Universidad de las Ciencias Informáticas
Facultad 3

Título: Componente de firma digital para el proyecto Sistema de Gestión Fiscal.

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor: Adrian Matos Sterling

Tutor: Ing. Alain Hernández López

Mayo 2009

"Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte."

Ernesto Che Guevara

DECLARACIÓN DE AUTORÍA

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

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

Adrian Matos Sterling	Ing. Alain Hernández López
_____	_____
Firma de Autor	Firma del Tutor

DATOS DE CONTACTO

Datos de tutor:

Nombre: Ing. Alain Hernández López

Correo electrónico: ahernandezlop@uci.cu

Datos del autor:

Nombre: Adrian Matos Sterling

Correo electrónico: asterling@estudiantes.uci.cu

AGRADECIMIENTOS

A mis padres por todo el apoyo que me han dado y por saber guiarme hasta este momento.

A mi madrastra por tenerme como un hijo, gracias por todo y te quiero como a mi madre.

A mis tías por sus consejos y el cariño que siempre me han dado.

A mi familia por confiar en mí.

A mis compañeros de la UCI y los de la vieja guardia que siempre han estado ahí para lo que sea, no vamos a poner nombre porque son muchos… ustedes saben quienes son…

A SAHILY, sin ti no se por donde estuviera este trabajo, de verdad que no tengo palabras para agradecer lo que has hecho por mi en este tiempo. Ojala siempre estés ahí, no te asustes que no hay más tesis, pero en verdad que eres la mejor amiga que he tenido. Sigue así niña…

A mis primos y primas por sus consejos y por ser guías para mí.

A Oscar, gracias por ayudarme tanto.

A mi tutor, he aprendido muchas cosas trabajando contigo, gracias.

Arisniubis, gracias por todo el apoyo que me has dado y por tu paciencia.

DEDICATORIA

A mis padres Leonardo y Magda y a mi otra madre Soreidis, los quiero mucho a todos.

A mis hermanitas Elena y Yuneyris, ustedes son lo más grande que yo tengo, estudien mucho para que alcancen todo lo que se propongan en la vida.

RESUMEN

En la Fiscalía General de la República de Cuba se necesita un componente de firma digital que sirva para concederle integridad y autenticidad a los documentos que se generan en dicha entidad. En el presente trabajo se realiza un estudio de los principales algoritmos que se emplean para llevar a cabo la firma digital, además se describen las técnicas y herramientas utilizadas durante el desarrollo del componente. También se explica detalladamente como es el funcionamiento de la firma digital y la forma en que se realizará, debido a la carencia en Cuba de una Infraestructura de Clave Pública.

Este documento incluye además las etapas de Diseño e Implementación, donde se muestras algunos de los principales artefactos generados en cada fase. Finalmente se hace la evaluación de los resultados obtenidos mediante el empleo de métricas para el diseño orientado a objetos y la aplicación de pruebas de caja negra sobre la interfaz gráfica de usuario.

PALABRAS CLAVE

Firma digital, algoritmos, diseño, implementación

TABLA DE CONTENIDOS

INTRODUCCIÓN

Con el desarrollo alcanzado por las redes telemáticas e Internet es cada vez más común el intercambio de información de cualquier tipo, ya sea de contenido financiero, administrativo o judicial entre personas distantes geográficamente. Sin embargo, este marco de relaciones está expuesto a todo tipo de amenazas, lo cual constituye un grave problema en la seguridad de los datos.

En la práctica existen diversos métodos para obtener seguridad en los datos, firmar documentos digitalmente es uno de ellos, este mecanismo puede ser tan sencillo como insertar la imagen escaneada de una firma manuscrita en un documento creado con un procesador de textos, lo que no permite otorgarle validez jurídica a la firma, o muy avanzado como la firma digital que utiliza la criptografía de clave pública.

La firma digital es una secuencia de datos electrónicos que se obtienen de la aplicación a un mensaje determinado, un algoritmo de cifrado asimétrico o de clave pública. Esta equivale funcionalmente a la firma autógrafa en orden a la identificación del autor del que procede el mensaje. Desde un punto de vista material, la firma digital es una simple cadena o secuencia de caracteres que se adjunta al final del cuerpo del mensaje firmado digitalmente.

En nuestro país son innumerables las instituciones que deberían usar la firma digital como un mecanismo de seguridad, una de ellas es la Fiscalía General de la República de Cuba (FGRC) que es el Órgano del Estado al que corresponde, como objetivos fundamentales, el control y la preservación de la legalidad, sobre la base de la vigilancia del estricto cumplimiento de la Constitución, las leyes y demás disposiciones legales, por los organismos del Estado, entidades económicas y sociales y por los ciudadanos; y la promoción y el ejercicio de la acción penal pública en representación del Estado.

El proyecto Sistema de Gestión Fiscal (SGF) surge por la necesidad que tiene la FGRC de informatizar los principales procesos que en ella se realizan, a fin de garantizar, con mayor economía, profesionalidad y celeridad el cumplimiento de los objetivos de esta entidad.

Los principales procesos en los que interviene la FGRC son:

- Protección de los Derechos Ciudadanos.
- Procesos Penales.
- Verificación Fiscal.
- Gestión de Cuadros y Personal de Apoyo.
- Registro de Control Primario.

En estos procesos se genera y manipula un gran número de información, como son: expedientes de acusados, documentos de pruebas y testimonios. Esta información es clasificada y sensible, solo puede ser modificada por los fiscales que estén trabajando directamente con el proceso que se esté llevando a cabo. Por lo que no se puede alterar a no ser que sea por el personal autorizado, de manera que se necesita un componente que permita darle integridad y autenticidad a toda la información que se genere digitalmente, garantizando con esto que se pueda detectar cualquier modificación realizada en estos documentos.

Por lo antes expuesto es que se plantea la siguiente **situación problemática:** La necesidad que tiene el proyecto SGF de contar con un componente de firma digital que permita garantizar la integridad y autenticidad de la información que se genere en la FGRC. Lo cual nos conlleva a resolver el siguiente **problema científico:** ¿Cómo desarrollar un componente de firma digital que garantice la integridad y autenticidad de los documentos que se generen en la FGCR? El **objeto de estudio** de este trabajo será la Criptografía y como **campo de acción:** Componentes de firma digital.

Objetivo General:

Desarrollar un componente de firma digital para integrarlo al proyecto SGF que garantice la autenticidad e integridad de los documentos que se generan en la FGRC.

Objetivos específicos:

Construir un marco teórico sobre los diferentes componentes que se utilizan para realizar la firma digital en la Universidad de las Ciencias Informáticas y el mundo.

Desarrollar un componente que satisfaga los requerimientos planteados por la FGRC a partir de los resultados obtenidos en el marco teórico.

Evaluación del componente desarrollado.

Hipótesis:

Si se desarrolla un componente de firma digital para el proyecto SGF, entonces se podrá garantizar la autenticidad e integridad de los documentos que se generan en la FGRC.

Tareas de la investigación

1. Realizar un estudio de los principales algoritmos de encriptación de clave pública que existen.
2. Indagar sobre otras aplicaciones o soluciones similares.
3. Especificar los requerimientos necesarios para desarrollar un componente de firma digital.
4. Diseñar el componente de firma digital para el proyecto SGF.
5. Implementar el componente de firma digital para el proyecto SGF.
6. Evaluar el componente desarrollado.

Con el objetivo de darle cumplimiento a estas tareas se utilizarán los siguientes métodos científicos.

Teóricos

Histórico lógico: se emplea para realizar un estudio los mecanismos de firma digital existentes y la evolución de estos, unido a los distintos algoritmos de encriptación, para obtener un patrón en cuanto a la utilización de estos en la actualidad.

Analítico-sintético: utilizado para obtener los elementos más importantes relacionados con el objeto de estudio, mediante el estudio de manuales y documentos, que permitan llegar a conclusiones y determinar elementos que resulten importantes para el desarrollo de un componente de firma digital.

El presente documento está estructurado en tres capítulos, los que se describirán a continuación.

Capítulo 1: este capítulo contiene un estudio del estado del arte de la firma digital, así como los principales algoritmos que se emplean hoy en día para su desarrollo, además de un estudio de las principales metodologías existentes para el desarrollo de software, junto a algunos lenguajes de programación que se pueden utilizar para la implementación del sistema.

Capítulo 2: este capítulo contiene la propuesta de solución, se presenta el modelo de dominio, se realiza una especificación de los requisitos funcionales y no funcionales con los que debe cumplir el sistema. Se realiza el Diagrama de Casos de Uso del Sistema (DCUS) a partir de los Casos de Uso (CU) y actores determinados y finalmente se realiza una descripción de cada CU establecidos.

Capítulo 3: este capítulo contiene todo lo referente al diseño e implementación de la propuesta realizada en el capítulo 2. Contiene el estándar de codificación a seguir así como los diagramas de clases del diseño y los diagramas de secuencia más importantes. Además se abordará las pruebas que se le hicieron al componente y algunas métricas que se le aplicaron.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se hace una referencia a los principales elementos a tener en cuenta para desarrollar un componente que garantice la integridad y autenticidad de documentos digitales, se ofrecen algunos conceptos de importancia para darle cumplimiento al objetivo del trabajo. También se realiza un estudio del arte de las principales metodologías, plataformas de desarrollo y lenguajes de programación existentes, mostrando las características de cada uno que lo diferencia del resto.

1.2 Definición de firma digital

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

La firma digital se basa en el empleo de dos técnicas muy distintas. La primera es la criptografía asimétrica o de clave pública y la otra es el uso de funciones hash o resumen.

1.3 Tipos de firma digital

Las normas TS 101 733 y TS 101 903 propuestas por el ETSI (*European Telecommunications Standards Institute*) definen los formatos técnicos de la firma electrónica. La primera se basa en el formato clásico PKCS#7 (*Public Key Cryptography Standards*) y la segunda en XMLDsig (*Extensible Markup Language Digital Signature*).

Bajo estas normas se definen tres modalidades de firma:

- **Firma básica:** Incluye el resultado de operación de *hash* y clave privada, identificando los algoritmos utilizados y el certificado asociado a la clave privada del firmante.
- **Firma fechada o avanzada.** A la firma básica se añade un sello de tiempo calculado a partir del

hash del documento firmado por una TSA (*Time Stamping Authority*)

Firma reconocida o completa: A la firma avanzada, se añade información sobre la validez del certificado procedente de una consulta de CRL (*Certificate Revocation List*) o de OCSP (*Online Certificate Status Provider*) realizada a la Autoridad de Certificación.

1.4 Importancia y funcionamiento de la firma digital

Debido al notable desarrollo de las tecnologías y la sociedad de la información, se hace necesario el intercambio de información mediante la utilización de documentos digitales, como principal medio de almacenamiento de información.

Con el uso de la firma digital puede ser detectado cualquier modificación a los datos que han sido firmados. También ofrece la certeza de que el autor no puede retractarse en un futuro de lo expresado por él.

La firma digital sirve para identificar la identidad del firmante, para autentificar que el que rubrica dice ser quien es. Además es imposible que sea suplantada, como ocurre con la firma tradicional que es fácil de falsificar. A continuación se hace una demostración de estas técnicas.

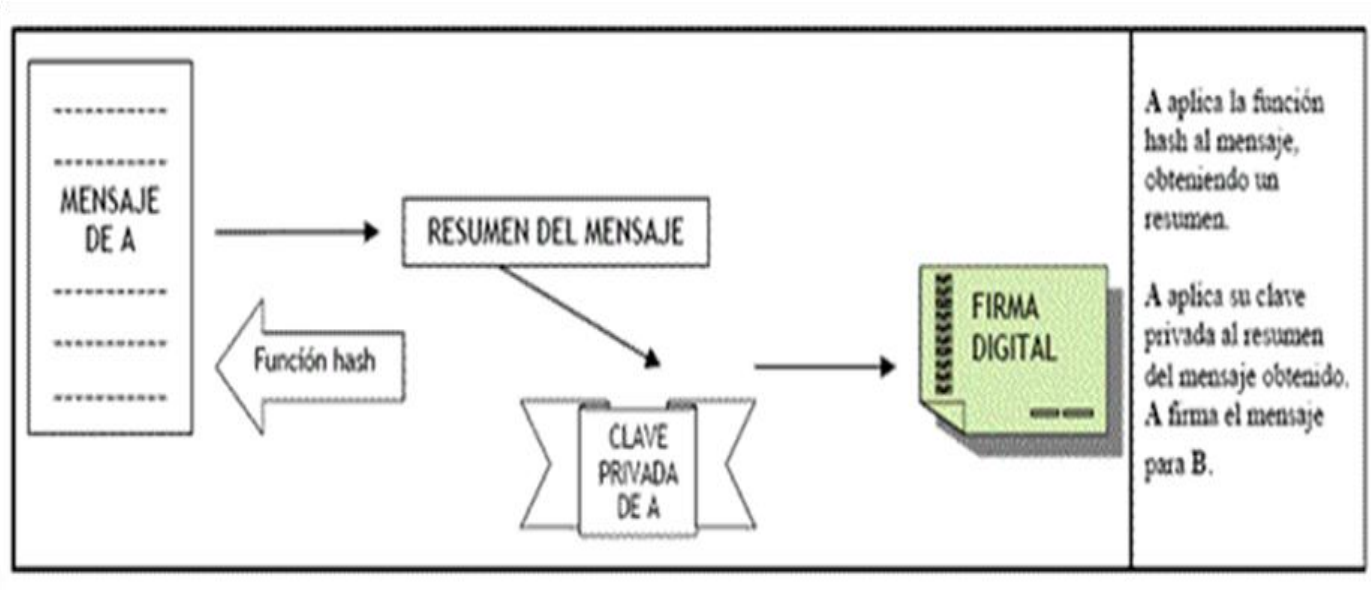


Figura 1.1: *Proceso para firmar digitalmente un documento.*

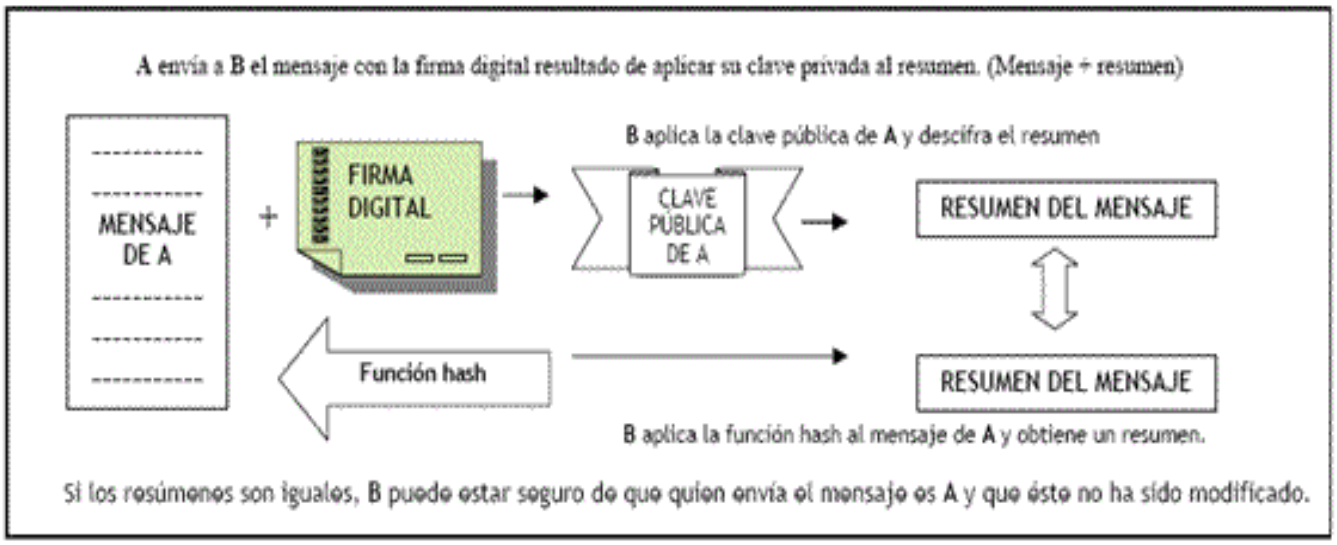


Figura 1.2: *Proceso para verificar la firma digital de un documento.*

Estas técnicas anteriormente descritas son correctas, pero indican un grave problema a nivel de seguridad, ya que de esta manera no se puede asegurar que una clave pública pertenece a un usuario determinado. Para esto es necesario poder vincular la clave pública de un usuario con su identidad, es por esta razón que surgen los certificados digitales que contiene entre otros datos la siguiente información:

- Identidad del usuario.
- Clave pública del usuario.
- Periodo de validez del certificado.
- Identidad de la Autoridad Certificadora (*CA por sus siglas en inglés*).
- Firma digital del certificado, generada por la CA.

