentencia de paquete, que indica el paquete al que pertenece(n) la(s) clase(s) incluida(s) en el fichero fuente. Por ejemplo package javax.crypto*;*
- Sentencias de importación: tras la declaración del paquete se incluirán las sentencias de importación de los paquetes necesarios.
- Comentarios de implementación:
  - ✓ Comentarios de bloque: permiten la descripción de ficheros, clases, bloques, estructuras de datos y algoritmos.

```
/*
 * Esto es un comentario
 * de bloque
 */
```

  - ✓ Comentarios de línea: son comentarios cortos localizados en una sola línea y tabulados al mismo nivel que el código que describen.

```
// Esto es un comentario de línea
```

- Una declaración por línea

```
String pdf_Afirmar;
Byte[] byte;
```

- Inicialización: toda variable local tendrá que ser inicializada en el momento de su declaración, salvo que su valor inicial dependa de algún valor que tenga que ser calculado previamente.

  Int cant = 0;
- La sentencia "try/catch" siempre debe tener el formato siguiente:

```
try {
    sentencias;
} catch (ClaseException e) {
    sentencias;
}
```

- Espacios en blanco
  - ✓ Entre una palabra clave y un paréntesis. Esto permite que se distingan las llamadas a métodos de las palabras clave. Por ejemplo:

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

- ✓ Tras cada coma en un listado de argumentos. Por ejemplo:

```
objeto.unMetodo(a, b, c);
```

- ✓ Al realizar el moldeo o "casting" de clases. Ejemplo:

```
Unidad unidad = (Unidad) objeto;
```

3.2.3. PATRONES GRASP

GRASP es el acrónimo de General Responsibility Assignment Software Patterns. Los GRASP son patrones para la asignación de responsabilidades, que ayudan a entender el diseño de objetos. Incluso cuando se utilizan metodologías ágiles como XP, es necesario elegir cuidadosamente las clases adecuadas y decidir como estas deben interactuar (Larman, 2003). Es por ello que resulta de vital importancia el uso de patrones GRASP. Algunos de los más importantes son (Larman, 2003):

- Experto: la responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase contiene toda la información necesaria para realizar la labor que tiene encomendada.
- Creador: este patrón como su nombre lo indica es el que crea, él guía la asignación de responsabilidades relacionadas con la creación de objetos.
- Bajo acoplamiento: el sistema debe contener pocas dependencias entre clases (principio básico para la creación un buen diseño).
- Alta cohesión: cada elemento del diseño debe realizar una labor única dentro del sistema. El empleo de este patrón proporciona refactorización en el sistema.
- Controlador: la responsabilidad de controlar el flujo de eventos del sistema se debe asignar a clases específicas. Esto facilita la centralización de actividades en el sistema.

3.2.4. PATRONES GOF

Dentro de los patrones GoF presentes que ayudan a tener un diseño más sencillo pero a la vez robusto se encuentran:

- Patrón estrategia: define una familia de algoritmos, los encapsula en clases distintas y los hace intercambiables. Esto se hace evidente en las clases realizar la firma digital como FirmarDocumento donde se trabaja de forma que

se encapsulan de forma independiente los algoritmos para realizar la firma digital por los diferentes estándares pkcs11 y pkcs12.

- Patrón observador: permite definir dependencias entre objetos de forma que si un objeto cambia de estado todos los dependientes de él son notificados y actualizados automáticamente, evidenciándose en la relación entre las clases Preferencias, FicheroConfiguración y las otras clases interfaces, así como la clase ValidarDocumento y FirmarDocumento, ya que los cambios que se realizan en un objeto de la clase Preferencia implica realizar cambios en los objetos de la clase FicheroConfiguración y otras clases, además de ser mínimo el acoplamiento entre estas clases.
- Patrón decorador: el paquete java.io de Java constituye un excelente ejemplo del patrón decorador. La jerarquía de clases cuya raíz es la clase abstracta OutputStream permite la combinación de filtros (FilterOutputStream) para combinar funciones diversas como son la salida sobre ficheros, utilizando un buffer y manipulando directamente datos básicos Java.
- Patrón compuesto: compone objetos en tres estructuras para representar jerarquías que relacionan el todo con las partes, poniéndose de manifiesto en la clase Component del paquete java.awt y la clase Container que están compuestas de un número indeterminado de componentes como JButton, JLabel, JTextField, JScrollPane, JTabbedPane o JSplitPane.

3.3. FASE DE DESARROLLO

En esta fase se realiza la implementación de las HU que fueron seleccionadas por cada iteración. Al inicio se lleva a cabo un chequeo del plan de iteraciones por si es necesario realizar modificaciones. Como parte de este plan se crean tareas de ingeniería para ayudar a organizar la implementación exitosa de las HU.

3.3.1. TAREAS DE INGENIERÍA POR ITERACIONES

Cada HU está compuesta por una o varias tareas de ingeniería, éstas no son más que pasos lógicos a seguir por el programador para realizar la implementación de una HU. A continuación se detallan para la iteración número uno, las tareas a desarrollar por cada HU.

Historias de Usuario	Tarea de ingeniería por historias de usuario
Gestionar certificados digitales	• Elaborar prototipo de interfaz.

	• Permitir seleccionar uno de los tres tipos de contenedores de certificados digitales. • Permitir escribir en el fichero de configuración las preferencias seleccionadas por el usuario.
Gestionar certificados para tarjeta inteligente	• Permitir cargar los driver del fabricante. • Permitir la interacción de la herramienta con la tarjeta inteligente.
Gestionar certificados para token usb	• Permitir cargar los driver del fabricante. • Permitir la interacción de la herramienta con el token usb.
Gestionar certificados para dispositivos de almacenamiento	• Permitir la interacción de la herramienta con el dispositivo de almacenamiento.
Firmar documentos	• Elaborar prototipo de interfaz. • Permitir cargar documentos. • Permitir la opción de tener la firma visible. • Permitir firma de documento. • Permitir obtener copia del documento original pero ya firmado. • Permitir eliminar documentos de la lista de documentos a firmar.

Tabla 11. Distribución de HU por cada iteración (iteración 1).

3.3.2. TAREAS DETALLADAS

Tarea de ingeniería

Número de tarea: 1	**Nombre Historias de usuario:** Firmar documentos.
Nombre de la tarea: Elaborar prototipo de interfaz	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 1
Fecha inicio: 10/02/2012	**Fecha fin:** 11/02/2012
Programador responsable: Rafael Rodríguez.	
Descripción: Esta tarea sirve de apoyo para la posterior implementación ya que muestra como se visualizará la pantalla de firmar documento.	

Tabla 12. Descripción de la tarea de ingeniería #1.

Tarea de ingeniería	
Número de tarea: 2	**Nombre Historias de usuario:** Firmar documentos.
Nombre de la tarea: Permitir cargar documentos	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 1
Fecha inicio: 11/02/2012	**Fecha fin:** 12/02/2012
Programador responsable: Rafael Rodríguez.	
Descripción: Esta tarea permitirá cargar los documentos para una lista, y posteriormente poder firmarlos y/o visualizarlos.	

Tabla 13. Descripción de la tarea de ingeniería #2.

Tarea de ingeniería	
Número de tarea: 3	**Nombre Historias de usuario:** Firmar documentos.
Nombre de la tarea: Permitir la opción de tener la firma visible	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 2
Fecha inicio: 11/02/2012	**Fecha fin:** 12/02/2012
Programador responsable: Rafael Rodríguez.	
Descripción:	

Esta tarea permitirá activar la opción de firma visible en el documento, en caso de no haberse configurado así en las preferencias.

Tabla 14. Descripción de la tarea de ingeniería #3.

3.4. CONCLUSIONES DEL CAPÍTULO

En el presente capítulo se trataron aspectos relacionados con las fases de diseño e implementación de la herramienta propuesta como solución, apoyándose en lo propuesto por la metodología XP. Se generaron los artefactos asociados a estas fases como las tarjetas CRC y las tareas de programación por cada una de las historias de usuario. También se vieron otros aspectos de interés como los patrones de diseño utilizados, así como los estándares de codificación empleados, obteniendo el producto final como resultado de la implementación, dejando todo listo para realizar las pruebas de funcionalidad, aspecto que será tratado en el próximo capítulo.

CAPÍTULO 4. PRUEBAS

En el presente capítulo se describe la validación de la herramienta, mostrando el resultado de las pruebas de aceptación y las pruebas unitarias.

4.1. PRUEBAS DE ACEPTACIÓN

Como se había mencionado anteriormente las pruebas de aceptación son una herramienta para validar que el sistema cumple con el funcionamiento requerido y esperado por el cliente. A continuación se muestran los casos de prueba de aceptación aplicados y los resultados arrojados luego de haber sido realizadas las pruebas.

4.1.1. CASOS DE PRUEBA

La aplicación de casos de pruebas apropiados es esencial para la validación y verificación del sistema construido. A continuación se muestran los casos de prueba de aceptación aplicados a las historias de usuario *Firmar documentos* y *Validar documentos firmados*.

Caso de prueba de aceptación	
Código: HU5_P1	**Historia de Usuario:** 5
Nombre: Firmar documentos.	
Descripción: Prueba para la funcionalidad de firmar los documentos seleccionados por el usuario.	
Condiciones de Ejecución: • El usuario debe seleccionar al menos un documento para ser firmado. • El usuario debe haber seleccionado un contenedor de certificados que almacena el certificado con el cual firmará el documento. • El usuario debe haber seleccionado un certificado para realizar la firma. • El usuario de forma opcional indica si la firma estará visible o no en el documento firmado. • El usuario debe haber especificado la posición que tendrá la firma en el documento firmado. • El usuario debe haber especificado si utilizará una imagen de fondo en la firma, o	

no en el documento firmado. • El usuario debe haber especificado la dirección donde se guardará el documento firmado. • El usuario debe seleccionar la opción firmar.
Resultados esperados: El sistema muestra un mensaje informando que se ha firmado correctamente el documento y se obtiene el documento firmado.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 15. Caso de prueba de aceptación Firmar documentos.

Caso de prueba de aceptación	
Código: HU6_P1	**Historia de Usuario:** 6
Nombre: Validar documentos firmados.	
Descripción: Prueba para la funcionalidad de validar los documentos firmados por el usuario.	
Condiciones de Ejecución: • El usuario debe seleccionar un documento para ser validado. • El usuario debe haber seleccionado el repositorio de certificados brindado por la entidad certificadora. • El usuario debe seleccionar la opción validar.	
Resultados esperados: En caso que la comprobación de la validez del certificado sea satisfactoria, el sistema muestra un mensaje informando que el certificado utilizado para la firma del documento es válido. En caso contrario el sistema muestra un mensaje de certificado desconocido o certificado revocado, en dependencia del motivo por el cual no se pudo comprobar la validez del documento.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 16. Caso de prueba de aceptación Validar documentos firmados.

4.1.2. ANÁLISIS DE LOS RESULTADOS.

Para validar que el resultado obtenido por el sistema coincide con el resultado esperado por el cliente se diseñaron un total de 13 casos de prueba de aceptación en conjunto cliente-desarrolladores. De este total, 10 arrojaron el resultado esperado mientras que 3 pruebas resultaron fallidas, las funcionalidades que respondían a estas pruebas fueron tratadas en la siguiente iteración y al volver a aplicar las pruebas de funcionalidad mostraron un resultado exitoso. Finalmente se obtuvieron un total 13 pruebas satisfactorias de 13 casos de prueba aplicados.

4.2. PRUEBAS UNITARIAS

Al principio de la investigación se señaló que las pruebas unitarias son la forma de probar el correcto funcionamiento en código de un sistema informático, lo que es de vital importancia a la hora de obtener un sistema de calidad. Seguidamente se describirá la realización de las pruebas unitarias realizadas, desglosándola en sus elementos fundamentales: confección del grafo de flujo, cálculo de la complejidad ciclomática, extracción de los caminos independientes, realización de los casos de pruebas y análisis de los resultados obtenidos.

A continuación se muestra el código del método **Firmar** de la clase **FirmarDocumento**, al cual se le aplicó la métrica de complejidad ciclomática debido a que es uno de los métodos relevantes en la solución informática brindada.

```
public void Firmar(String dir_certif, String clave, String pdf_afirmar,Boolean firma_visible , String coord_x,
                   String coord_y, String directorio_docfirmado, String nombre_doc)
{
    try
    {
        KeyStore ks= KeyStore.getInstance("pkcs12");
        ks.load(new FileInputStream(dir_certif),
        clave.toCharArray());
        String alias = (String)ks.aliases().nextElement();
        PrivateKey key=(PrivateKey)ks.getKey(alias, clave.toCharArray());
        Certificate[] chain = ks.getCertificateChain(alias);
        PdfReader reader = new PdfReader(pdf_afirmar);
        FileOutputStream fout=new FileOutputStream(directorio_docfirmado +"/"+ nombre_doc);
        //La siguiente linea permite la multifirma
        PdfStamper stp = PdfStamper.createSignature(reader, fout, '\0', null, true);

        PdfSignatureAppearance sap= stp.getSignatureAppearance();
        sap.setCrypto(key, chain, null, PdfSignatureAppearance.WINCER_SIGNED);
        sap.setReason("Otorgar validez legal al documento");
        sap.setLocation("Centro CEGEL");
        // El siguiente codigo permite firma visible.
        if(firma_visible==true)
        {
            int coorden_x = Integer.parseInt(coord_x);
            int coorden_y = Integer.parseInt(coord_y);
            sap.setVisibleSignature(new Rectangle(coorden_x, coorden_y, coorden_x+60, coorden_y+60), 1, null);
        }
        stp.close();
    }
    catch(Exception e) {
            e.printStackTrace();
    }
}
```

Figura 1. Código del método Firmar de la clase FirmarDocumento.

Grafo de flujo asociado.

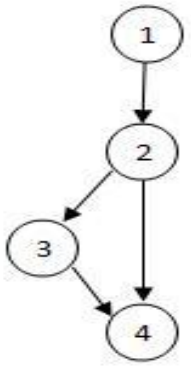


Figura 2. Grafo de flujo asociado al método Firmar de la clase FirmarDocumento.

Cálculo de la complejidad ciclomática.

V (G) = R = 2

V (G) = A – N + 2 = 4 – 4 + 2 = 2

V (G) = P + 1 = 1 + 1 = 2

Caminos independientes determinados.

Camino 1: 1-2-3-4

Camino 2: 1-2-4

4.2.1. CASOS DE PRUEBA

Para llevar a cabo las pruebas unitarias se utilizó el framework de pruebas JUnit. Al elaborar un caso de prueba en JUnit es necesario hacer una “clase de test”, que se llama igual a la clase original pero añadiendo el sufijo “Test” al final, además se puede elegir sobre qué método específico se desea realizar dicha prueba o si es a toda la clase. Para realizar una prueba el test utiliza el método assertEquals para comprobar si el resultado de llamar al método coincide con el esperado. Otras posibles funciones son assertTrue, assertFalse, etc. Si este comando falla cuando se ejecuta el caso de prueba, JUnit dará el caso por fallido, de lo contrario el resultado es correcto.

Se realizaron las pruebas por cada uno de los caminos independientes determinados, obteniendo resultados satisfactorios. A continuación se muestran las pruebas realizadas al método Firmar de la clase FirmarDocumento.