La CA no es más que la entidad que asegura la identidad de los usuarios de los certificados digitales. Posee su propio par de claves y firma digitalmente todos los certificados generados por ella. La CA es la que procesa todas las peticiones de certificados a través de Autoridad de Registro (*RA pos sus siglas en inglés*), estas peticiones están compuestas por los datos y la clave pública del solicitante, además genera los certificados y los almacena en un repositorio público y gestiona la caducidad y revocación de

certificados.

Toda fiabilidad de la CA se basa en la inviolabilidad de su clave privada, la cual debe ser protegida empleando medios técnicos y humanos.

Para que un usuario solicite su propio certificado digital se debe llevar a cabo un proceso de registro para poder asegurar la identidad de dicho usuario y este proceso se realiza mediante una RA que es la entidad encargada de registrar las peticiones que hagan los usuarios para obtener un certificado, además comprueba la veracidad de la información provista por los usuarios y finalmente envía las peticiones a una CA para que sean procesadas.

Los sistemas antes descritos pueden englobarse en único sistema, denominado Infraestructura de Clave Pública (*PKI pos sus siglas en inglés*) que está compuesta por un conjunto de hardware, software, políticas de seguridad y procedimientos de seguridad que permiten la realización de operaciones criptográficas como el cifrado de datos y la firma digital, todo esto manteniendo el no repudio en origen.

Una PKI también está compuesta por una Autoridad de Sellado de Tiempo (TSA pos sus siglas en inglés) que se encarga de firmar documentos con la finalidad de probar que existían antes de una fecha determinada. También se compone por los repositorios, donde los más importantes son los repositorios de certificados que almacena todos los certificados válidos y el repositorio de Listas de Revocación de Certificados (*CRL por sus siglas en inglés*) donde se encuentran los certificados que por alguna razón dejaron de ser válidos antes de la fecha de expiración que presentan.

1.5 Criptografía de clave pública

Según la Real Academia Española la criptografía es el arte de escribir con clave secreta o de un modo enigmático. El objetivo de la criptografía es garantizar la seguridad en la comunicación entre dos entidades, además de asegurar que la información que se transmite es auténtica.

La criptografía de clave pública también llamada asimétrica consiste en el uso de un par de claves diferentes, privada y pública, que se le atribuye a una persona determinada y tiene como principales características:

- La clave privada debe permanecer secreta, es conocida sólo por la persona a quien se le ha atribuido el par de claves y se va a utilizar para cifrar los mensajes, mientras que la clave pública puede ser conocida por cualquier individuo.
- Ambas claves sirven para cifrar y descifrar mensajes.
- A partir de la clave pública no se puede obtener ni deducir la privada.

1.5.1 Algoritmo RSA

El algoritmo RSA fue creado en 1978 y debe su nombre a sus tres inventores Ronald Rivest, Adi Shamir y Leonard Adleman, estuvo bajo patente de los Laboratorios RSA hasta el 20 de septiembre del 2000. Es considerado uno de los algoritmos más seguros de la actualidad.

Este algoritmo puede ser usado para la encriptación de llave pública y firma digital. Se basa en la dificultad para factorizar grandes números. La clave privada y pública se obtiene a partir del producto de dos números primos grandes.

Si se desea generar el par de claves (Kp, Kp), se eligen aleatoriamente dos números primos grandes, p y q. Luego se calcula el producto $n = pq$.

Después se elige un número primo *e* relativo con $(p - 1)$ y $(q – 1)$. (n, p) será la clave pública. Nótese que debe tener inversa $mod\ (p -1)(q - 1)$**,** por lo que existirá un número *d* tal que $de = 1\ (mod(p - 1)(q - 1))$ decir que *d* es la inversa de *e*. (d, n) será la clave privada. Se puede calcular esta inversa mediante el Algoritmo Extendido de Euclides.

Para llevar a cabo el cifrado se emplea la expresión $c = m^e\ mod(n)$ y para el descifrado se utilizará $m = c^d\ mod(n)$ (Menezes, y otros).

Hoy en día RSA es el algoritmo asimétrico de cifrado más usado, tanto en conexiones de Internet y protocolos seguros, como en cifrado de datos. Las longitudes de clave usadas hoy en día varían desde los 512 bits hasta los 4096 bits, aunque se suelen tomar de forma habitual claves de 1024 bits puesto que las de 512 bits no se consideran suficientemente seguras.

Desventajas de RSA

A pesar de que el algoritmo RSA es bastante seguro, posee algunas deficiencias en la forma de utilizarlo que pueden ser aprovechadas por los atacantes. A continuación se muestran algunas de estas vulnerabilidades.

- **Claves demasiadas cortas**

Para que se considere segura una clave de RSA se recomienda que tenga una longitud de *n* no menor que 1024 bits. Esto se debe que en 1999 Adi Shamir desarrolló un dispositivo capaz de factorizar números con gran rapidez lo que ponía en peligro los mensajes encriptados con una clave con longitud menor o igual que 512 bits. La longitud de la clave está en función del tiempo que se quiera mantener la información en secreto, esto se debe al nivel de desarrollo que ha alcanzado la tecnología.

- **Ataques de texto en claro escogido**

Esta vulnerabilidad explota la posibilidad de que un usuario codifique y firme un único mensaje empleando el mismo par de claves.

- **Firmar y codificar**

Nunca se debe firmar un mensaje después de ser codificado, sino todo lo contrario, porque existen ataques que dan la posibilidad de manipular mensajes primero codificados y luego firmados.

1.5.2 Algoritmo ElGamal

Fue desarrollado en 1986 por T. ElGamal, en un principio la idea era utilizarlo para realizar firma digital, pero en la actualidad es muy utilizado en el cifrado de mensajes. Está basado en el problema de los logaritmos discretos.

En este algoritmo para generar un par de claves, se escoge un número primo *p* y dos números *g* y *x*. Utilizando la siguiente fórmula: $y = g^x \pmod{p}$.

Donde *(g, y, p)* es la clave pública y *x* es la privada.

Para firmar un documento utilizando este algoritmo, se debe escoger un número aleatorio k, te forma tal que *mcd(k; p-1) = 1* calcular $a = g^k \pmod p$.

Inmediatamente se emplea el algoritmo de Euclides para resolver la siguiente ecuación:

$m = (xa + kb) \pmod{(p - 1)}$.

La firma queda constituida por el par *(a, b)*. El valor *k* se debe mantener en secreto y ser diferente cada vez.

Para verificar la firma se comprueba que: $y^a a^b = g^m \pmod p$.

Mientras que para codificar un mensaje *m* se debe escoger un número primo *k* relativo con *(p - 1)*, que será mantenido en secreto. Entonces se procede a calcular las siguientes expresiones: $a = g^k \pmod p$ y $b = y^k m \pmod p$.

El texto cifrado queda confeccionado por el par *(a, b)*, que posee el doble de la longitud del mensaje original. Lo que representa el principal inconveniente de este sistema de cifrado.

Para llevar a cabo el proceso de descodificación se calcula $m = b/a^x \pmod p$

1.5.3 Criptografía de Curva Elíptica

Son los criptosistemas más recientes dentro del campo de los sistemas de clave pública y representan sólo otra forma de implementar métodos de logaritmo discreto.

En criptografía se habla de curvas elípticas se hace referencia a una ecuación $y^2=x^3+Ax+B$ siempre que sean considerados un cuerpo finito con características *p (con p>3).*

Las curvas elípticas tienen ciertas características que las hacen especiales en el mundo de la criptografía. Una de estas características consiste en la posibilidad de poder generar un punto en una curva partiendo de dos puntos dados (o incluso de uno). Las curvas elípticas pueden ser implementadas con gran eficiencia y pueden ser comparados con la velocidad de algoritmos como RSA y DSS (*Digital Signature Standard*). Aunque no se ha podido demostrar se cree que son bastante seguros.

1.6 Funciones Hash

Los mensajes que se firman digitalmente pueden ser de gran tamaño, lo que dificulta el proceso de cifrado, por lo cual no se cifra el mensaje completo sino un resumen obtenido de haberle aplicado una función hash o resumen.

Una función hash es un método para generar claves o llaves que identifiquen de manera unívoca a un documento. Estos métodos son muy variados, llegan a tomar en cuenta diversos parámetros tales como el nombre de un archivo, su longitud, hora de creación, datos que contenga, aplicándole diversas transformaciones y operaciones matemáticas. Estas funciones se caracterizan por:

- Su algoritmo es conocido públicamente.
- Es de un solo sentido: a partir del valor hash no se puede obtener los datos originales.
- Es muy poco probable obtener el mismo valor hash a partir de otros datos.

Son empleadas para:

- Identificar algún archivo de computadora independientemente de su nombre o ubicación.
- Comprobar que el archivo no ha sido modificado.
- Identificar un registro en una base de datos y permitir con ello un acceso más rápido a los registros.

Entre los algoritmos hash más utilizados se encuentran los siguientes:

1.6.1 Algoritmo MD5 (Message Digest 5)

Es uno de los algoritmos de signaturas más populares, resultado de un conjunto de mejoras realizadas al algoritmo MD4, diseñado por Ron Riverst. Produce resúmenes de 128 bits a partir de bloques de 512 bits del mensaje original.

En los últimos tiempos el algoritmo MD5 ha mostrado ciertas debilidades, aunque sin implicaciones prácticas reales, por lo que se sigue considerando en la actualidad un algoritmo seguro, si bien su uso tiende a disminuir. Es por esto que es considerado uno de los algoritmos hash más débiles.

1.6.2 Algoritmo SHA-1

SHA-1 (*Secure Hash Algorithm 1*) fue desarrollado por la NSA (*National Security Agency*) como parte del estándar DSS en 1994, es una ampliación al algoritmo SHA. Este algoritmo es considerado seguro y libre de puertas traseras. Produce firmas de 168 bits a partir de bloques de 512 bits.

Es similar al MD5, aunque un poco más lento, pero la mayor longitud de clave lo hace más resistente a ataques de colisión por fuerza bruta y de inversión.

A pesar de que se ha logrado demostrar que este algoritmo puede ser roto, esto no es una preocupación por el momento, porque con los métodos actuales encontrar una colisión en el SHA-1 tomaría cerca de 5 millones de años. Sin embargo SHA-1 sigue siendo muy utilizado.

1.7 Principales empresas proveedoras de firma digital

Para que la firma digital sea aplicada correctamente debe estar basada en una Infraestructura de Clave Pública (*PKI* por sus siglas en inglés), con el objetivo de proporcionar una mayor seguridad en la autenticación, integridad y no rechazo de los documentos.

En este momento existen muchas empresas que ofrecen este tipo de servicio, entre las que se encuentran:

- **Entrust**

Se introduce en el mercado de la PKI en el año 1994 y desde entonces ha llevado un proceso continuo de mejora de sus productos y la creación de otros, lo que lo ha convertido en un estándar en la industria de la certificación. Entre los productos desarrollados por esta empresa se encuentra Entrust/PKI, la cual está dividida en 7 componentes individuales: Entrust/Authority, el directorio, Entrust/RA, Entrust/AutoRA, Entrust/Profile Server, Entrust/Timestamp y Entrust/Entelligence. Este es un producto de muy altas prestaciones, con unas capacidades de gestión de certificados impresionantes apropiado para el nuevo modelo de sociedad basado en certificados digitales.

- **CERES**

El proyecto denominado CERES (CERtificación ESpañola) que lidera la Fábrica Nacional de Moneda y Timbre, y que en líneas generales, consiste en establecer una Entidad Pública de Certificación, que permita autentificar y garantizar la confidencialidad de las comunicaciones entre ciudadanos, empresas u otras instituciones y administraciones públicas a través de las redes abiertas de comunicación.

El objetivo principal de este proyecto es garantizar la identidad de los ciudadanos y Administraciones que participen en una comunicación, así como la confidencialidad e integridad de los mensajes que se envíen entre ellos.

Entre los servicios que ofrece se encuentran los de certificación, servicios avanzados como el voto electrónico, validación de firmas y presta servicios de tercera parte de confianza, entre los que se encuentra el sellado de tiempo y la custodia documental.

- **Silanis Technology**

Es una empresa que automatiza los procesos empresariales que requieren seguro, compatible y legalmente exigibles, las firmas electrónicas y registros. Son considerados pioneros en el mundo de la firma electrónica desde hace más de 17 años.

Silanis ha ayudado a dar forma a la firma electrónica del mercado mediante una acción sostenida de programas educativos y la participación en asociaciones de la industria.

Como miembro fundador de la Firma Electrónica & Records Association (ESRA por sus siglas en inglés) y el Comité de Normas y Procedimientos de registros electrónicos y firmas (SPERS), Silanis sigue apoyando a la creciente comunidad de usuarios de firma electrónica.

1.8 Metodologías de desarrollo de software

Para obtener un producto que cumpla con todas las especificaciones y esté en el tiempo estimado, se necesita una metodología que guie el proceso de desarrollo de software.

Una metodología es la que define Quién debe hacer Qué, Cuándo y Cómo. Es un proceso donde se realizan tareas para obtener un producto a través de un conjunto de actividades definidas previamente.

En la actualidad no existe una metodología universal que se le pueda aplicar a todos los proyectos y su selección depende de las características de cada proyecto. A continuación se hace un breve resumen de algunas de ellas.

1.8.1 SCRUM

Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle (H. Canós, y otros, 2003). Es una metodología ágil indicada para proyectos en entornos complejos, donde se necesita obtener resultados rápidamente y con un continuo cambio de requisitos. En SCRUM el desarrollo de software se realiza a través de iteraciones o sprints que pueden oscilar entre dos o cuatro semanas, en dependencia de lo que se desee. En cada iteración se debe obtener un incremento del producto para ser mostrado al cliente. Otra característica de esta metodología es que el equipo se debe reunir diariamente durante 15 minutos, donde cada miembro deberá mostrar lo que ha hecho desde la última reunión, lo que hará a partir de esta y que inconvenientes tiene o se le pueden presentar para realizar su tarea.

Beneficios de SCRUM (Albaladejo).

1. Entrega mensual o quincenal de resultados, lo que produce los siguientes resultados.
    - Gestión regular de las expectativas del cliente y basada en resultados tangibles.
    - Resultados anticipados.
    - Flexibilidad y adaptación.
    - Gestión sistemática del Retorno de Inversión.
    - Mitigación sistemática de los riesgos del proyecto.
2. Productividad y calidad.
3. Alineamiento entre el cliente y el equipo de desarrollo.
4. Equipo motivado.

1.8.2 XP

XP (Extreme Programming) es una metodología ágil creada por Kent Beck, se centra en el desarrollo de las relaciones interpersonales para logar un mayor éxito en el desarrollo de software, potenciando el trabajo en equipo y preocupándose por la superación continua de los desarrolladores.

Entre los objetivos de XP está la satisfacción del cliente, proporcionándole el producto que necesita y cuando lo necesita. También es objetivo elevar al máximo el trabajo en grupo, ya que tanto los jefes, desarrolladores y clientes están vinculados en el proceso de desarrollo de software y este a su vez consiste en 5 pasos fundamentales (H. Canós, y otros, 2003).

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.

En estas etapas no se recomienda forzar al desarrollador a realizar trabajo que no estaba entre sus tareas, ya que esto puede disminuir la calidad del producto o una demora en la entrega de la tarea.

Seis son las fases del ciclo de vida de XP:

1. Exploración
2. Planificación de la Entrega
3. Iteraciones
4. Producción
5. Mantenimiento
6. Muerte del Proyecto

1.8.3 RUP

El Proceso Unificado de Desarrollos (*RUP* por sus siglas en inglés) como lo dice su nombre es un proceso de desarrollo de software que ha unificado técnicas de desarrollo a través del Lenguaje de Unificado de Modelado (*Unified Modeling Language, UML* (Booch, y otros, 2005))

En RUP las actividades se realizan en a través de 9 flujos de trabajos y 4 fases. Ver figura 1.3.

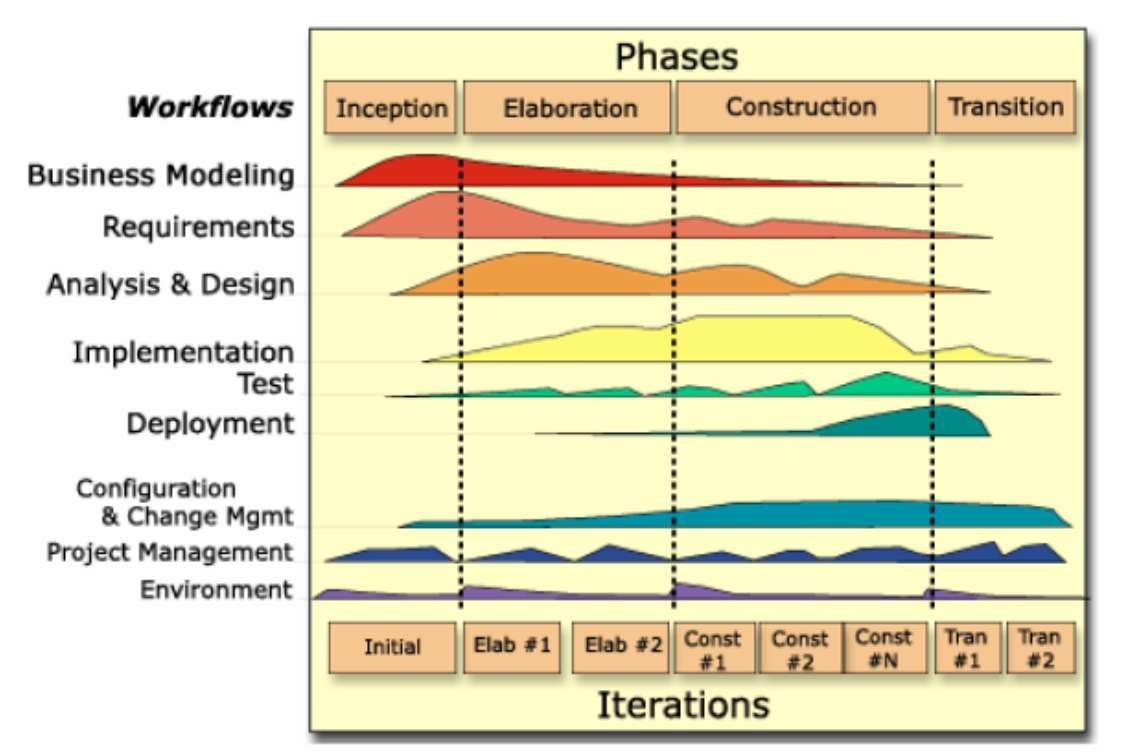


***Figura 1.3**: RUP en dos dimensiones*

Las características que hace a RUP único son tres:

1. *Dirigido por casos de uso:* Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante (Booch, y otros, 2005). Es el reflejo de lo que los clientes necesitan y desean. Además de constituir una herramienta para especificar los requisitos de un sistema, estos guían el diseño, implementación y prueba, es decir, guían el proceso de

desarrollo.

2. *Centrado en la arquitectura:* La arquitectura de software engloba aspectos estáticos y dinámicos más significativos del sistema, esta muestra una visión común del sistema con la que el equipo de proyecto y los clientes deben estar de acuerdo. Para darle inicio al desarrollo del sistema se comienza por los casos de uso más relevantes desde el punto de vista de la arquitectura.
3. *Iterativo e incremental:* RUP propone que el trabajo se divida en partes más pequeñas denominadas miniproyectos. Cada miniproyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en el flujo de trabajo y los incrementos al crecimiento del producto.

1.9 Lenguaje Unificado de Modelado

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software (Booch, y otros, 2000)

Es un conjunto de herramientas, que permite modelar sistemas orientados a objetos y se ha convertido en un estándar industrial respaldado por OMG (*Object Management Group*).

UML no es un método de desarrollo, es decir, no es una guía para realizar el análisis y diseño orientado a objetos, es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos.

Le puede dar soporte a una metodología de desarrollo de software de distintas formas, pero no define una metodología o proceso a usar. UML está constituido por un conjunto de diagramas proporcionando un estándar que muestra diversas perspectivas de un sistema, siendo comprensibles para los desarrolladores, clientes y todos aquellos que estén involucrados en el proceso de desarrollo.

Diagramas UML

- Diagrama de clases
- Diagrama de objetos
- Diagrama de casos de uso
- Diagrama de interacción
- Diagrama de estados

- Diagrama de actividad
- Diagrama de componentes
- Diagrama de despliegue

1.10 Herramientas CASE

Las herramientas CASE (*Computer Aided Software Engineering*) que sirven de ayuda en el ciclo de desarrollo de un software, también facilitan la automatización del ciclo de vida del desarrollo de sistemas informáticos, completamente o en algunas de sus fases, lo que trae consigo un incremento en la velocidad de desarrollo.

Estas herramientas le permiten a los analistas tener mucho más tiempo para el análisis y diseño, minimizando el tiempo para codificar y probar. Obtener una mejor calidad en el producto es una de las ventajas de este tipo de herramientas, así como alcanzar una mayor productividad, también posee otras ventajas:

- Automatiza la realización de diagramas.
- Genera estructura de código.
- Ayuda a la creación de relaciones en la base de datos.
- Ayuda a la documentación del sistema.

En la actualidad existen muchas herramientas CASE, a continuación se hace un breve resumen de algunas de ellas.

1.10.1 Enterprise Architect

Esta herramienta CASE es considerada una de las más potentes y flexibles para sistemas operativos Windows ya que soporta el ciclo completo de desarrollo de software, es basada en UML para crear sus diagramas, además posee un gran número de características entre las que se encuentran:

- Ayuda comprensiva para UML 2.1.
- Interfaz de usuario intuitiva.
- Documentación flexible y comprensible.

- Ingeniería de código directa e inversa.
- Plugins para vincular Enterprise Architect a Visual Studio.NET o Eclipse.
- Modelado de base de datos.
- Soporta control de versiones.
- Soporte en administración de requisitos.
- Soporta seguridad de usuario.
- Soporte para prueba.
- Soporte para el mantenimiento.
- Soporte para información de estado del sistema.

1.10.2 Visual Paradigm for UML

Es una herramienta CASE que utiliza UML como lenguaje de modelado, ha sido creado para todas las personas que estén interesadas en el diseño de software orientado a objetos. Con esta herramienta es muy fácil la creación de diferentes diagramas UML, también soporta el ciclo de vida completo del desarrollo de software. Visual Paradigm ayuda a obtener productos con una mayor calidad a un menor coste, además posibilita generar código a partir de diagramas y generar documentación.

Entre sus principales características se encuentran:

- Soporta la última notación UML 2.1.
- Modelado colaborativo con CVS y Subversion.
- Editor de detalles de casos de uso.
- Generación de bases de datos.
- Ingeniería inversa de bases de datos.
- Generador de informes para generación de documentación.
- Multiplataforma.
- Provee soporte para la generación de código PHP.

1.10.3 Rational Rose Enterprise Edition 2003

Es una herramienta CASE que soporta UML, es muy utilizada debido a que se encarga de llevar a cabo tanto la automatización de los sistemas para la posterior generación de código, como para labores de ingeniería inversa. Cubre todo el ciclo de desarrollo de un proyecto.

Además presenta otras características entre las que se encuentran:

- Soporte a modelos de análisis, ANSI C++, Rose J y Visual C++.
- Generación de código en lenguaje Ada, ANSI C++, C++, CORBA, Java y Visual Basic, con funciones configurables de sincronización entre los modelos y el código.
- Funciones de análisis de calidad de código.
- Complemento de modelado Web que incluye funciones de visualización, modelado y herramientas para desarrollar aplicaciones Web.
- Posibilidad de publicar en la Web modelos e informes para mejorar la comunicación entre los miembros del equipo.
- Modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos.

A pesar de estas características esta herramienta solo puede ser utilizada en sistemas operativos Windows y se necesita una licencia para su uso.

1.11 Lenguajes de programación

Un lenguaje de programación es una herramienta que permite el desarrollo de software. Está constituido por un grupo de reglas gramaticales, un grupo de símbolos utilizables, un grupo de términos monosémicos y una regla principal que resume las demás.

Entre los lenguajes de programación que existen en la actualidad se encuentran:

1.11.1 Java

Java es un lenguaje de programación orientado a objetos desarrollado por *Sun Microsystem* a principios de los años 90. Este lenguaje toma mucha sintaxis de C y C++, pero elimina herramientas de bajo nivel que inducen muchos errores como es la manipulación directa de punteros y memoria.

Entre sus principales características están:

- Gestión de memoria automáticamente.
- No permite el uso de técnicas de programación inadecuadas.
- Posee mecanismos de seguridad incorporados
- Herramientas de documentación incorporadas.
- Independiente de la plataforma.
- Robusto.
- Es un lenguaje de objetos, en Java todo es un objeto.

1.11.2 C++

Es un lenguaje de programación orientado a objetos. Surge en 1980 de la mano de Bjarne Stroustrup, con el objetivo de añadirle al lenguaje de programación C nuevas características que lo hacen muy original, incorpora clases, funciones virtuales, tipos genéricos y expresiones, declaración de variables en cualquier punto del programa y otras características que hasta el momento no se conocían, herencia múltiple y funciones virtuales puras.

Debido a esto C++ posee en la actualidad una notable aceptación en el mundo de la informática, siendo uno de los lenguajes clásicos de programación, utilizándose en la creación de sistemas operativos, compiladores y aplicaciones.

1.11.3 PHP

PHP (acrónimo de *Hypertext Preprocessor*) es un lenguaje de interpretado de alto nivel, habitualmente se usa para el desarrollo de aplicaciones web, pero también puede ser empleado en la creación de otros tipos de programas como aplicaciones con interfaz gráfica usando bibliotecas como GTK+.

Este es uno de los lenguajes más utilizados en la actualidad y se debe a sus características, entre las que se encuentran:

- Es un lenguaje multiplataforma.
- Posee una amplia documentación en su página oficial (The PHP Group), entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Biblioteca nativa de funciones.

GTK+: Es un grupo de librerías para el desarrollo de Interfaces Gráficas de Usuario (GUI por sus siglas en inglés), mayormente utilizado en los entornos gráficos GNOME, XFCE y ROX de sistemas Linux.

Esta herramienta se conforma de 3 librerías fundamentales:

1. **ATK**: es una librería que permite a los programadores crear interfaces con características de gran accesibilidad que puede ser muy importante para personas con problemas (discapacitados o minusválidos). Contiene útiles como lupas de aumento, entradas de datos alternativas al clásico teclado o ratón y lectores de pantalla.
2. **Pango**: es una librería para el diseño y renderizado de texto, hace hincapié especialmente en la internacionalización. Es el núcleo para manejar las fuentes y el texto de GTK+ 2 (Pango).
3. **Glib**: librería de bajo nivel que proporcionar el manejo de estructuras de datos para C, portabilidad, interfaces para funcionalidades de tiempo de ejecución como ciclos, hilos, carga dinámica y un sistema de objetos.

1.11.4 Python

Python es un lenguaje de programación orientado a objetos, es un lenguaje interpretado o de script, es decir que se ejecuta mediante un programa intermedio llamado intérprete, a pesar de que su ejecución es más lenta que la de los lenguajes compilados, esta característica lo hace más flexible y más portable. A

pesar de esto Python puede considerarse un lenguaje semi interpretado porque presenta muchas características de los lenguajes compilados.

También es considerado un lenguaje con tipado dinámico por lo que no es necesario declarar el tipo de dato de la variable ya que su tipo se determinará en tiempo de ejecución según el tipo de valor que se le asigne a la variable. Además es un lenguaje fuertemente tipado debido a que no se permite tratar a una variable como si fuera de un tipo distinto, es necesario convertir de forma explícita dicha variable al nuevo tipo previamente. Otra de las ventajas que ofrece Python es que es multiplataforma.

1.12 Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado (*IDE* por sus siglas en inglés) es una aplicación informática destinada a los programadores que incluye un número de herramientas que le facilita el trabajo a los desarrolladores, tales como un editor de texto, un compilador, depuradores de código, administración de proyectos, integración con sistemas controladores de versiones o repositorios y algunos poseen un constructor de interfaz gráfica.

Es posible con un IDE programar en varios lenguajes, tal es el caso de Eclipse que mediante el uso de plugin se le pueden añadir soporte para otros lenguajes.

1.12.1 Eclipse

Eclipse es un IDE de código abierto independiente de una plataforma para desarrollar. Esta plataforma, ha sido usada para desarrollar un IDE, como el llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se embarca como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).

Este entorno de desarrollo emplea módulos (plug-in) para proporcionar toda su funcionalidad frente de la plataforma de cliente rico, lo que le permite extenderse usando otros lenguajes de programación como PHP, Python, C++ a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no.

Por ejemplo el plugin PDT (*PHP Development Tools*) es usado para el desarrollo en PHP y es de ayuda también en lenguajes para web como HTML y CSS.

Eclipse presenta muchas características entre las que se encuentran:

- Editor de texto.
- Resaltado de sintaxis.
- Compilación en tiempo real.
- Pruebas unitarias con JUnit.
- Control de versiones con CVS.
- Asistentes para creación de proyectos, clases y test.
- Refactorización.

1.13 Conclusiones

Después de haber realizado un estudio de la firma digital y la necesidad de poder contar con un componente que permita comprobar la integridad de los documentos que se generan en la FGRC se ha decidido usar para el desarrollo de la aplicación:

- *RUP* como metodología de desarrollo.
- *Visual Paradigm for UML* como herramienta CASE.
- *UML* como lenguaje de modelado.
- *PHP* como lenguaje de programación que se adapta a las necesidades de la FGRC.
- *Eclipse* como entorno de desarrollo unido al plugin *PDT* que permite la integración con PHP.
- *RSA* como algoritmo generador de claves asimétricas y para la encriptación de datos.
- *SHA-1* como algoritmo para calcular el resumen hash de los documentos.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En este capítulo se esboza la propuesta de solución para el desarrollo de un componente de firma digital. Para ello se comienza con la realización del modelo de dominio, así como el glosario de términos correspondiente, en el cual se definen los principales conceptos que se emplearán. Luego se puntualizan los requisitos funcionales y no funcionales que debe cumplir el sistema. También se identifican actores y como interactúan con el sistema mediante los casos de usos, que son representados en un diagrama y luego se hace una detallada descripción de cada uno de ellos.

2.2 Propuesta a desarrollar

Actualmente Cuba no cuenta con una PKI, a pesar de esto existen numerosas instituciones que necesitan de una infraestructura que permita otorgarle integridad y autenticidad en los documentos digitales que manipulan. La FGRC es una de estas instituciones y para darle cumplimiento a parte de sus necesidades es que hace esta propuesta de solución, que consiste en desarrollar un componente de firma digital capaz de proveer a los documentos digitales una firma por la cual se pueda identificar si dicho documento ha sido modificado, también debe garantizar que el documento ha sido firmado por una persona determinada.

Una PKI como se explicó en el capítulo anterior engloba varias entidades que funcionan como un todo, debido a la carencia de estas entidades en Cuba es que se decide crear un componente que permita unificarlas a todas.

El componente desarrollado debe dar la posibilidad al usuario de generar su par de claves, donde la clave pública se guardará en una base de datos para ser accedida por otros usuarios a la hora de verificar una firma digital, mientras que la clave privada será guardada en un archivo externo, encriptado por una contraseña introducida por el usuario.

También el componente deberá firmar cualquier tipo de documento y guardará la firma digital en un archivo separado. A la hora de verificarla se debe seleccionar el documento y su firma para que la

aplicación verifique si el documento fue modificado y debe de mostrar los datos de la persona que firmo dicho documento.

En cuanto a la seguridad del componente, antes de acceder a cualquier funcionalidad, el usuario deberá autenticarse, de lo contrario no conseguirá realizar ninguna tarea, además las operaciones a efectuar en base de datos se realizarán a través de varios usuarios en dependencia de los permisos que tengan los usuarios en la aplicación y la acción que se ejecute. A la hora de guardar la clave privada en el archivo externo, esta será encriptada como se especificó anteriormente. Con las medidas mencionadas se pretende dar un mayor nivel de seguridad a la aplicación a desarrollar.

2.3 Modelo conceptual o modelo del dominio

Un modelo del dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las cosas que existen o los eventos que suceden en el entorno en el que trabaja el sistema.

Muchos de los objetos del dominio o clases pueden obtenerse de una especificación de requisitos o mediante la entrevista con los expertos del dominio. Las clases del dominio aparecen en tres formas típicas:

- Objetos del negocio que representan cosas que se manipulan en el negocio, como pedidos, cuentas y contratos.
- Objetos del mundo real y conceptos de los que el sistema debe hacer un seguimiento, como la aviación enemiga, misiles y trayectorias.
- Sucesos que ocurrirán o han ocurrido, como la llegada de un avión, su salida y la hora de la comida.