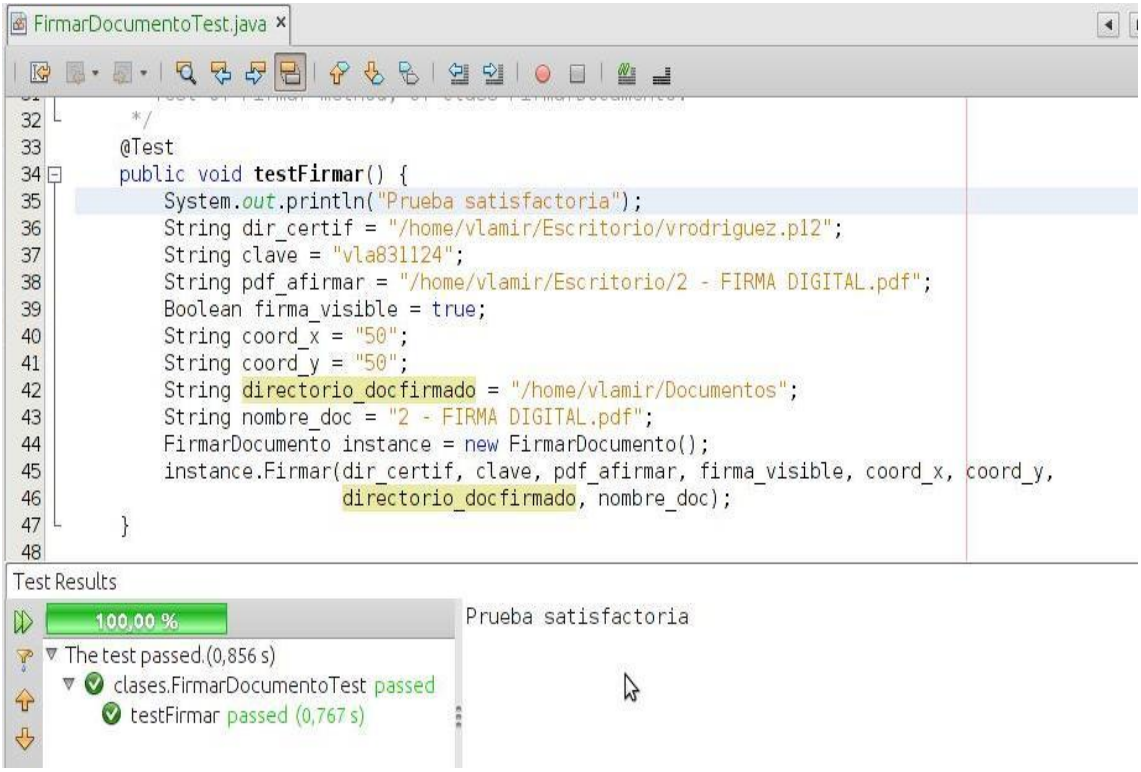


Figura 3. Caso de prueba del método Firmar para el camino 1.

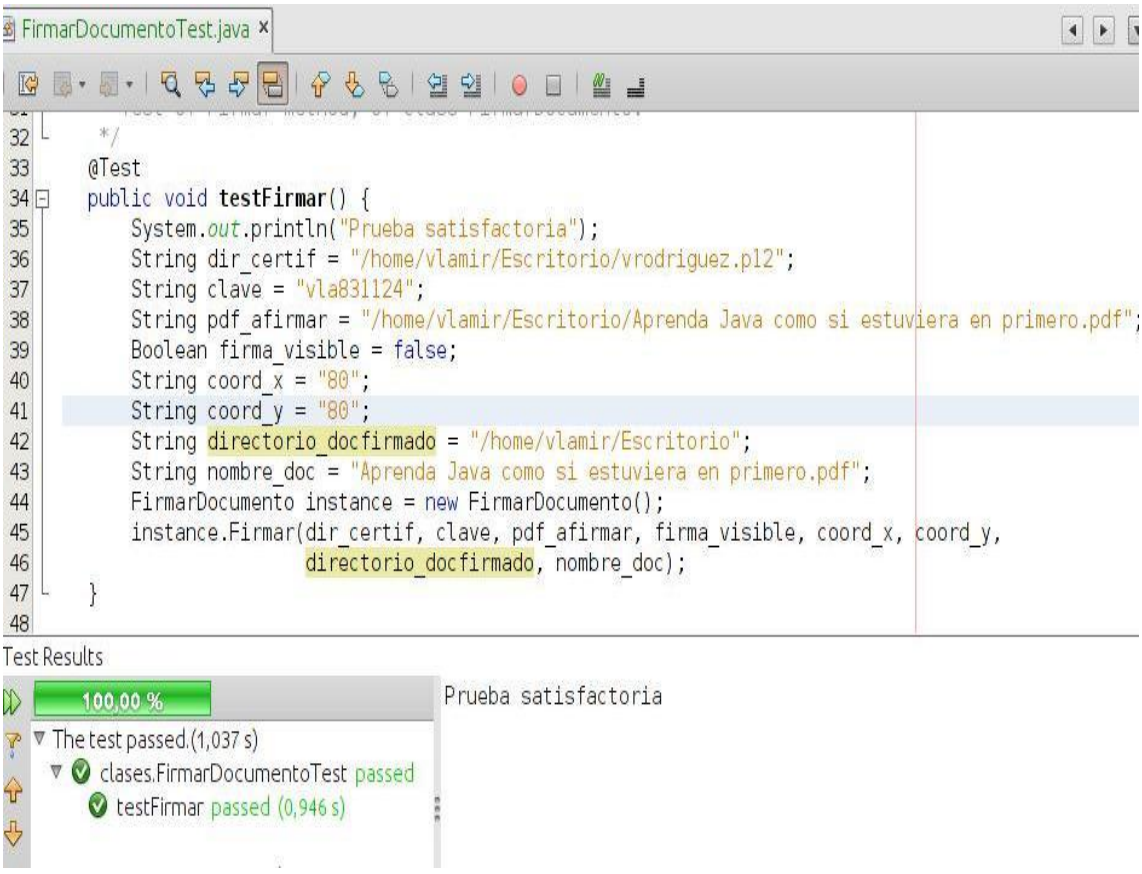


Figura 4. Caso de prueba del método Firmar para el camino 2.

4.2.2. Análisis de los resultados

Para la validación del código generado en el desarrollo de la herramienta se seleccionaron los métodos más relevantes, a los cuales se le realizaron las pruebas para poder evaluar si el funcionamiento de cada uno de estos métodos se comportó de la manera esperada. Se realizaron un total de 12 pruebas a las 5 funcionalidades seleccionadas como relevantes, de las cuales 10 resultaron satisfactorias en una primera iteración de pruebas y 2 no satisfactorias, estas últimas fueron solucionadas en una segunda iteración para obtener un 100 por ciento de pruebas satisfactorias, comprobándose la estabilidad de la lógica aplicada en el código generado en el desarrollo de la herramienta informática.

4.3. Conclusiones del capítulo

En el presente capítulo se abordaron los temas referentes a la fase de pruebas. Se realizaron las pruebas de aceptación y las pruebas unitarias, presentándose en ambas, los casos de prueba correspondientes y se realizó un análisis de los resultados obtenidos. Se logró una cobertura de pruebas elevada, garantizando así la calidad del sistema construido.

CONCLUSIONES GENERALES

Al término de esta investigación se concluye:

1. Se realizó un detallado proceso investigativo sobre las herramientas, lenguajes y metodologías existentes para el desarrollo de software, seleccionando luego las más apropiadas para llevar a cabo el ciclo completo de desarrollo de la herramienta en cuestión.
2. Con la generación de los artefactos correspondientes a cada una de las fases de desarrollo del software que propone la metodología XP, así como el uso de tecnologías Java, se desarrolló una herramienta multiplataforma para la firma digital de documentos en formato PDF que asegura la autenticidad, integridad y no repudio de la información plasmada en este tipo de documentos, proporcionando valor probatorio a los mismos.
3. Se realizó la validación de la herramienta a través de la aplicación de las pruebas unitarias y de aceptación, obteniendo resultados satisfactorios, demostrándose la calidad del sistema construido.

RECOMENDACIONES

- Incorporar a la herramienta la posibilidad de firmar nuevos tipos de formatos (jpg, xml, mp3, otros).
- Recomendar al centro CEGEL la utilización de la herramienta para la firma digital, en caso que los clientes de algún proyecto que se desarrolle necesite dar valor probatorio a los documentos digitales generados.

BIBLIOGRAFÍA

Acuña, Karenny Brito. 2009. *Selección de Metodologías de Desarrollo para Aplicaciones Web en la Facultad de Informática de la Universidad de Cienfuegos.* Cienfuegos : s.n., 2009.

Adams, Carlisle y Lloyd, Steve. 2002. *Understanding PKI: Concepts, Standards, and Deployment Considerations.* Estados Unidos : Addison-Wesley Longman Publishing Co., 2002. 0672323915.

Adobe. Adobe. *Adobe.* [En línea] [Citado el: 27 de 1 de 2012.] http://help.adobe.com/es_ES/Acrobat/9.0/Standard/WS58a04a822e3e50102bd615109794195ff-7d43.w.html.

Alba, Julio. 2006. *Firma y certificado electrónico.* España : s.n., 2006.

Beck, Kent. 1999. *Extreme Programming Explained.* Primera Edición. 1999. pág. 224. 0201616416.

—. 2002. *Una explicación de la Programación extrema: aceptar el cambio.* 2002. 8478290559.

Beck, Kent y Fowler, Martin. 2000. *Planning Extreme Programming.* 2000. 0201710919.

Comisión de las Naciones Unidas para el Derecho Mercantil Internacional. 2009. *Fomento de la confianza en el comercio electrónico: cuestiones jurídicas de la utilización internacional de métodos de autenticación y firma electrónicas.* Viena : s.n., 2009. 978-92-1-133663-4.

Cuervo, Mauro Callejas y Moreno, Oscar Yovany Baquero. 2005. *Herramientas libres para modelar software.* Colombia : s.n., 2005. 0121–1129.

Domingo, Ignacio Alamillo. 2002. *Tipología legal de la firma electrónica en la Unión Europea.* 2002. 1576-2033.

eParapher. *eParapher.* [En línea] [Citado el: 4 de 12 de 2011.] http://www.eparapher.com/index.html.

Fidanza., Dr. Jorge Alberto Lopez. 2004. *Firma Digital.* El Salvador : s.n., 2004.

Figueroa, Roberth G., Solís, Camilo J. y Cabrera, Armando A. 2009. *Metodologías Tradicionales vs. Metodologías Ágiles.* Universidad Técnica Particular de Loja,Escuela de Ciencias de la Computación. : s.n., 2009.

García de Jalón, Javier, y otros. 2000. *Aprenda Java como si estuviera en primero.* San Sebastián : s.n., 2000. pág. 155.

Grupo Soluciones Innnova S.A. 2007. Grupo Soluciones Innova. *Grupo Soluciones Innova.* [En línea] 2007. [Citado el: 15 de 1 de 2012.] http://www.rational.com.ar/herramientas/rup.html.

H. Canós, José , Letelier, Patricio y Penadés, María Carmen. *Métodologías Ágiles en el Desarrollo de Software.* Valencia : s.n. 46022.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000. 84-7829-036-2.

Jeffries, Ron, y otros. 2000. *Extreme Programming Installed.* 2000. 0201708426.

Joskowicz, José. 2008. *Reglas y Prácticas en eXtreme Programming.* España : s.n., 2008.

JUnit.org. 2012. JUnit.org. *JUnit.org.* [En línea] 2012. [Citado el: 20 de abril de 2012.] http://www.junit.org.

Lenguajes de Programación. 2009. Lenguajes de Programación. [En línea] 2009. [Citado el: 05 de 02 de 2012.] http://www.lenguajes-de-programacion.com/programacion-orientada-a-objetos.shtml.

López, Angel. 1997. *JAVA la programación del futuro.* Primera edición. Montevideo : s.n., 1997. 987-9131-38-X.

Maddison, R.N. 1983. *Information system methodologies.* Reino Unido : s.n., 1983. 0471903329.

Merlino, H. *Patrón de Diseño de Vistas Adaptables.* Buenos Aires : s.n. 1667-5002.

NetBeans Project. NetBeans. *NetBeans.* [En línea] [Citado el: 17 de 1 de 2012.] http://netbeans.org/community/releases/69/index_es.html.

Newkirk, James y Martin, Robert C. 2002. *La Programación Extrema en la práctica.* 2002. 8478290575.

Oracle. Java. *Java.* [En línea] [Citado el: 7 de 2 de 2012.] http://www.java.com/es/about/.

Pascale, Maricarmen. 2000. *Firma electrónica.* [CD de Memorias] Ciudad de México : s.n., 2000.

Peñalver Romero, Gladys Marsi. 2008. *Metodología ágil para proyectos de software libre.* La Habana : s.n., 2008.

Piattini, Mario. 2007. *Análisis y diseño detallado de aplicaciones informáticas de gestión.* Madrid : RA-MA, 2007. 9788478977765.

Pozo, Luis Salvador del. *Herramientas de validación de certificados en PKI con tarjeta inteligente.* Madrid : s.n.

Pressman, Roger. 2002. *Ingeniería de Software, un enfoque práctico. Quinta edición.* s.l. : McGraw-Hill, 2002. 8448132149.

2012. proyectosagiles.org. *proyectosagiles.org.* [En línea] 2012. [Citado el: 11 de 12 de 2011.] http://www.proyectosagiles.org/que-es-scrum.

Real Academia Española. 2001. Diccionario de la Lengua Española. *Diccionario de la Lengua Española.* [En línea] 2001. [Citado el: 5 de 3 de 2012.] http://buscon.rae.es/draeI/. 9788423968145.

Rising, L. y Janoff, N.S. 2000. The Scrum Software Development Process for Small Teams Retrieved. [En línea] 2000.

Rodríguez, Dra. Gladys. *De la firma autógrafa a la firma digital: perspectiva venezolana.* Zulia, Venezuela : s.n.

Schwaber, K. 2010. *Advanced Development Methods. SCRUM Development Process Retrieved.* 2010.

Schwaber, Ken y Sutherland, Jeff. 2011. *The Scrum Guide.* Estados Unidos : s.n., 2011.

Sinadura. *Sinadura.* [En línea] [Citado el: 25 de 1 de 2012.] http://www.sinadura.net/products-and-services/sinadura-desktop.

Sommerville, Ian. 2005. *Ingeniería del software.* Madrid : Pearson Educación S.A., 2005. 84-7829-074-5.

Visual Paradigm International. Visual Paradigm. *Visual Paradigm.* [En línea] [Citado el: 20 de 1 de 2012.] http:www.visual-paradigm.com.

Zúñiga, Lizbeth Angélica Barreto. 2011. *Evolución de la firma autógrafa a la firma electrónica avanzada.* [Digital] México : Coordinación de Publicaciones Digitales. Dirección General de Cómputo y de Tecnologías de Información y Comunicación -UNAM, 2011. 1067-6079.