El modelo del dominio se describe mediante diagramas UML (especialmente mediante diagramas de clases).

Estos diagramas muestran a los clientes, usuarios, revisores y a otros desarrolladores las clases del dominio y cómo se relacionan unas con otras mediantes asociaciones (Booch, y otros, 2005).

En pocas palabras el objetivo del modelado del dominio es contribuir a la comprensión del contexto del sistema, por lo tanto también contribuir a la comprensión de los requisitos del sistema que se desprenden de este contexto.

2.4 Modelo del dominio propuesto

Se ha decidido realizar un modelo del dominio ya que no se considera necesario realizar un modelamiento completo del negocio debido a que no se pueden determinar con claridad las fronteras del negocio y los procesos del negocio no están claramente definidos.

A continuación se muestra el modelo del dominio propuesto.

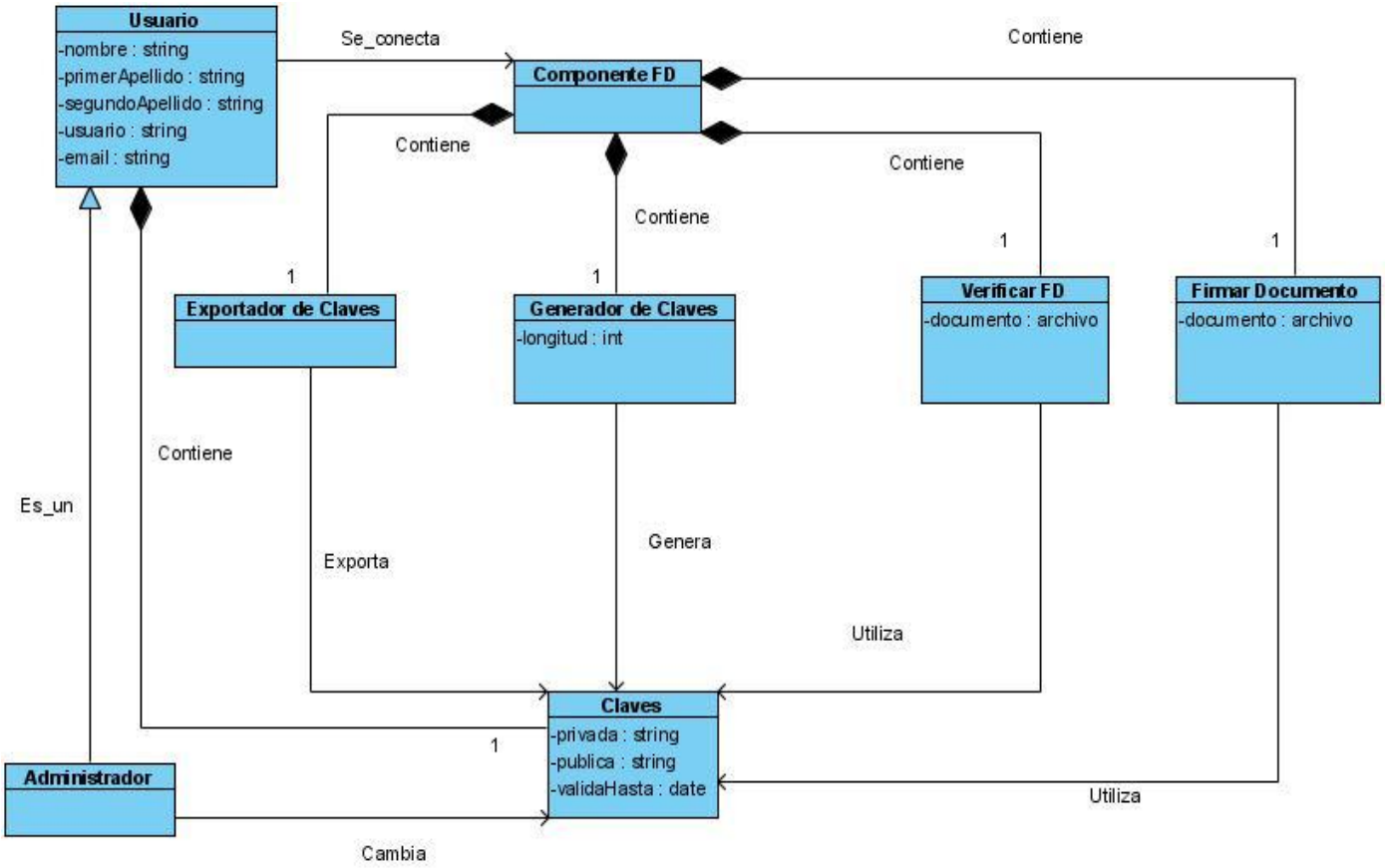


Figura 2.1: *Modelo de Dominio*

2.5 Glosario de términos

En ocasiones, como en los dominios de negocio muy pequeños, no es necesario desarrollar un modelo de objetos para el dominio; en su lugar, puede ser suficiente un glosario de términos.

El glosario de términos y el modelo del dominio ayudan a los usuarios, clientes, desarrolladores y otros interesados a utilizar un vocabulario común. La terminología común es necesaria para compartir el conocimiento con los otros. El glosario es muy importante para lograr un acuerdo entre los desarrolladores, en lo referente a la definición de los diferentes conceptos y nociones, y para disminuir en general, el riesgo de confusiones. En el glosario se definirán los términos comunes útiles para describir el sistema.

A continuación se presenta el glosario de términos.

Administrador: Persona encargada de la administración de la aplicación.

Claves: Par de claves pública y privada.

Componente de FD: Aplicación encargada de gestionar el proceso de creación de la firma digital, así como la generación y cambio de claves.

Exportador de claves: interfaz gráfica que le permite al usuario exportar en un fichero externo sus claves.

Firmar Documento: interfaz gráfica que le permite al usuario firmar digitalmente un documento.

Generador de claves: interfaz gráfica que le permite al usuario generar sus claves.

Usuario: persona que utilizará la aplicación.

Verificar FD: interfaz gráfica que le permite al usuario verificar la firma digital de un documento.

Principales eventos

Cambia: proceso mediante el cual el administrador cambia las claves de un usuario determinado.

Contiene: expresa que A está contenido lógicamente en B. Ejemplo Componente de FD contiene Exportador de Claves, Generador de Claves, Verificar FD y Firmar Documento.

Es un: se establece entre dos conceptos cuando el primero es un ejemplar especializado del segundo. Ejemplo un administrador es un usuario

Exporta: proceso mediante el cual el usuario exporta sus claves a un archivo.

Genera: proceso mediante el cual el usuario genera sus claves.

Se conecta: proceso mediante el cual el usuario se conecta a la aplicación.

Utiliza: proceso mediante el cual el usuario utiliza una o las dos claves para firmar un documento o verificar una firma digital.

2.6 Modelo del sistema

En el presente epígrafe se definen los requisitos funcionales y no funcionales que el sistema deberá cumplir y se hace el modelamiento del sistema en términos de casos de uso.

2.6.1 Requerimientos funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. A continuación se muestran los requerimientos funcionales del sistema:

R1 Autenticar usuario.

R2 Generar claves.

R3 Guardar claves.

R4 Exportar claves.

R4.1 Definir la ubicación del archivo a exportar.

R5 Cargar documento.

R5.1 Definir la ubicación del documento a cargar.

R6 Firmar documento.

R7 Guardar documento firmado.

R8 Verificar firma.

2.6.2 Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. A continuación se muestran los requerimientos no funcionales del sistema:

Usabilidad

- Debe poseer una interfaz agradable para el cliente.
- Contendrá una ayuda que instruirá al usuario en el trabajo con el sistema.

Rendimiento

- El sistema debe requerir un consumo mínimo de recursos.

Portabilidad

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

Diseño e implementación

- El lenguaje de programación es PHP 5.2.0+GTK 2.0.
- La herramienta IDE de desarrollo utilizada será Eclipse 3.3.1.1+PDT.
- La herramienta case utilizada es el Visual Paradigm 6.0.

- La herramienta gestor de base de datos es el PostgreSQL 8.3.

Software

PC cliente

- Sistema Operativo Microsoft Windows 2000 o superior; Sistema Operativo Linux con ambiente grafico KDE o GNOME.
- Librerías GTK 2.0 o superior y PHP 5.2.0 o superior.

PC servidor

- Gestor de base de datos PostgreSQL 8.3.

Hardware

- Memoria RAM de 128 Mb. o superior.

Apariencia e interfaz externa

- La interfaz debe ser sencilla y amigable de manera que potencie la comodidad del usuario para su trabajo.

2.6.3 Modelo de Casos de Uso del Sistema

El Modelo de Casos de Uso del Sistema es un modelo del sistema que contiene actores, casos de uso y sus relaciones.

2.6.3.1 Actores del sistema

Son terceros fuera del sistema que interactúan con él y a su vez pueden ser parte de él.

Actores	Justificación

Usuario	Representa a cualquier persona que interactúa con el sistema y tiene permiso para generar el par de claves, firmar documentos y verificar firmas.
Administrador	Representa a la persona que interactúa con el sistema, tiene todos los privilegios de un usuario, pero también puede cambiar las claves de los usuarios.

Tabla 2.1 *Actores del sistema*

2.6.3.2 Casos de uso del sistema

Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre las condiciones y requisitos que debe cumplir el sistema. A continuación se muestran los casos de uso del sistema.

CU_1	Autenticar usuario
Actor	Usuario
Descripción	El caso de uso se inicia cuando el usuario decide entrar a la aplicación para realizar alguna operación.
Referencia	R1

Tabla 2.2 *Resumen del CU Autenticar usuario*

CU_2	Generar claves
Actor	Usuario
Descripción	El caso de uso se inicia cuando el usuario accede por primera vez a la aplicación y desea generar su par de claves.
Referencia	R2, R3

Tabla 2.3 *Resumen del CU Generar claves*

CU_3	Exportar claves

Actor	Usuario
Descripción	El caso de uso se inicia cuando el usuario desea guardar sus claves en un archivo externo.
Referencia	R4

Tabla 2.4 *Resumen del CU Exportar claves*

CU_4	Firmar documento
Actor	Usuario
Descripción	El caso de uso se inicia cuando el usuario desea firmar un documento.
Referencia	R5, R6, R7

Tabla 2.5 *Resumen del CU Firmar documento*

CU_5	Verificar firma
Actor	Usuario
Descripción	El caso de uso se inicia cuando el usuario desea verificar la firma de un documento.
Referencia	R5, R8, R9

Tabla 2.6 *Resumen del CU Verificar firma*

CU_6	Buscar usuario
Actor	Usuario
Descripción	El caso de uso se inicia cuando el usuario desea buscar un usuario determinado.
Referencia	R9

Tabla 2.7 *Resumen del CU Buscar usuario*

CU_7	Cambiar claves

Actor	Administrador
Descripción	El caso de uso se inicia cuando el administrador desee cambiarle la clave a un usuario determinado.
Referencia	R2, R9, R10

Tabla 2.8 *Resumen del CU Cambiar claves*

2.6.3.3 Diagrama de Casos de Uso del Sistema

El Diagrama de Casos de Uso del Sistema (DCUS) es una representación grafica de los procesos y su interacción con los actores. A continuación se muestra el DCUS propuesto.

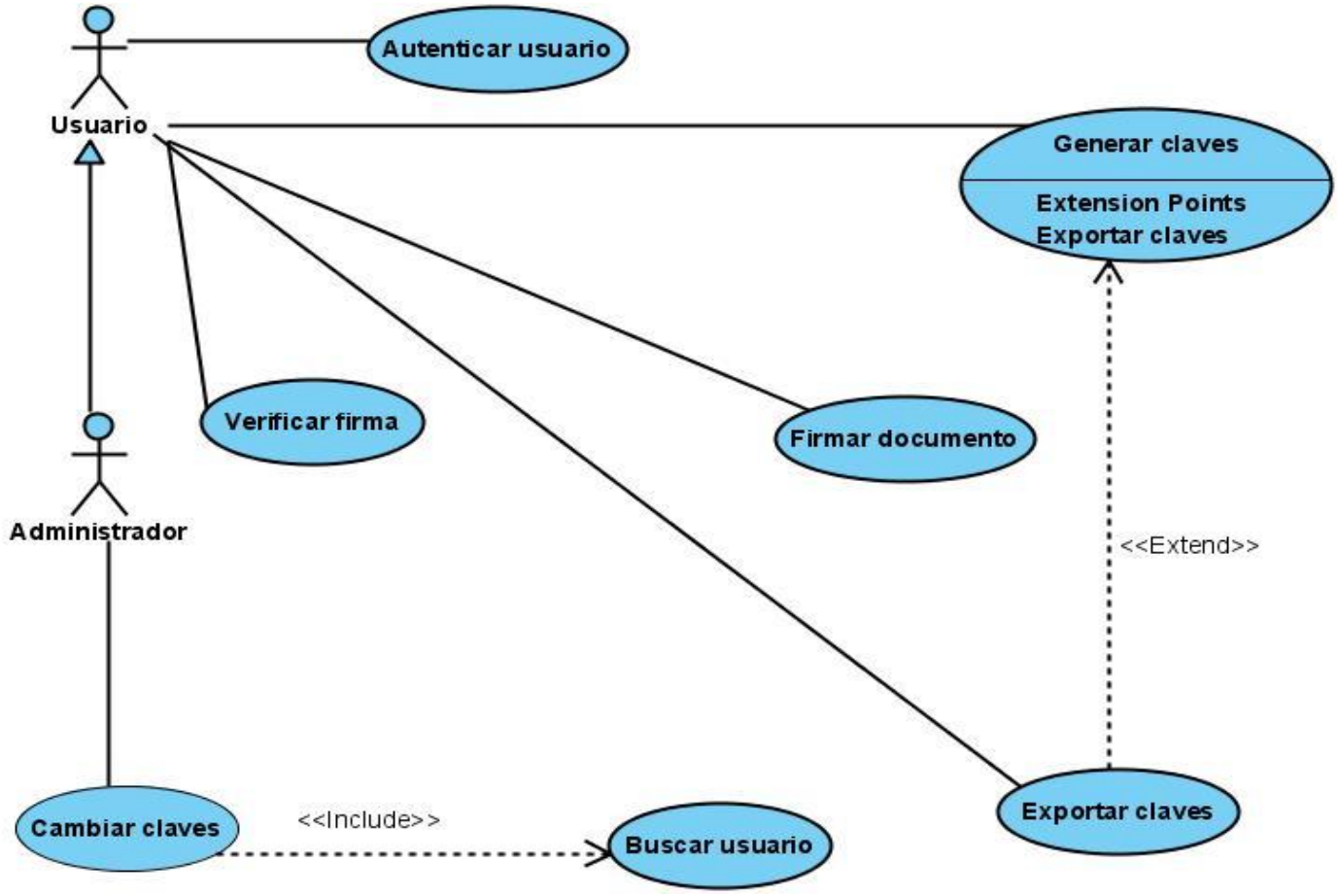


Figura 2.2: *Diagrama de Casos de Uso del Sistema*

2.6.3.4 Descripción de los CU

Para comprender el funcionamiento de cada caso de uso no es suficiente con la representación gráfica del Diagrama de casos de uso. La descripción de los casos de uso puede ser elaborada de forma breve o extendida y debe ir acompañada del prototipo de interfaz de usuario correspondiente. A continuación se describen los casos de uso del sistema:

<table>
<tr><td>Caso de uso:</td><td colspan="2">Autenticar usuario</td></tr>
<tr><td>Actores:</td><td colspan="2">Usuario</td></tr>
<tr><td>Resumen:</td><td colspan="2">El caso de uso se inicia cuando el usuario decide entrar a la aplicación para realizar alguna operación.</td></tr>
<tr><td>Precondiciones:</td><td colspan="2"></td></tr>
<tr><td>Referencias</td><td colspan="2">R1</td></tr>
<tr><td>Prioridad</td><td colspan="2">Critico</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td>Acción del actor</td><td colspan="2">Respuesta del sistema</td></tr>
<tr><td>1 Introduce los datos:
Usuario
Contraseña
y selecciona la opción Aceptar, en caso contrario ir al Flujo alterno 1.</td><td colspan="2">1.1 El sistema verifica que los datos introducidos sean correctos, en caso contrario ir al Flujo alterno 2.</td></tr>
<tr><td></td><td colspan="2">1.2 El sistema muestra la interfaz Componente de Firma Digital.
Finaliza el caso de uso</td></tr>
<tr><td colspan="3">Prototipo de interfaz</td></tr>
<tr><td colspan="3"></td></tr>
</table>