ANEXOS

ANEXO 1. HISTORIAS DE USUARIO

Historia de Usuario	
Número: 1	Nombre: Gestionar certificados digitales
Usuario: Desarrollador	
Prioridad en Negocio: Muy alta	Riesgo en Desarrollo: Baja
Puntos Estimados: 1	Iteración Asignada:1
Programador responsable: Rafael	
Descripción: Permitirá seleccionar el contenedor del certificado digital para la firma y escribir en el fichero de configuración las preferencias seleccionadas por el usuario.	
Observaciones:	

Tabla 17. Historia de usuario número 1: Gestionar certificados digitales.

Historia de Usuario	
Número: 2	Nombre: Gestionar certificados para tarjeta inteligente
Usuario: Desarrollador	
Prioridad en Negocio: Muy alta	Riesgo en Desarrollo: Baja
Puntos Estimados:3	Iteración Asignada:1
Programador responsable: Rafael	
Descripción: Permite cargar los driver del fabricante que permitirá que la herramienta interactúe con la tarjeta inteligente para poder utilizar el certificado digital almacenado en la misma.	
Observaciones:	

Tabla 18. Historia de usuario número 2: Gestionar certificados para tarjeta inteligente.

Historia de Usuario	
Número: 3	Nombre: Gestionar certificados para token usb
Usuario: Desarrollador	
Prioridad en Negocio: Muy alta	Riesgo en Desarrollo: Baja
Puntos Estimados: 3	Iteración Asignada:1
Programador responsable: Rafael	
Descripción: Permite cargar los driver del fabricante que permitirá que la herramienta interactúe con el token usb para poder utilizar el certificado digital almacenado en el mismo.	
Observaciones:	

Tabla 19. Historia de usuario número 3: Gestionar certificados para token usb.

Historia de Usuario	
Número: 4	Nombre: Gestionar certificados para dispositivos de almacenamiento
Usuario: Desarrollador	
Prioridad en Negocio: Muy alta	Riesgo en Desarrollo: Baja
Puntos Estimados: 3	Iteración Asignada:1
Programador responsable: Rafael	
Descripción: Permite cargar los driver del fabricante que permitirá que la herramienta interactúe con el dispositivo de almacenamiento para poder utilizar el certificado digital	

almacenado en el mismo.
Observaciones:

Tabla 20. Historia de usuario número 4: Gestionar certificados para dispositivos de almacenamiento.

Historia de Usuario	
Número: 7	Nombre: Firmar varios documentos de forma simultánea
Usuario: Desarrollador	
Prioridad en Negocio: Alta	Riesgo en Desarrollo: Medio
Puntos Estimados: 2	Iteración Asignada:2
Programador responsable: Rafael	
Descripción: Permitirá firmar varios documentos al mismo tiempo, solo se tendrán que seleccionar los documentos a firmar y pulsar el botón 'firmar'. Sin tener que firmar cada documento de forma individual.	
Observaciones:	

Tabla 21. Historia de usuario número 7: Firmar varios documentos de forma simultánea.

Historia de Usuario	
Número: 8	Nombre: Permitir multifirma del documento
Usuario: Desarrollador	
Prioridad en Negocio: Media	Riesgo en Desarrollo: Bajo
Puntos Estimados: 2	Iteración Asignada:2
Programador responsable: Rafael	
Descripción: Permitirá la opción de firma múltiple, de forma que un mismo documento pueda ser firmado por más de una persona.	
Observaciones:	

Tabla 22. Historia de usuario número 8: Permitir multifirma del documento.

Historia de Usuario	
Número: 9	Nombre: Visualizar documentos
Usuario: Desarrollador	
Prioridad en Negocio: Media	Riesgo en Desarrollo: Bajo
Puntos Estimados: 1	Iteración Asignada:3
Programador responsable: Rafael	
Descripción: Permitirá visualizar los documentos seleccionados para poder observar su contenido.	
Observaciones:	

Tabla 23. Historia de usuario número 9: Visualizar documentos.

Historia de Usuario	
Número: 10	Nombre: Personalizar firma para el documento
Usuario: Desarrollador	
Prioridad en Negocio: Media	Riesgo en Desarrollo: Bajo
Puntos Estimados: 1	Iteración Asignada:3
Programador responsable: Rafael	
Descripción: Permitirá salvar en el fichero de configuración las preferencias del usuario en cuanto a las características de la firma (firma visible, posición de la firma dentro del documento, imagen de fondo en la firma).	
Observaciones:	

Tabla 24. Historia de usuario número 10: Personalizar firma para el documento.

ANEXO 2. PROTOTIPOS DE INTERFACES NO FUNCIONALES

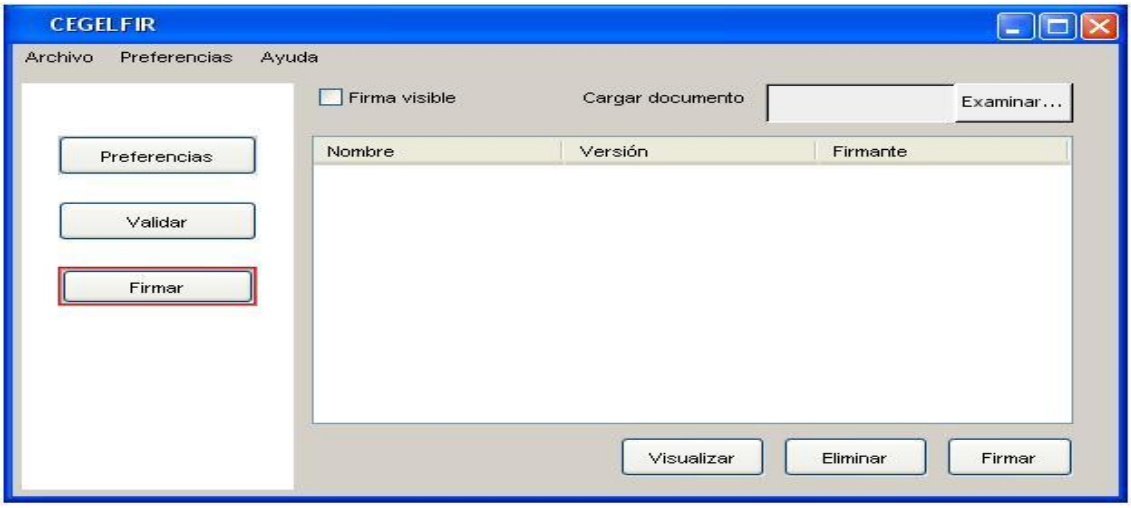


Figura 5. Prototipo de interfaz no funcional: interfaz firmar documento.

Figura 6. Prototipo de interfaz no funcional: interfaz gestionar certificado digital.

Figura 7. Prototipo de interfaz no funcional: interfaz preferencias generales.