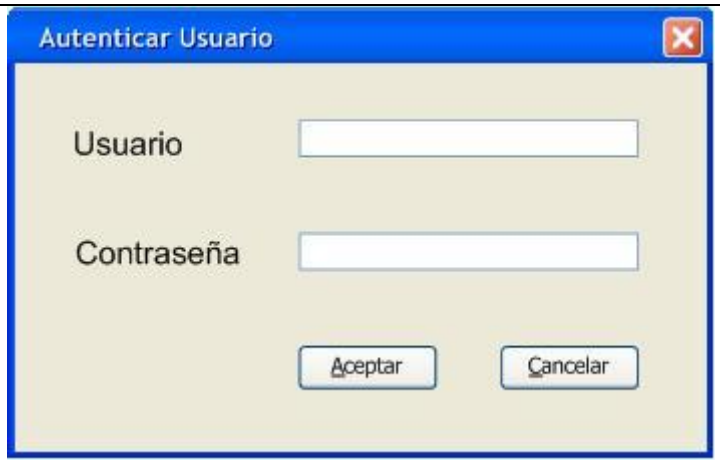
	
Flujos Alternos	
Flujo Alterno 1 "Cancelar"	
Acción del Actor	**Respuesta del Sistema**
1 Selecciona opción Cancelar.	1.1 El sistema cierra la interfaz correspondiente.
Flujo Alterno 2 "Verificar datos"	
	1.2 El sistema en caso de que los datos introducidos sean incorrectos muestra un mensaje con el siguiente texto: **"Verifique los datos".**
Poscondiciones	

Tabla 2.9 *Descripción detallada del CU Autenticar usuario*

Caso de uso:	Generar claves
Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el usuario accede por primera vez a la aplicación y desea generar su par de claves.
Precondiciones:	El usuario debe estar autenticado.
Referencias	**R2, R3**
Prioridad	Critico
Flujo Normal de Eventos	

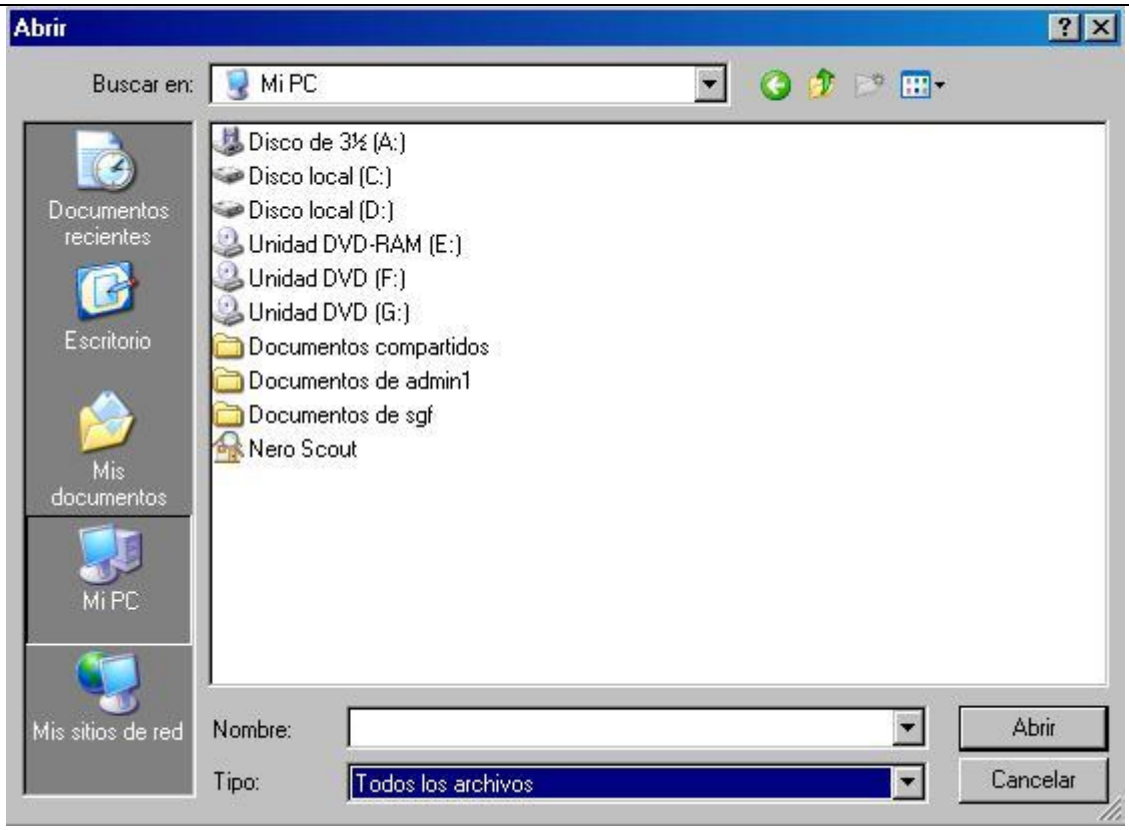
Acción del actor	Respuesta del sistema
1 El usuario selecciona de la interfaz principal la opción Generar claves.	1.1 El sistema muestra la interfaz **Generar claves**.
2 El usuario selecciona la longitud que tendrá la clave.	
3 El usuario selecciona la opción Explorar.	3.1 El sistema muestra la interfaz **Exportar.**
4 El usuario elige el directorio donde se guardará la clave y selecciona la opción Aceptar, en caso contrario ir al **Flujo alterno 2**.	
5 El usuario introduce la contraseña con la cual se encriptará la clave privada y luego la repite dicha contraseña.	
6 El usuario selecciona la opción Generar claves, en caso contrario ir al **Flujo alterno 1**.	6.1 El sistema verifica que las contraseñas sean iguales, en caso contrario muestra el mensaje con el siguiente texto "**Verifique las contraseñas**".
	6.2 El sistema verifica que el usuario haya seleccionado un directorio donde se guardarán la clave privada. En caso de que no se haya seleccionado se muestra un mensaje con el siguiente texto "**Debe seleccionar un directorio donde se guardará la clave privada**".
	6.3 El sistema genera las claves.
	6.4 El sistema guarda la clave privada en el directorio seleccionado y la pública en la base de datos.
	6.5 El sistema muestra un mensaje con el siguiente texto: "**Sus claves han sido generadas correctamente**".

7 El usuario selecciona la opción Exportar clave, en caso contrario ir al **Flujo alterno 1**.	7.1 El sistema invoca al caso de uso extendido Exportar clave. • Finaliza el caso de uso.
Prototipo de interfaz	
	
Flujos Alternos	
Flujo Alterno 1 "Cancelar"	
Acción del Actor	**Respuesta del Sistema**
1 Selecciona opción Cancelar.	1.1- El sistema cierra la interfaz correspondiente.
Flujo alterno 2 "Cancelar-cargar"	
Acción del Actor	**Respuesta del Sistema**
1 Selecciona opción Cancelar.	1.1 El sistema cierra la interfaz correspondiente. 1.2 Ir al paso 3 del **Flujo Normal de Eventos**
Poscondiciones	

Tabla 2.10 *Descripción detallada del CU Generar claves*

Caso de uso:	Exportar claves
Actores:	Usuario

<table>
<tr><td>Resumen:</td><td colspan="2">El caso de uso se inicia cuando el usuario desea guardar su clave pública en un archivo externo.</td></tr>
<tr><td>Precondiciones:</td><td colspan="2">El usuario debe estar autenticado y se deben haber generado las claves.</td></tr>
<tr><td>Referencias</td><td colspan="2">R4</td></tr>
<tr><td>Prioridad</td><td colspan="2">Secundario</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td colspan="2">Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td colspan="2">1 El usuario selecciona la opción Exportar claves.</td><td>1.1 El sistema muestra la interfaz Exportar clave.</td></tr>
<tr><td colspan="2">2 El usuario selecciona la opción Exportar, en caso contrario ir al Flujo alterno 1.</td><td>2.1 El sistema muestra la interfaz Exportar.</td></tr>
<tr><td colspan="2">3 El usuario selecciona el destino donde será guardada la clave.</td><td></td></tr>
<tr><td colspan="2">4 El usuario selecciona la opción Guardar, en caso contrario ir al Flujo alterno 2.</td><td>4.1 El sistema guarda las claves en el destino seleccionado por el usuario.
• Finaliza el caso de uso.</td></tr>
<tr><td colspan="3">Prototipo de interfaz</td></tr>
<tr><td colspan="3"> 
</td></tr>
</table>



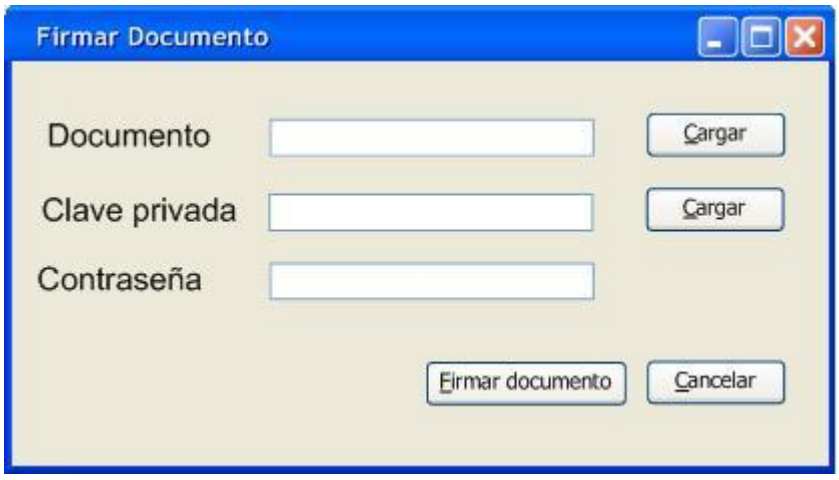
Flujos Alternos	
Flujo alterno 1 "Cancelar"	
Acción del Actor	**Respuesta del Sistema**
1- Selecciona opción Cancelar.	1.1- El sistema cierra la interfaz correspondiente.
Flujo alterno 2 "Cancelar-guardar"	
Acción del Actor	**Respuesta del Sistema**
1- Selecciona opción Cancelar.	1.1- El sistema cierra la interfaz correspondiente. 1.2- Ir al paso 2 del **Flujo Normal de Eventos**
Poscondiciones	

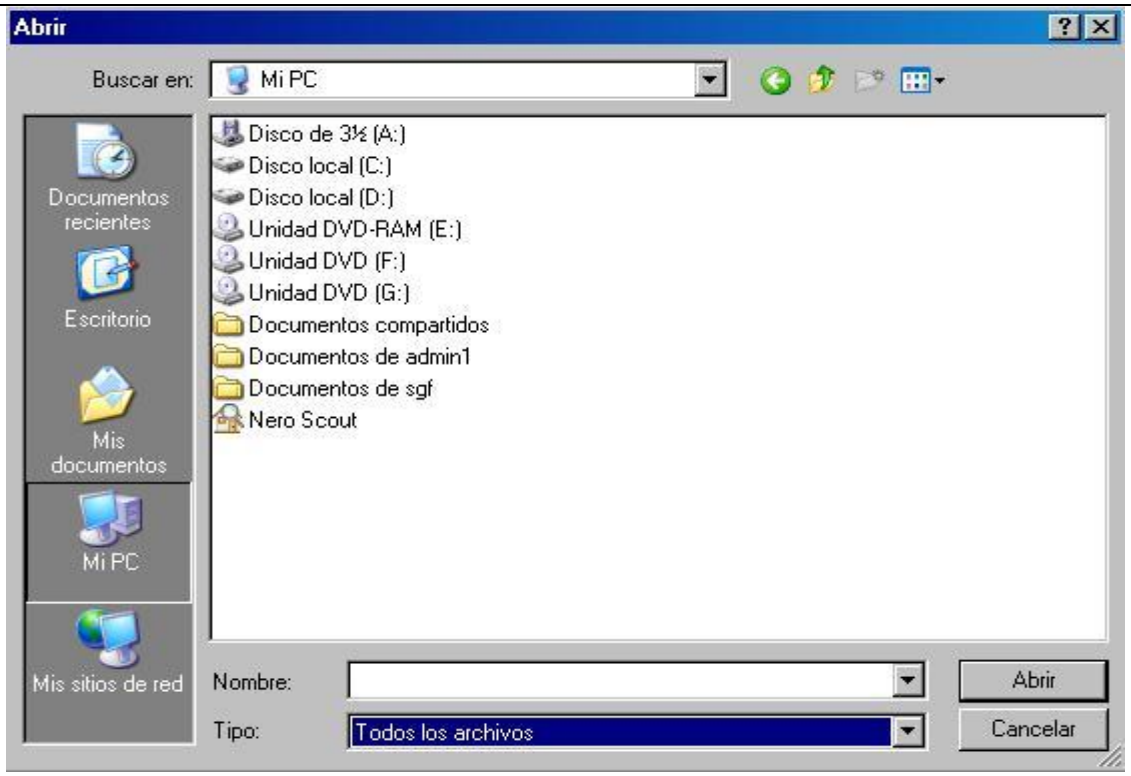
Tabla 2.11 *Descripción detallada del CU Exportar claves*

Caso de uso:	Firmar documento

Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el usuario desea firmar un documento.
Precondiciones:	El usuario debe estar autenticado y se deben haber generado las claves.
Referencias	**R5, R6, R7**
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1 El usuario selecciona la opción Firmar documento.	1.1 El sistema muestra la interfaz **Firmar documento**.
2 El usuario selecciona la opción Cargar documento, en caso contrario ir al **Flujo alterno 1**.	2.1 El sistema muestra la interfaz **Cargar documento**.
3 El usuario selecciona el documento a firmar.	
4 El usuario selecciona la opción Aceptar, en caso contrario ir al **Flujo alterno 2**.	4.1 El sistema carga el documento seleccionado y cierra la interfaz correspondiente.
5 El usuario selecciona la opción Cargar clave privada.	5.1 El sistema muestra la interfaz **Cargar documento**.
6 El usuario selecciona el documento donde se encuentra la clave privada.	
7 El usuario selecciona la opción Aceptar, en caso contrario ir al **Flujo alterno 2**.	7.1 El sistema carga el documento donde se encuentra la clave privada.

8 El usuario selecciona la opción Firmar documento, en caso contrario ir al **Flujo alterno 1**.	8.1 El sistema verifica que la contraseña introducida por al usuario es la correcta, en caso contrario muestra un mensaje con el siguiente texto "**Contraseña inválida**". Firma el documento y lo guarda en el mismo directorio donde se encuentra el documento original.
	8.2 El sistema firma el documento, guarda la firma en el mismo directorio y luego muestra un mensaje con el siguiente texto "**Su documento ha sido firmado**". • Finaliza el caso de uso.
Prototipo de interfaz	
	



Flujos Alternos	
Flujo alterno 1 "Cancelar"	
Acción del Actor	**Respuesta del Sistema**
1 Selecciona opción Cancelar.	1.1 El sistema cierra la interfaz correspondiente.
Flujo alterno 2 "Cancelar-cargar"	
Acción del Actor	**Respuesta del Sistema**
1 Selecciona opción Cancelar.	1.1 El sistema cierra la interfaz correspondiente. 1.2 Ir al paso 2 del **Flujo Normal de Eventos**
Poscondiciones	

Tabla 2.12 *Descripción detallada del CU Firmar documento*

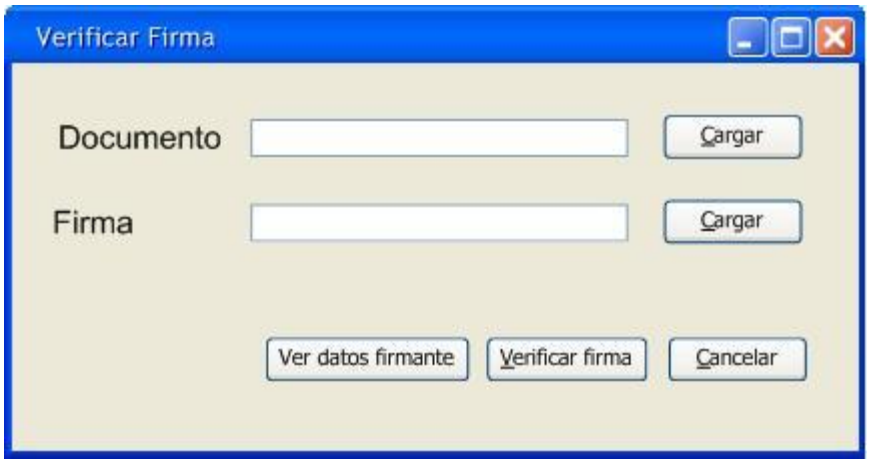
Caso de uso:	Verificar firma

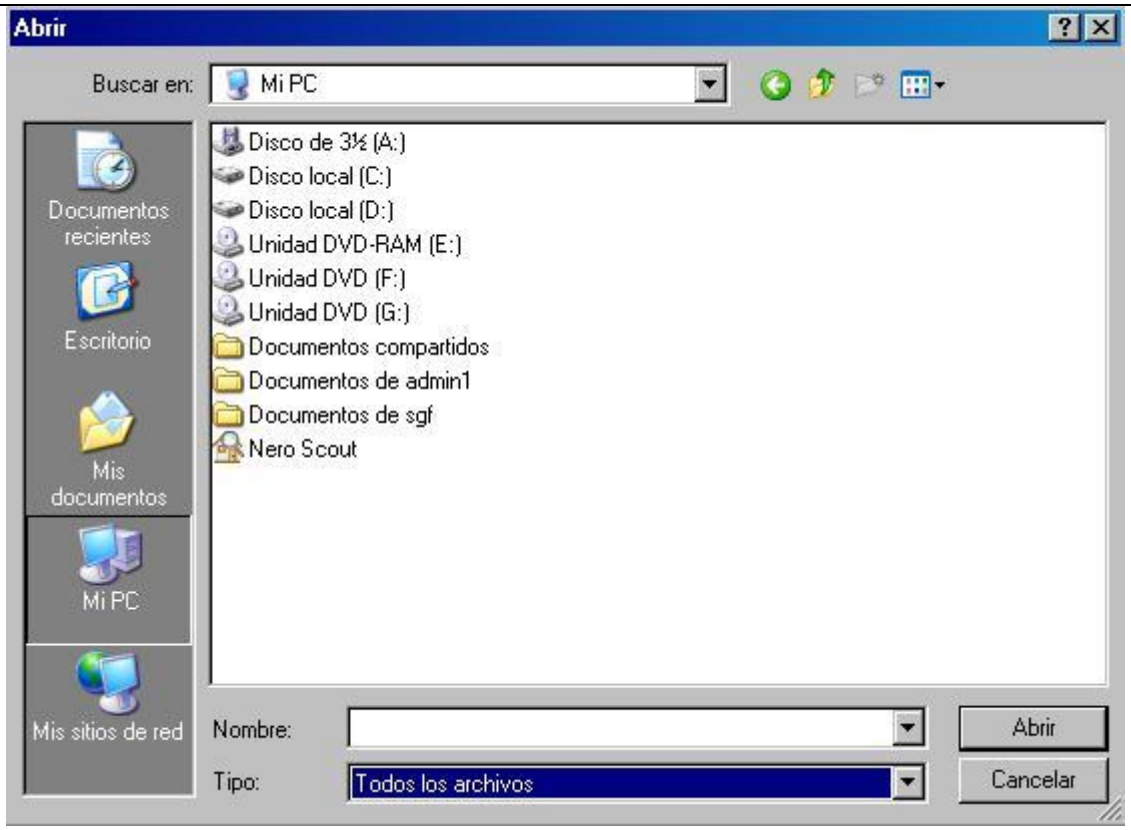
Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el usuario desea verificar la firma de un documento.
Precondiciones:	El usuario debe estar autenticado.
Referencias	**R5, R8, R9**
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1 El usuario selecciona la opción Verificar firma.	1.1 El sistema muestra la interfaz **Verificar firma**.
2. El usuario selecciona la opción Cargar documento, en caso contrario ir al **Flujo alterno 1**.	2.1 El sistema muestra la interfaz **Cargar documento**.
3 El usuario selecciona el documento a verificar.	
4 El usuario selecciona la opción Aceptar, en caso contrario ir al **Flujo alterno 2**.	4.1 El sistema carga el documento seleccionado y cierra la interfaz correspondiente.
5. El usuario selecciona la opción Cargar firma, en caso contrario ir al **Flujo alterno 1**.	5.1 El sistema muestra la interfaz **Cargar documento**.
6 El usuario selecciona la firma del documento.	
7 El usuario selecciona la opción Aceptar, en caso contrario ir al **Flujo alterno 2**.	7.1 El sistema carga el documento seleccionado y cierra la interfaz correspondiente.
8 Selecciona la opción Verificar firma, en caso contrario ir al **Flujo alterno 1**.	8.1 Verifica la firma y muestra el siguiente mensaje: **"La firma es válida"**, en caso contrario ir al **Flujo alterno 3**.

9 El usuario selecciona la opción Ver datos firmante, en caso contrario ir al **Flujo alterno 1**.	9.1 El sistema muestra una ventana con todos datos personales de la persona que firmó el documento. • Finaliza el caso de uso.

Prototipo de interfaz



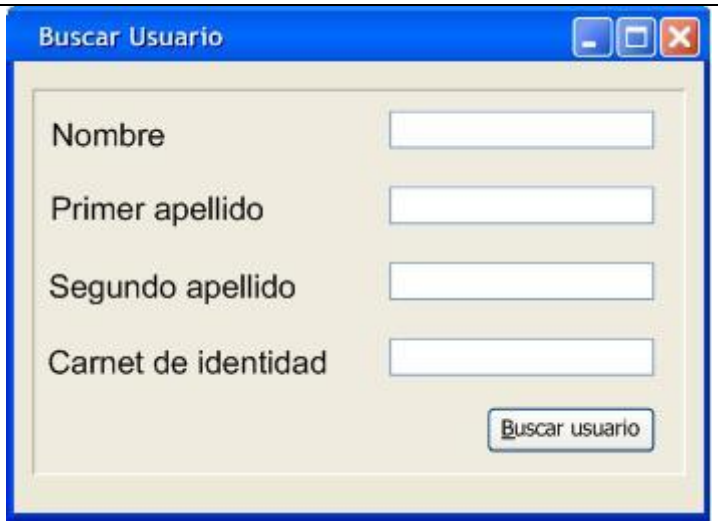


Flujos Alternos	
Flujo alterno 1 "Cancelar"	
Acción del Actor	**Respuesta del Sistema**
1 Selecciona opción Cancelar.	1.1 El sistema cierra la interfaz correspondiente.
Flujo alterno 2 "Cancelar-cargar"	
Acción del Actor	**Respuesta del Sistema**
1 Selecciona opción Cancelar.	1.1 El sistema cierra la interfaz correspondiente. 1.2 Ir al paso 2 del **Flujo Normal de Eventos**
Flujo alterno 3 "Verificar firma"	
Acción del Actor	**Respuesta del Sistema**
	1 Si la firma es incorrecta muestra el siguiente mensaje: **"La firma es inválida"**

Poscondiciones	

Tabla 2.13 *Descripción detallada del CU Verificar firma*

Caso de uso:	Buscar usuario
Actores:	Administrador
Resumen:	El caso de uso se inicia cuando el usuario desea buscar un usuario determinado.
Precondiciones:	El usuario debe estar autenticado.
Referencias	**R9**
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
	1 El sistema muestra la interfaz **Buscar usuario**.
2 Introduce el nombre, apellidos o carnet de identidad y selecciona la opción Buscar.	2.1 El sistema verifica que esté introducido al menos un parámetro de búsqueda, en caso de que no haya alguno ir a al **Flujo alterno 1**.
	2.2 El sistema muestra una lista con todos los usuarios que cumplan con el criterio de búsqueda introducido. • Finaliza el caso de uso.
Prototipo de interfaz	



Flujos Alternos	
Flujo alterno 1 "No existe parámetro por donde buscar"	
Acción del Actor	**Respuesta del Sistema**
	1 El sistema muestra un mensaje con el siguiente texto:**"Debe introducir al menos un criterio de búsqueda"**.
Poscondiciones	

Tabla 2.14 *Descripción detallada del CU Buscar usuario*

Caso de uso:	Cambiar claves

Actores:	Administrador
Resumen:	El caso de uso se inicia cuando el administrador desee cambiarle la clave a un usuario determinado.
Precondiciones:	El administrador debe estar autenticado.
Referencias	**R2, R9, R10**
Prioridad	Secundario

Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1 Accede a la opción Cambiar claves.	1. El sistema muestra la interfaz **Cambiar claves**.
2 Selecciona la opción Buscar usuario, en caso contrario ir al **Flujo alterno 1**.	2.1 Invoca al caso de uso incluido Buscar usuario.
3 Selecciona el usuario al cual le va a cambiar las claves.	
4 Selecciona el directorio donde se guardara la clave privada y deja que el usuario introduzca la contraseña por la cual se va a encriptar la clave privada.	
5 Selecciona la longitud de la clave.	
6 Elige la opción Cambiar claves, en caso contrario ir al **Flujo alterno 1**.	6.1 Verifica que el administrador haya seleccionado un usuario, en caso contrario ir al **Flujo alterno 2**.
	6.2 El sistema cambia las claves del usuario seleccionado y muestra el mensaje: **"Se han cambiado las claves"**. • Finaliza el caso de uso.
Prototipo de interfaz	


Cambiar Claves
Longitud
Elegir
Salvar en
Explorar
Contraseña
Repetir contraseña
Buscar Usuario
Nombre
Primer apellido
Segundo apellido
Carnet de identidad
Buscar usuario
Cambiar claves
Cancelar

Flujos Alternos	
Flujo alterno 1 "Cancelar"	
Acción del Actor	**Respuesta del Sistema**
1 Selecciona opción Cancelar.	1.1 El sistema cierra la interfaz correspondiente.
Flujo alterno 2 " No existe selección"	
Acción del Actor	**Respuesta del Sistema**
	1 El sistema muestra un mensaje con el siguiente texto:**"Debe seleccionar un usuario"**.
Poscondiciones	

Tabla 2.15 *Descripción detallada del CU Cambiar claves*

2.7 Conclusiones

Durante el desarrollo de este capítulo se obtuvieron algunos artefactos que serán empleados para la creación de un componente de firma digital para la FGRC. Seguidamente se muestran cada uno de ellos:

- Los requerimientos funcionales necesarios para implementar el sistema.
- El diagrama de casos de uso del sistema.
- La descripción de casos de uso del sistema.

CAPÍTULO 3: DISEÑO, IMPLEMENTACIÓN Y PRUEBA

3.1 Introducción

Este capítulo contiene el diseño e implementación del sistema, mostrando los diagramas desarrollados durante los flujos correspondientes de importancia en la solución propuesta. También se realiza un análisis de los resultados obtenidos mediante la aplicación de métricas y pruebas de unidad.

3.2 Modelo del diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en como los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Además, el modelo de diseño sirve de abstracción de la implementación del sistema y es, de ese modo, utilizada como una entrada fundamental de las actividades de implementación (Booch, y otros, 2005).

3.2.1 Patrones de diseño empleados

- **Singleton**: Este patrón es utilizado para crear una única instancia de una clase, permitiendo el acceso global de dicha instancia por medio de un método de la clase y para esto se declara el constructor de la clase privado para que no pueda ser instanciada mediante este.

```
public static function getInstance()
        {
                if (!self::$instancia instanceof self) {
                        self::$instancia = new self;
                }
                return self::$instancia;

                }
```

3.2.2 Estándar de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento

del código.

A continuación se presentan el estándar de codificación que se utilizará durante la programación del componente de firma digital.

Funciones y métodos

Los nombres de las funciones solo pueden tener caracteres alfanuméricos, no se permite utilizar subrayas. Los nombres de las funciones siempre deben comenzar con una letra minúscula. En caso de estar compuesta por más de una palabra, a partir de la segunda cada una debe comenzar con mayúscula. Los nombres deben ser tan detallados como sean posibles para describir su finalidad y comportamiento de forma óptima.

```
public function  autenticar()
public function  alert()
private function verificarFirma()
```

Variables

El nombre de las variables solo puede contener caracteres alfanuméricos, no se permite utilizar subrayas a menos que la variable sea declarada *private* o *protected*. Al igual que ocurre con el nombre de las funciones los nombres de las variables siempre comienzan con una letra minúscula cumpliendo con las mismas características. También debe ser un nombre lo más detallado posible, que pueda describir los datos que representan.

```
private $_host;
private $_tbUsuario;
$host;
```

Literales de cadena que contengan apóstrofes

Cuando una cadena literal contiene apóstrofes, se permite delimitar la cadena con comillas dobles (“”). Esto es especialmente útil para consultas SQL.

```
$consulta= " Select public.tbpersona.idpersona, public.tbpersona.usuario,
public.tbpersona.sha1_password,
public.tbpersona.salt FROM public.tbpersona where  public.tbpersona.usuario='$usuario'";
```

Declaración de clases

El nombre de la clase debe comenzar con una letra mayúscula, en caso de que un nombre compuesto las otras palabras también comienzan con. Sólo se permite una clase en cada archivo PHP. Todo el código en una clase deben ser indentados con cuatro espacios.

Se puede colocar código adicional en el archivo pero no se recomienda, en caso que esto ocurra debe estar separado de la clase por dos líneas en blanco.

```
class Ejemplo
{
    //contenido de la clase
    //indentado por 4 espacios
}
```

Declaraciones de control

Para las declaraciones de control basadas en *if* y *elseif* se debe poner un espacio antes del paréntesis de apertura de la condición y un espacio después de que el paréntesis se cierre. Dentro de las condiciones entre los paréntesis los operadores deben estar separados por espacios para mejorar la legibilidad.

La llave de apertura debe estar en la misma línea donde se encuentra la condición y la llave de cierre siempre está en su propia línea. El contenido dentro de las llaves debe estar indentado 4 espacios.

```
if ($a != 2) {
    $a = 2;
}
```

Para sentencias *if* que incluyan *elseif* o *else*, el formato será similar a la construcción del *if* mostrado.

```
if ($a != 2) {
    $a = 2;
} else {
   $a = 7;
}

if ($a != 2) {
    $a = 2;
} elseif ($a == 3) {
```

```
   $a = 4;
} else {
   $a = 7;
}
```

3.2.3 Diagrama de clases

Una clase de diseño y sus objetos, y de este modo también los subsistemas que contienen las clases de diseño, a menudo participan en varias realizaciones de casos de uso. También puede darse el caso de algunas operaciones, atributos y asociaciones sobre una clase específica que son relevantes para sólo una realización de caso de uso. Esto es importante para coordinar todos los requisitos que diferentes realizaciones de casos de uso imponen a una clase, a sus objetos y a los subsistemas que contiene. Para manejar todo esto, utilizamos diagramas de clases conectados a una realización de caso de uso, mostrando sus clases principales, subsistemas y sus relaciones (Booch, y otros, 2005).

A continuación se muestran los Diagramas de Clases del Diseño por casos de uso necesarios para llevar a cabo la implementación del sistema.