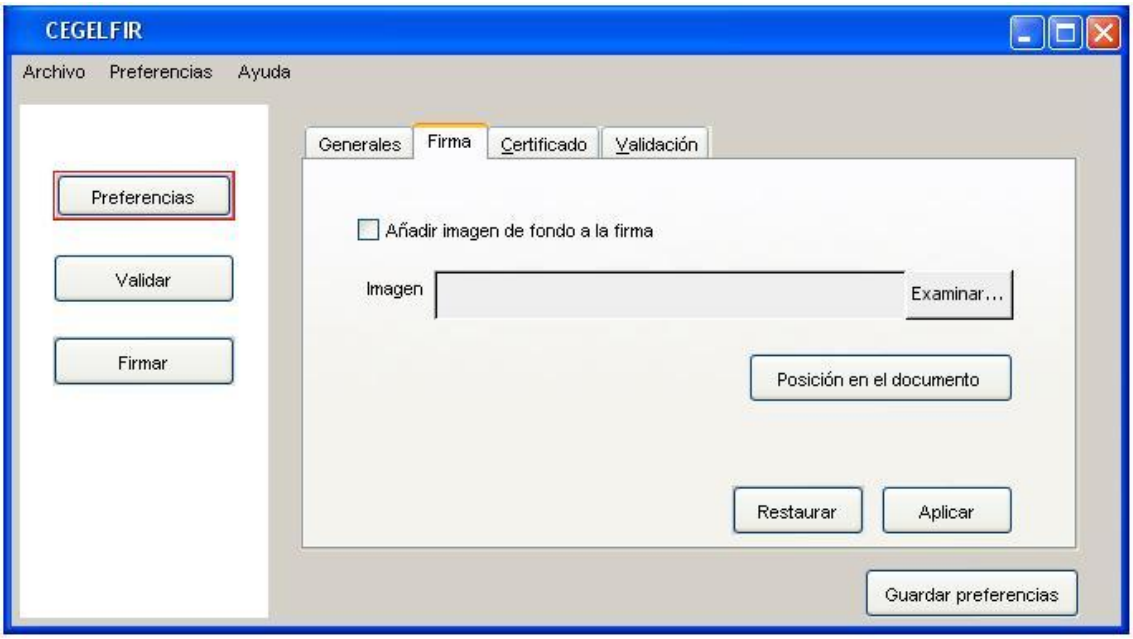


Figura 8. Prototipo de interfaz no funcional: interfaz preferencias de firma.

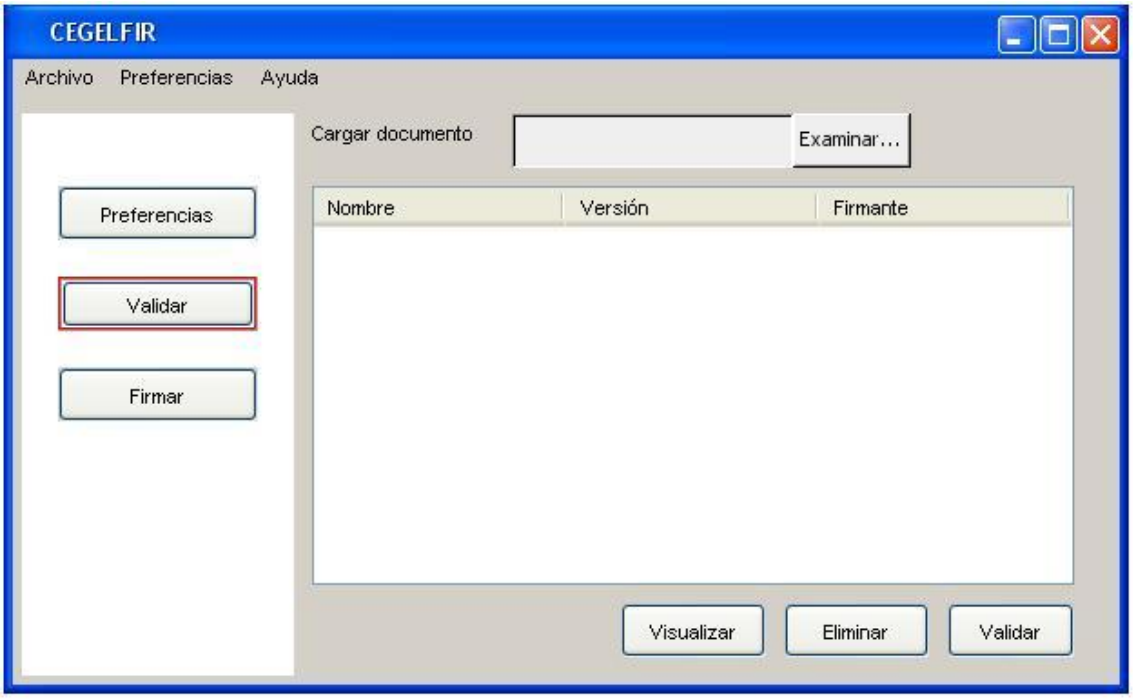


Figura 9. Prototipo de interfaz no funcional: interfaz validar documento.

ANEXO 3. TARJETAS CRC

Clase: Controladora	
Responsabilidad	Colaboración
Interviene en el flujo de firmar documento.	FirmarDocumento, FicheroConfiguracion, Principal.
Interviene en el flujo de validar documento.	ValidarDocumento, FicheroConfiguracion, Validar.
Está encargada de gestionar la escritura y lectura del fichero de configuración.	FicheroConfiguracion, Preferencias.

Tabla 25. Tarjeta CRC para la clase Controladora.

Clase: Login	
Responsabilidad	Colaboración
Permite capturar la contraseña del usuario para poder acceder a los certificados contenidos en los dispositivos.	Controladora, Principal.

Tabla 26. Tarjeta CRC para la clase Login.

Clase: Preferencias	
Responsabilidad	Colaboración.

• Permite guardar y mostrar las preferencias de la firma configuradas por el usuario. • Permite seleccionar el tipo de dispositivo contenedor de certificados que se va a utilizar. • Permite cargar los drivers para poder interactuar con los dispositivos contenedores de los certificados digitales para la realizar la firma. • Permite especificar la dirección donde serán guardados los documentos firmados. • Permite especificar el almacén de certificados con el cual se van a validar los documentos firmados.	Controladora.

Tabla 27. Tarjeta CRC para la clase Preferencias.

Clase: Principal	
Responsabilidad	Colaboración
Permite cargar los documentos que se van a firmar.	ExtensionFichero.
Permite visualizar los documentos.	
Permite seleccionar la funcionalidad validar a través del botón del mismo nombre.	Validar.
Permite seleccionar la funcionalidad preferencias a través del botón del mismo nombre.	Preferencias.
Permite realizar la firma de los documentos.	Controladora.

Tabla 28. Tarjeta CRC para la clase Principal.

Clase: Validar	
Responsabilidad	Colaboración
Permite cargar los documentos que se	ExtensionFichero.

van a validar.	
Permite visualizar los documentos.	
Permite realizar la validación de los documentos.	Controladora.

Tabla 29. Tarjeta CRC para la clase Validar.

ANEXO 4. TAREAS DE INGENIERÍA POR ITERACIONES

Historias de Usuario	Tarea de ingeniería por historias de usuario
Validar documentos firmados	• Elaborar prototipo de interfaz. • Permitir cargar documentos. • Permitir validación de documento. • Permitir eliminar documentos de la lista de documentos a validar.
Firmar varios documentos de forma simultánea	• Permitir realizar la firma de todos los documentos seleccionados para firmar de forma simultánea.
Permitir multifirma del documento	• Permitir que un documento pueda ser firmado por más de un usuario.

Tabla 30. Tareas de ingeniería por iteraciones: segunda iteración.

Historias de Usuario	Tarea de ingeniería por historias de usuario
Visualizar documentos	• Permitir visualizar documento.
Personalizar firma para el documento	• Permitir salvar la configuración de las características que tendrá la firma.
Permitir firma visible	• Permitir que la firma pueda estar visible en los documentos firmados.
Personalizar imagen de fondo de la firma	• Permitir cargar imagen. • Permitir añadir la imagen cargada

	como fondo de la firma que aparecerá en los documentos firmados.
Posicionar firma en el documento	• Permitir entrada de coordenadas. • Posicionar firma dentro del documento en las coordenadas introducidas por el usuario.

Tabla 31. Tareas de ingeniería por iteraciones: tercera iteración.

ANEXO 5. TAREAS DE INGENIERÍA DETALLADAS

Tarea de ingeniería	
Número de tarea: 4	**Nombre Historias de usuario:** Firmar documentos.
Nombre de la tarea: Permitir firma de documento	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 4
Fecha inicio: 15/02/2012	**Fecha fin:** 18/02/2012
Programador responsable: Rafael Rodríguez.	
Descripción: Esta tarea permitirá realizar la firma de los documentos PDF seleccionados para firmar.	

Tabla 32. Descripción de la tarea de ingeniería #4.

Tarea de ingeniería	
Número de tarea: 5	**Nombre Historias de usuario:** Firmar documentos.
Nombre de la tarea: Permitir eliminar documentos de la lista de documentos a firmar.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 1
Fecha inicio: 20/02/2012	**Fecha fin:** 20/02/2012
Programador responsable: Rafael Rodríguez.	
Descripción:	

Esta tarea permitirá eliminar documentos de la lista de documentos a firmar.

Tabla 33. Descripción de la tarea de ingeniería #5.

Tarea de ingeniería	
Número de tarea: 6	**Nombre Historias de usuario:** Gestionar certificados para dispositivos de almacenamiento.
Nombre de la tarea: Permitir la interacción de la herramienta con el dispositivo de almacenamiento.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 3
Fecha inicio: 21/02/2012	**Fecha fin:** 23/02/2012
Programador responsable: Rafael Rodríguez.	
Descripción: Esta tarea permitirá acceder a los certificados contenidos en el dispositivo de almacenamiento.	

Tabla 34. Descripción de la tarea de ingeniería #6.

Tarea de ingeniería	
Número de tarea: 7	**Nombre Historias de usuario:** Gestionar certificados para el token usb.
Nombre de la tarea: Permitir cargar los driver del fabricante.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 1
Fecha inicio: 24/02/2012	**Fecha fin:** 25/02/2012
Programador responsable: Rafael Rodríguez.	
Descripción: Esta tarea permitirá cargar los driver proporcionados por el fabricante del token para poder interactuar con dicho dispositivo.	

Tabla 35. Descripción de la tarea de ingeniería #7.

Tarea de ingeniería	
Número de tarea: 8	**Nombre Historias de usuario:** Gestionar certificados para el token usb.
Nombre de la tarea: Permitir la interacción de la herramienta con el token usb.	

Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 3
Fecha inicio: 24/02/2012	**Fecha fin:** 26/02/2012
Programador responsable: Rafael Rodríguez.	
Descripción: Esta tarea permitirá interactuar con el token y acceder a los certificados contenidos en el mismo.	

Tabla 36. Descripción de la tarea de ingeniería #8.

Tarea de ingeniería	
Número de tarea: 9	**Nombre Historias de usuario:** Gestionar certificados para tarjeta inteligente.
Nombre de la tarea: Permitir cargar los driver del fabricante.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 1
Fecha inicio: 27/02/2012	**Fecha fin:** 27/02/2012
Programador responsable: Rafael Rodríguez.	
Descripción: Esta tarea permitirá cargar los driver proporcionados por el fabricante de la tarjeta para poder interactuar con dicho dispositivo.	

Tabla 37. Descripción de la tarea de ingeniería #9.

Tarea de ingeniería	
Número de tarea: 10	**Nombre Historias de usuario:** Gestionar certificados para la tarjeta inteligente.
Nombre de la tarea: Permitir la interacción de la herramienta con la tarjeta inteligente.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 3
Fecha inicio: 1/03/2012	**Fecha fin:** 3/03/2012
Programador responsable: Rafael Rodríguez.	
Descripción: Esta tarea permitirá interactuar con la tarjeta y acceder a los certificados contenidos en la misma.	

Tabla 38. Descripción de la tarea de ingeniería #10.

Tarea de ingeniería	
Número de tarea: 11	**Nombre Historias de usuario:** Gestionar certificados digitales.
Nombre de la tarea: Elaborar prototipo de interfaz.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 1
Fecha inicio: 4/03/2012	**Fecha fin:** 4/03/2012
Programador responsable: Rafael Rodríguez.	
Descripción: Esta tarea sirve de apoyo para la posterior implementación ya que muestra como se visualizará la pantalla gestionar certificados digitales.	

Tabla 39. Descripción de la tarea de ingeniería #11.

Tarea de ingeniería	
Número de tarea: 12	**Nombre Historias de usuario:** Gestionar certificados digitales.
Nombre de la tarea: Permitir seleccionar uno de los tres tipos de contenedores de certificados digitales.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 1
Fecha inicio: 5/03/2012	**Fecha fin:** 5/03/2012
Programador responsable: Rafael Rodríguez.	
Descripción: Esta tarea permitirá la posibilidad de seleccionar el tipo de dispositivo contenedor de certificados para la realización de la firma.	

Tabla 40. Descripción de la tarea de ingeniería #12.

Tarea de ingeniería	
Número de tarea: 13	**Nombre Historias de usuario:** Gestionar certificados digitales.
Nombre de la tarea: Permitir escribir en el fichero de configuración las preferencias seleccionadas por el usuario.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 1

Fecha inicio: 6/03/2012	**Fecha fin:** 6/03/2012
Programador responsable: Rafael Rodríguez.	
Descripción: Esta tarea permitirá guardar en el fichero de configuración el tipo de dispositivo seleccionado por el usuario.	

Tabla 41. Descripción de la tarea de ingeniería #13.

Tarea de ingeniería	
Número de tarea: 14	**Nombre Historias de usuario:** Validar documentos firmados.
Nombre de la tarea: Elaborar prototipo de interfaz.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 1
Fecha inicio: 7/03/2012	**Fecha fin:** 7/03/2012
Programador responsable: Rafael Rodríguez.	
Descripción: Esta tarea sirve de apoyo para la posterior implementación ya que muestra como se visualizará la pantalla de validar documentos.	

Tabla 42. Descripción de la tarea de ingeniería #14.

Tarea de ingeniería	
Número de tarea: 15	**Nombre Historias de usuario:** Validar documentos firmados.
Nombre de la tarea: Permitir cargar documentos.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 1
Fecha inicio: 8/03/2012	**Fecha fin:** 8/03/2012
Programador responsable: Rafael Rodríguez.	
Descripción: Esta tarea permitirá cargar los documentos para una lista, para posteriormente poder validarlos y/o visualizarlos.	

Tabla 43. Descripción de la tarea de ingeniería #15.

Tarea de ingeniería

Número de tarea: 16	**Nombre Historias de usuario:** Validar documentos firmados.
Nombre de la tarea: Permitir validación de documento.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 4
Fecha inicio: 9/03/2012	**Fecha fin:** 13/03/2012
Programador responsable: Rafael Rodríguez.	
Descripción: Esta tarea permitirá realizar la validación de los documentos PDF firmados que sean seleccionados.	

Tabla 44. Descripción de la tarea de ingeniería #16.

Tarea de ingeniería	
Número de tarea: 17	**Nombre Historias de usuario:** Validar documentos firmados.
Nombre de la tarea: Permitir eliminar documentos de la lista de documentos a validar.	
Tipo de tarea: Desarrollo **(Desarrollo/Corrección/Mejora)**	**Puntos Estimados(días):** 1
Fecha inicio: 15/03/2012	**Fecha fin:** 15/03/2012
Programador responsable: Rafael Rodríguez.	
Descripción: Esta tarea permitirá eliminar documentos de la lista de documentos a validar.	

Tabla 45. Descripción de la tarea de ingeniería #17.

ANEXO 6. CASOS DE PRUEBAS DE ACEPTACIÓN

Caso de prueba de aceptación	
Código: HU1_P1	**Historia de Usuario:** 1
Nombre: Gestionar certificados digitales.	
Descripción: Prueba para la funcionalidad de gestionar certificados digitales.	
Condiciones de Ejecución: • El usuario debe entrar a la sección de preferencias. • El usuario debe entrar a la pestaña certificado de la sección preferencias. • El usuario debe seleccionar el tipo de contenedor de certificados.	