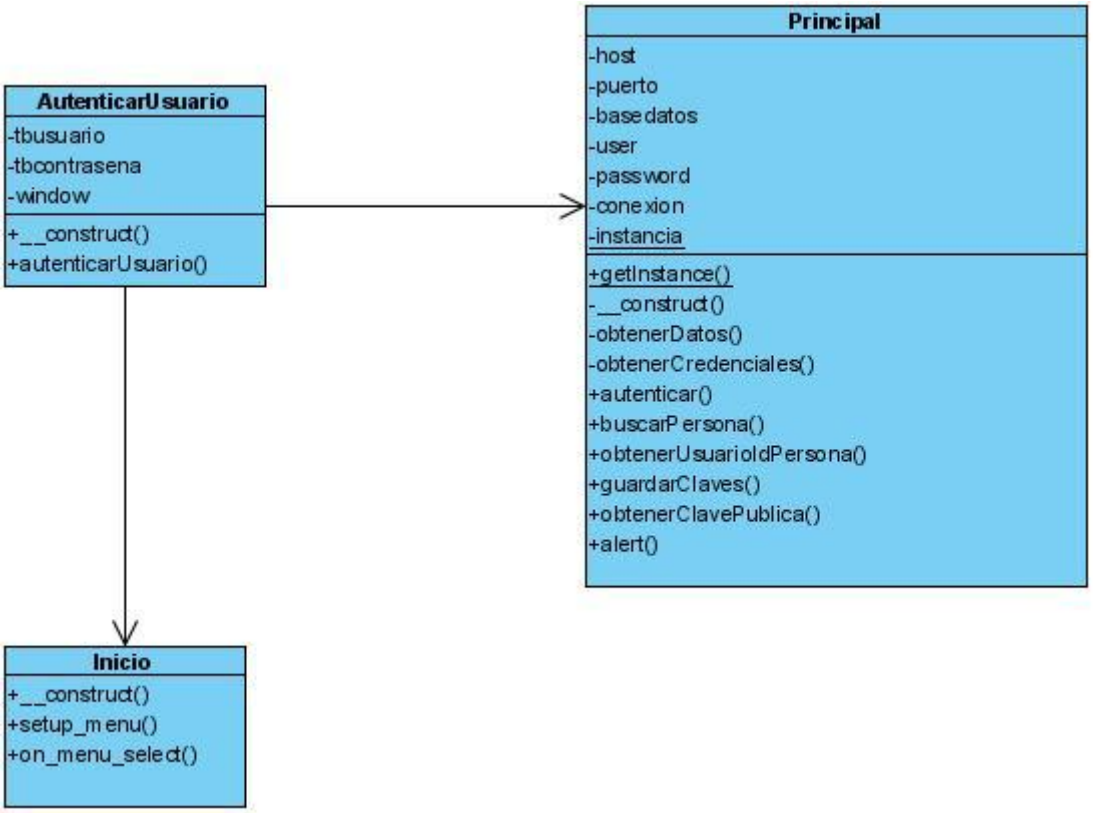


Figura 3.1: *Diagrama de clases del CU: Autenticar Usuario*

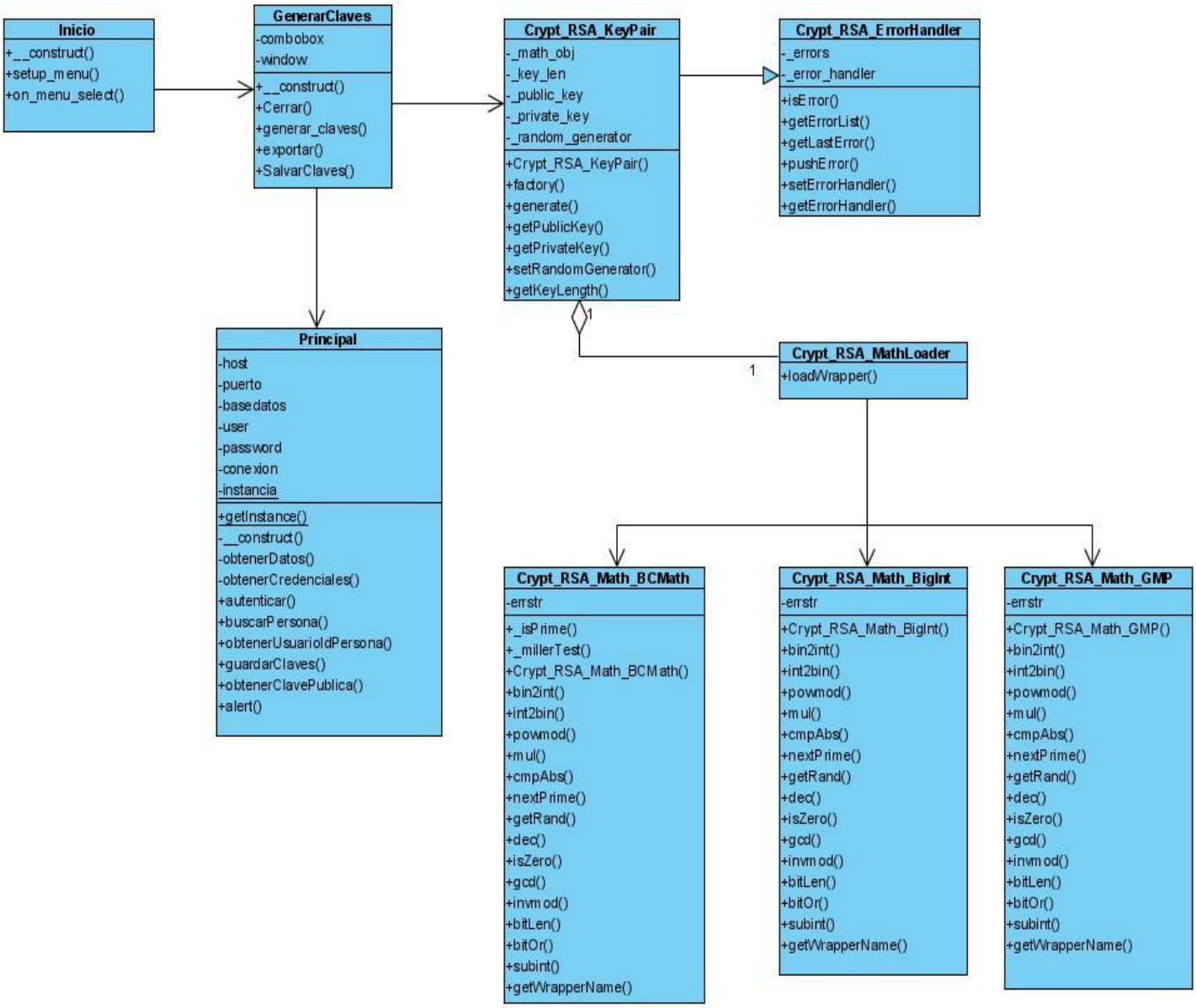


Figura 3.2: *Diagrama de clases del CU: Generar Claves*

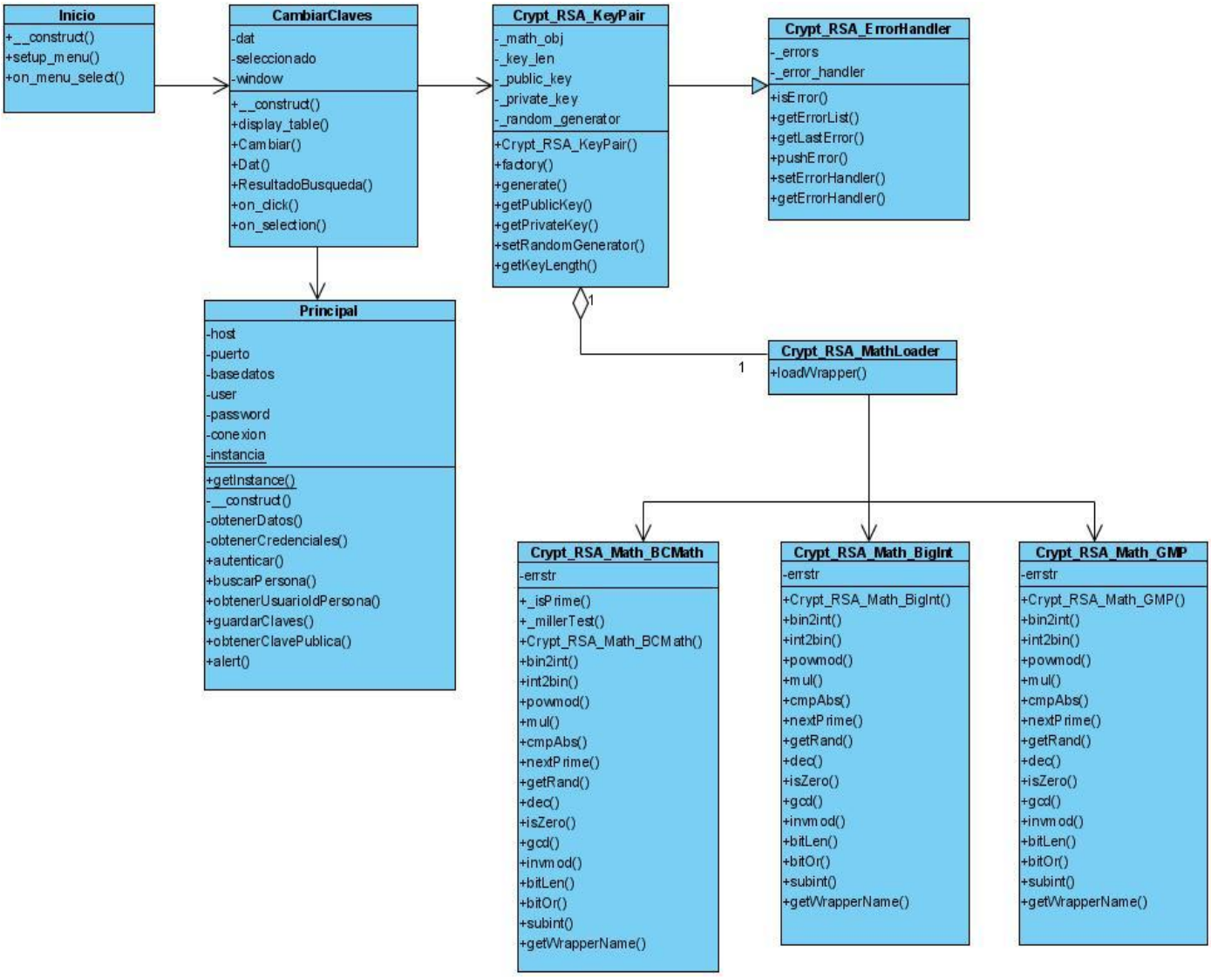


Figura 3.3: *Diagrama de clases del CU: Cambiar Claves*

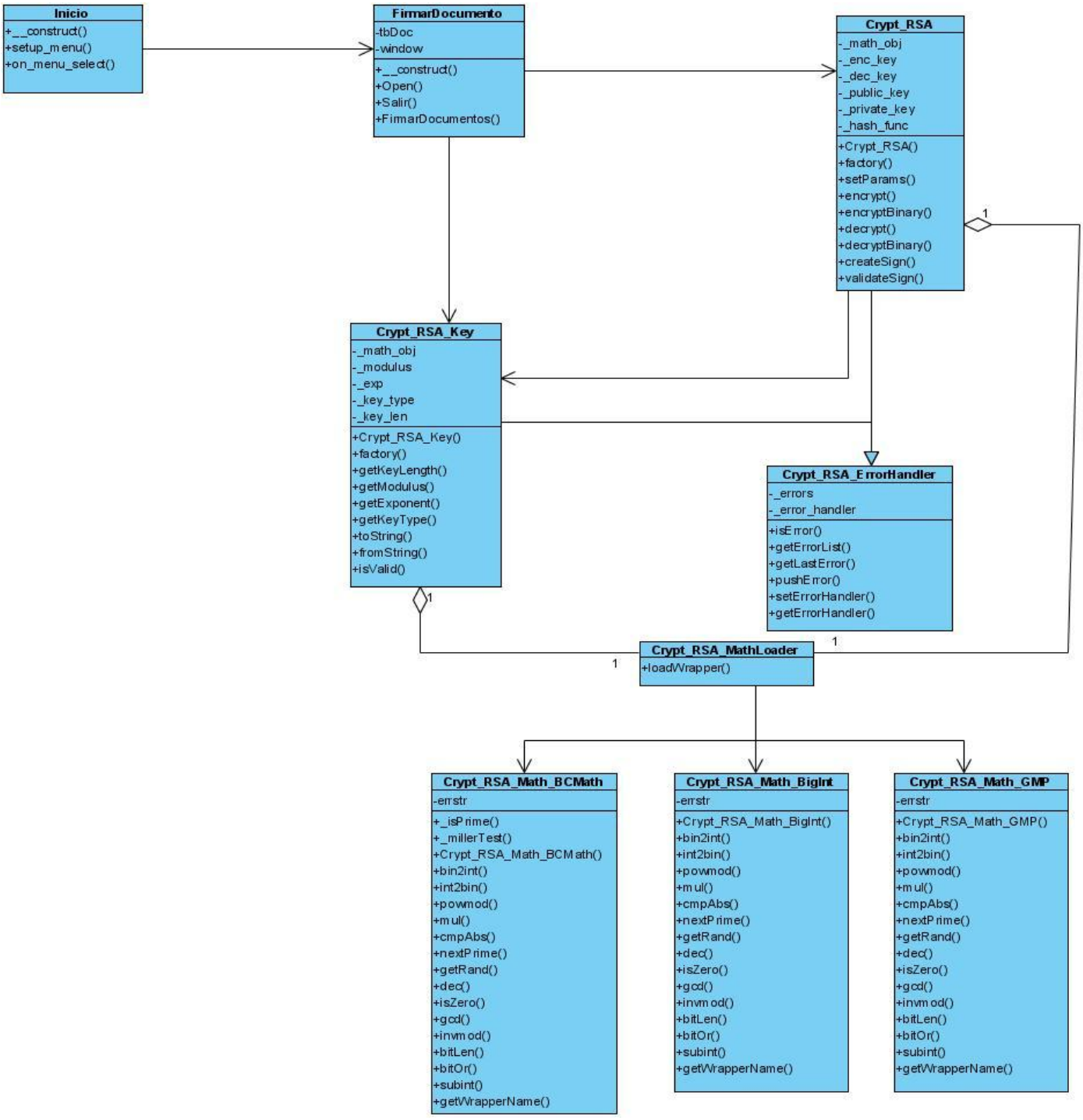


Figura 3.4: *Diagrama de clases del CU: Firmar Documento*

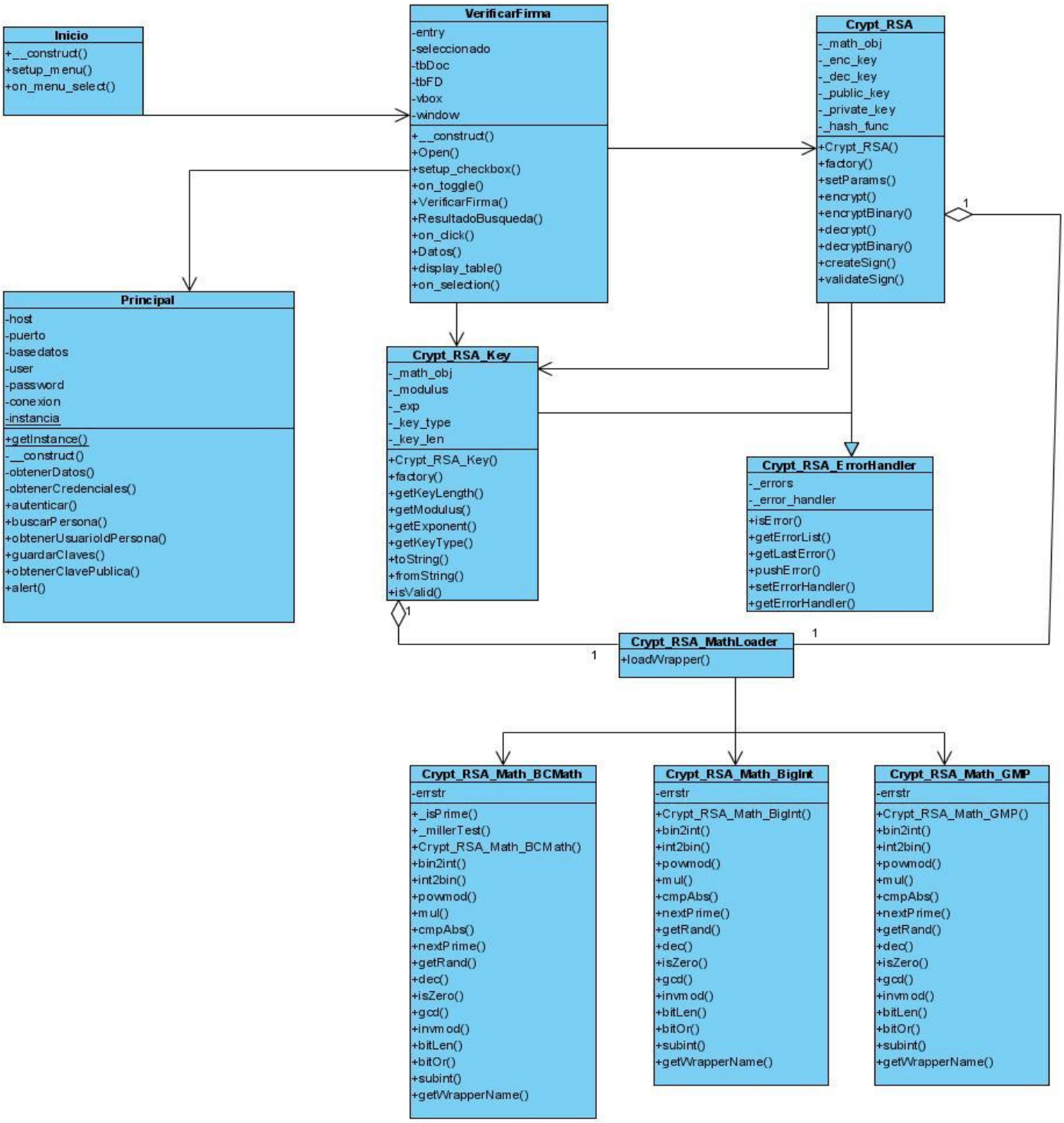


Figura 3.5: *Diagrama de clases del CU: Verificar Firma*

3.2.4 Diagramas de interacción.

Un diagrama de interacción es una representación de secuencias de intercambios y mensajes entre los roles que participan y se relacionan con un sistema, es decir, modelan los aspectos dinámicos de un sistema. Estos pueden ser de dos tipos: colaboración y secuencia.