Resultados esperados: El sistema guarda en el fichero configuración el tipo de contenedor de certificados seleccionado por el usuario.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 46. Caso de prueba de aceptación Gestionar certificados digitales.

Caso de prueba de aceptación	
Código: HU2_P1	**Historia de Usuario:** 2
Nombre: Gestionar certificados para tarjeta inteligente.	
Descripción: Prueba para la funcionalidad de gestionar certificados para tarjeta inteligente.	
Condiciones de Ejecución: ● El usuario debe entrar a la sección de preferencias. ● El usuario debe entrar a la pestaña certificado de la sección preferencias. ● El usuario debe seleccionar como tipo contenedor de certificados la tarjeta inteligente. ● El usuario debe seleccionar los driver del fabricante para poder acceder al certificado almacenado en la tarjeta.	
Resultados esperados: El sistema guarda en el fichero configuración los driver para interactuar con la tarjeta inteligente.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 47. Caso de prueba de aceptación Gestionar certificados para tarjeta inteligente.

Caso de prueba de aceptación	
Código: HU3_P1	**Historia de Usuario:** 3
Nombre: Gestionar certificados para token usb.	
Descripción: Prueba para la funcionalidad de gestionar certificados para token usb.	
Condiciones de Ejecución: ● El usuario debe entrar a la sección de preferencias. ● El usuario debe entrar a la pestaña certificado de la sección preferencias. ● El usuario debe seleccionar como tipo contenedor de certificados el token usb. ● El usuario debe seleccionar los driver del fabricante para poder acceder al certificado almacenado en el token.	
Resultados esperados: El sistema guarda en el fichero configuración los driver para interactuar con el token usb.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 48. Caso de prueba de aceptación Gestionar certificados para token usb.

Caso de prueba de aceptación

Código: HU4_P1	**Historia de Usuario:** 4
Nombre: Gestionar certificados para dispositivos de almacenamiento.	
Descripción: Prueba para la funcionalidad de gestionar certificados para dispositivos de almacenamiento.	
Condiciones de Ejecución: • El usuario debe entrar a la sección de preferencias. • El usuario debe entrar a la pestaña certificado de la sección preferencias. • El usuario debe seleccionar como tipo contenedor de certificados el dispositivo de almacenamiento. • El usuario debe seleccionar el certificado almacenado en el dispositivo de almacenamiento.	
Resultados esperados: El sistema guarda en el fichero configuración el certificado seleccionado.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 49. Caso de prueba de aceptación Gestionar certificados para dispositivos de almacenamiento.

ANEXO 7. CASOS DE PRUEBAS UNITARIAS

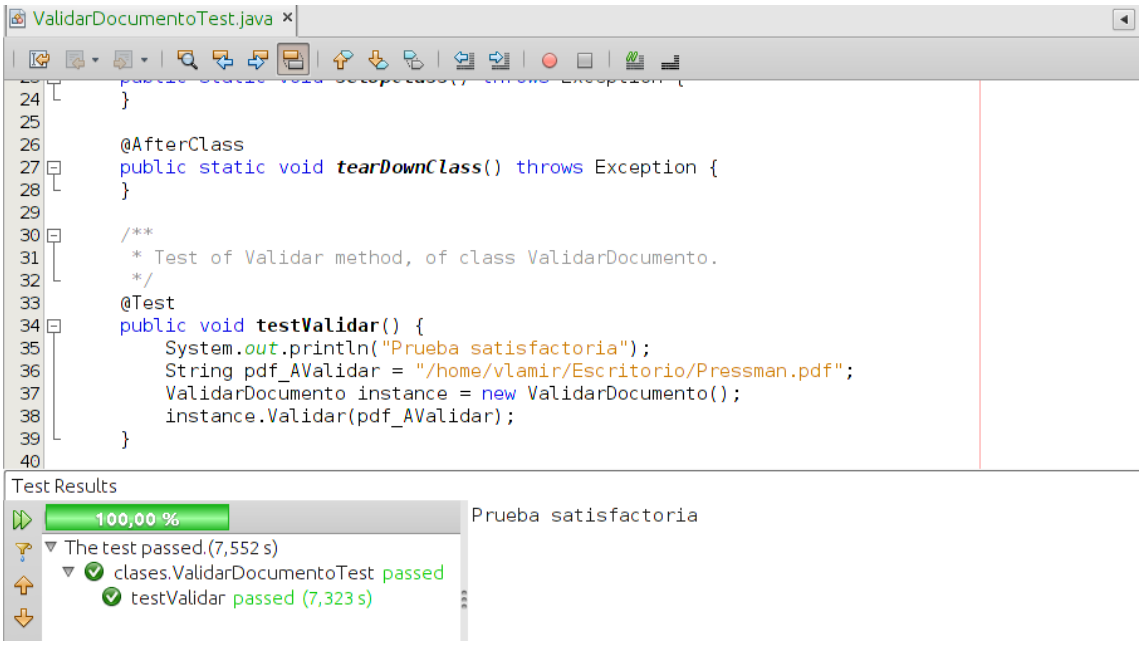


Figura 10. Caso de prueba del método Validar para el camino 1.

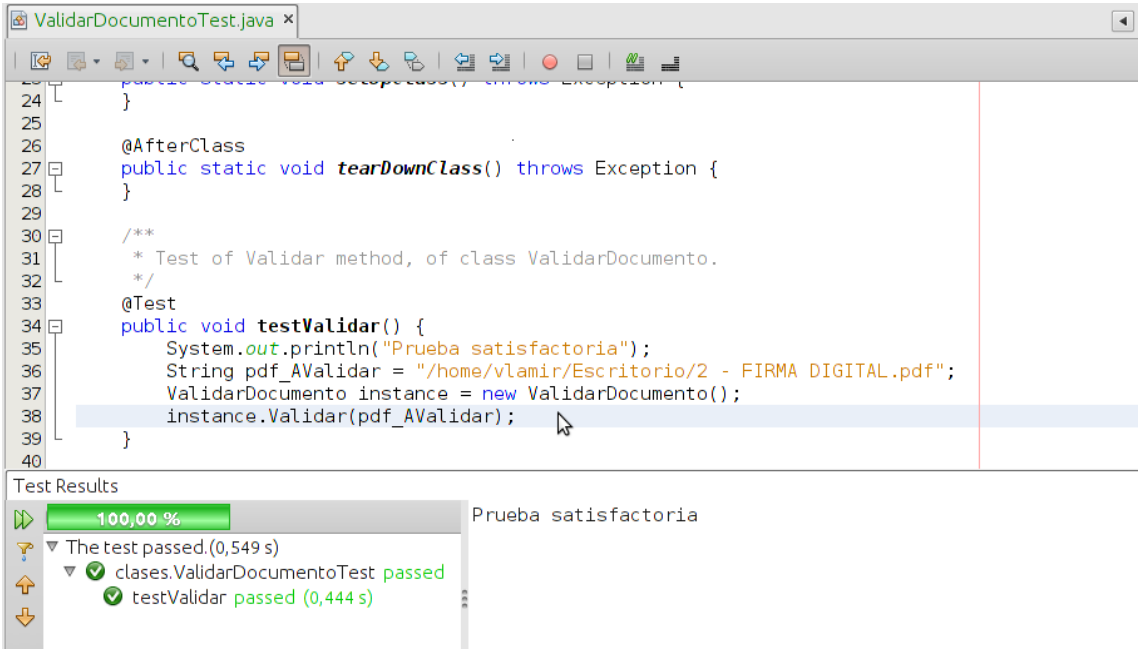


Figura 11. Caso de prueba del método Validar para el camino 2.