En este epígrafe se hace referencia a los diagramas de secuencia que los cuales muestra las interacciones entre objetos ordenadas en secuencia temporal. También muestra los objetos que se encuentran en el escenario y la secuencia de mensajes intercambiados entre los objetos para llevar a cabo la funcionalidad descrita por el escenario. A continuación se muestras los diagramas de secuencia del diseño.

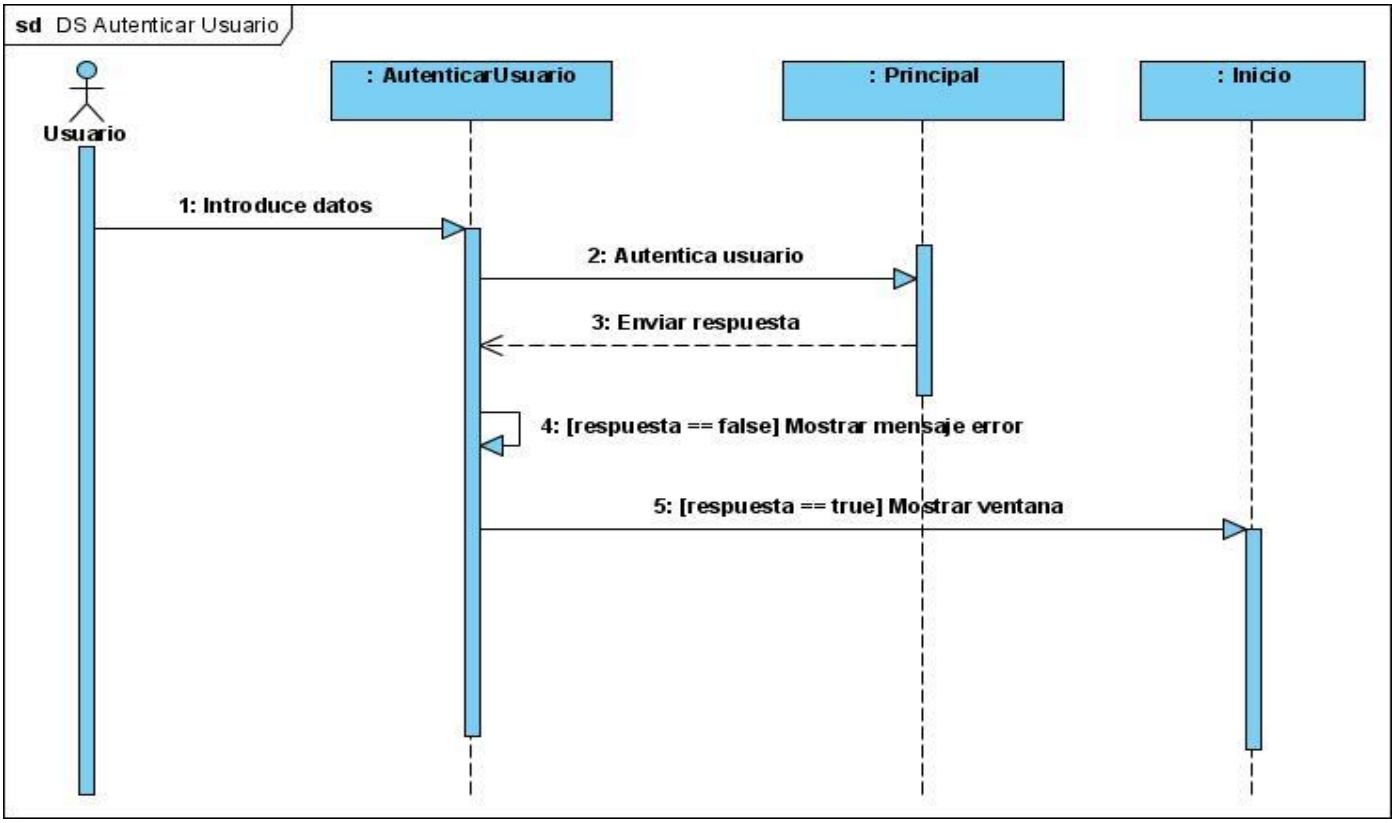


Figura 3.6: *Diagrama de secuencia del CU: Autenticar Usuario*

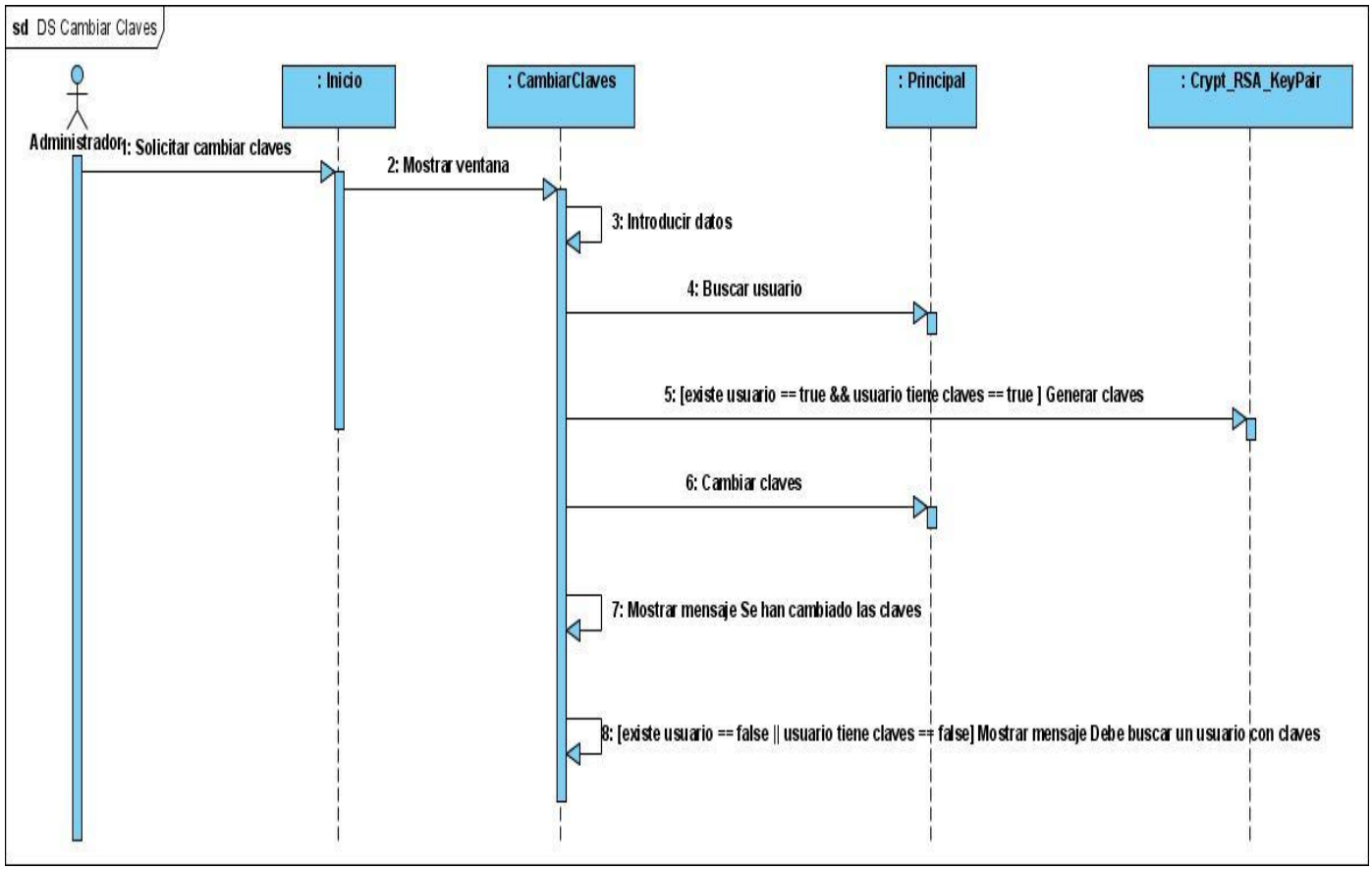


Figura 3.7: *Diagrama de secuencia del CU: Cambiar Claves*

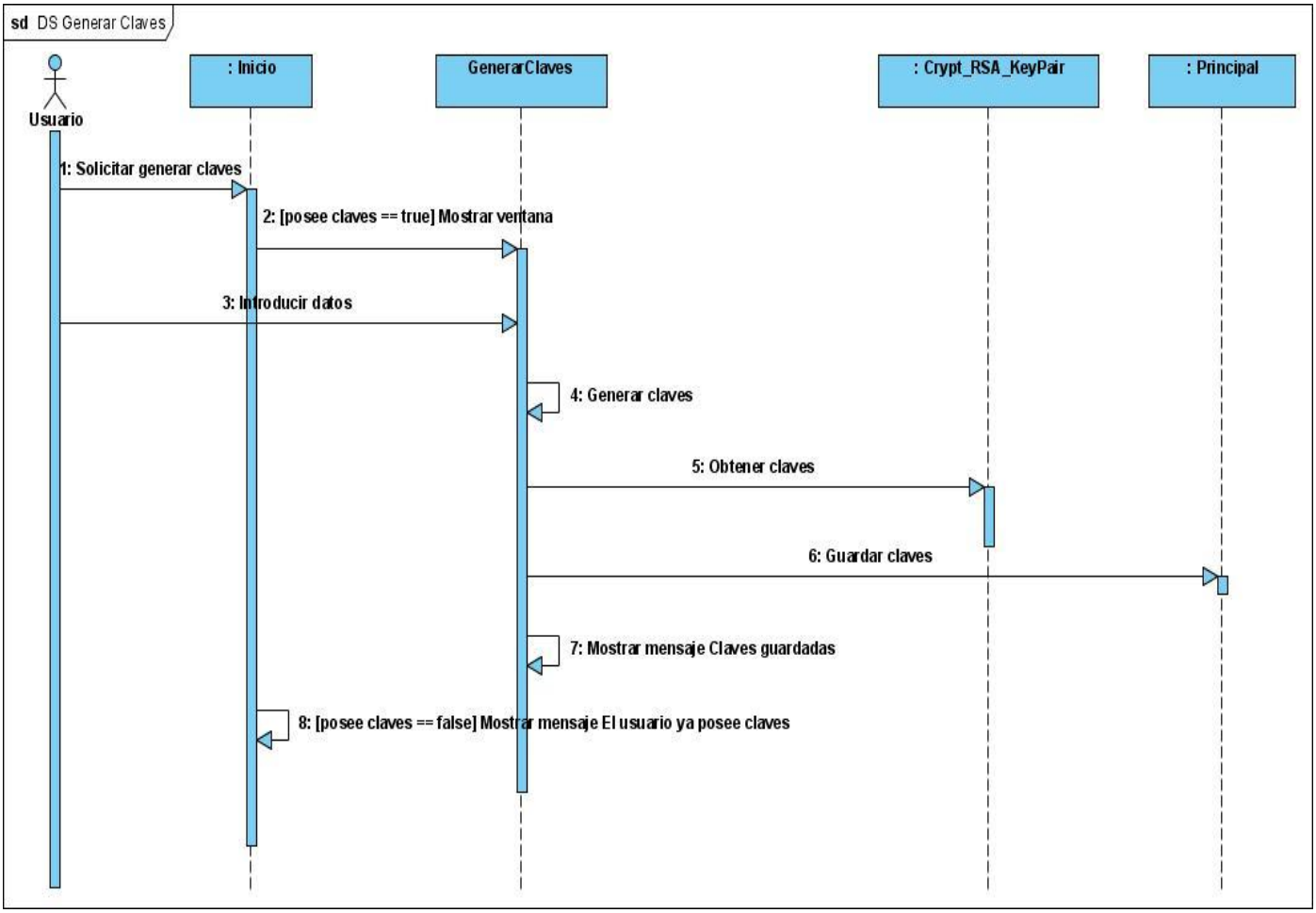


Figura 3.8: *Diagrama de secuencia del CU: Generar Claves*

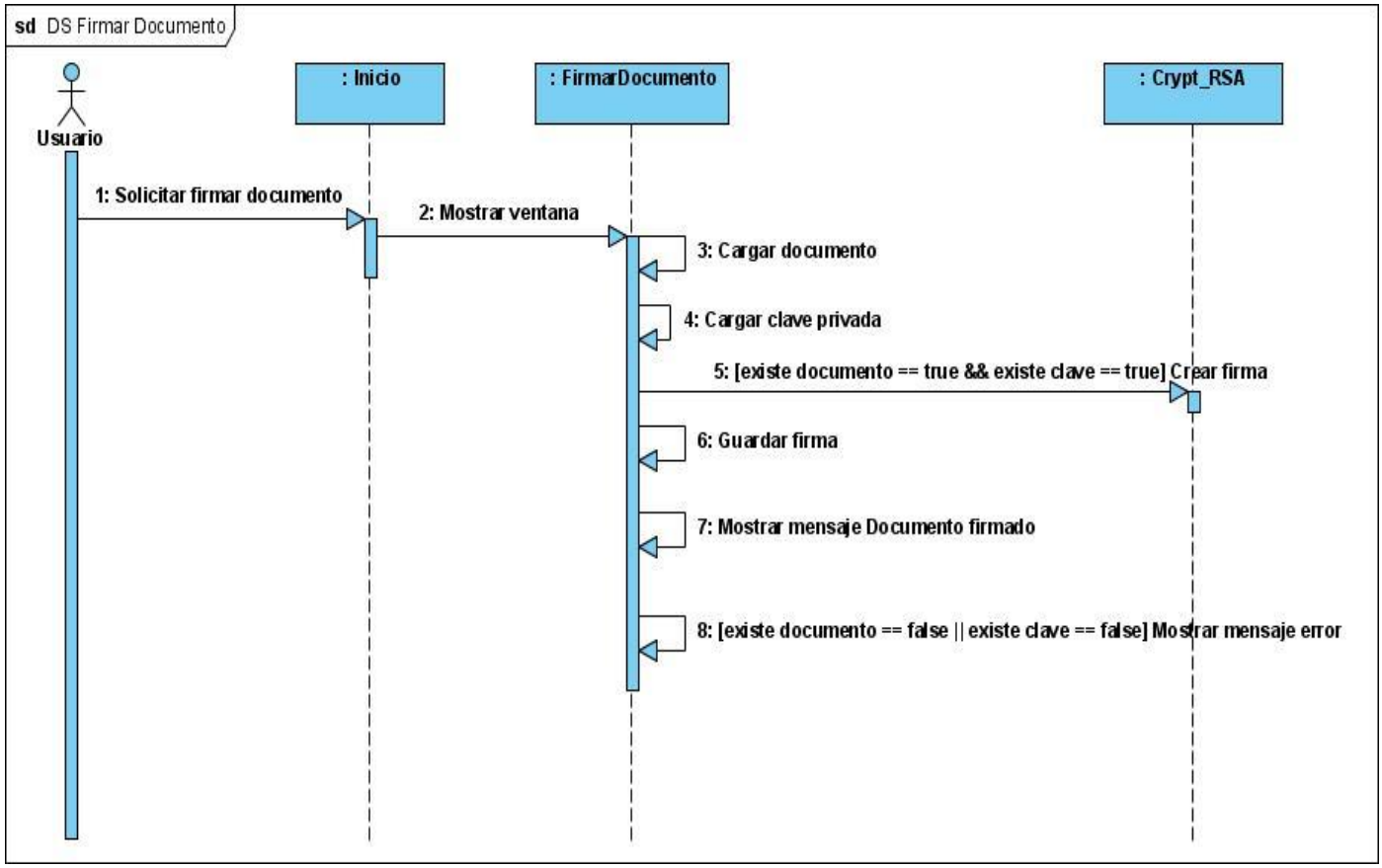


Figura 3.9: *Diagrama de secuencia del CU: Firmar Documento*

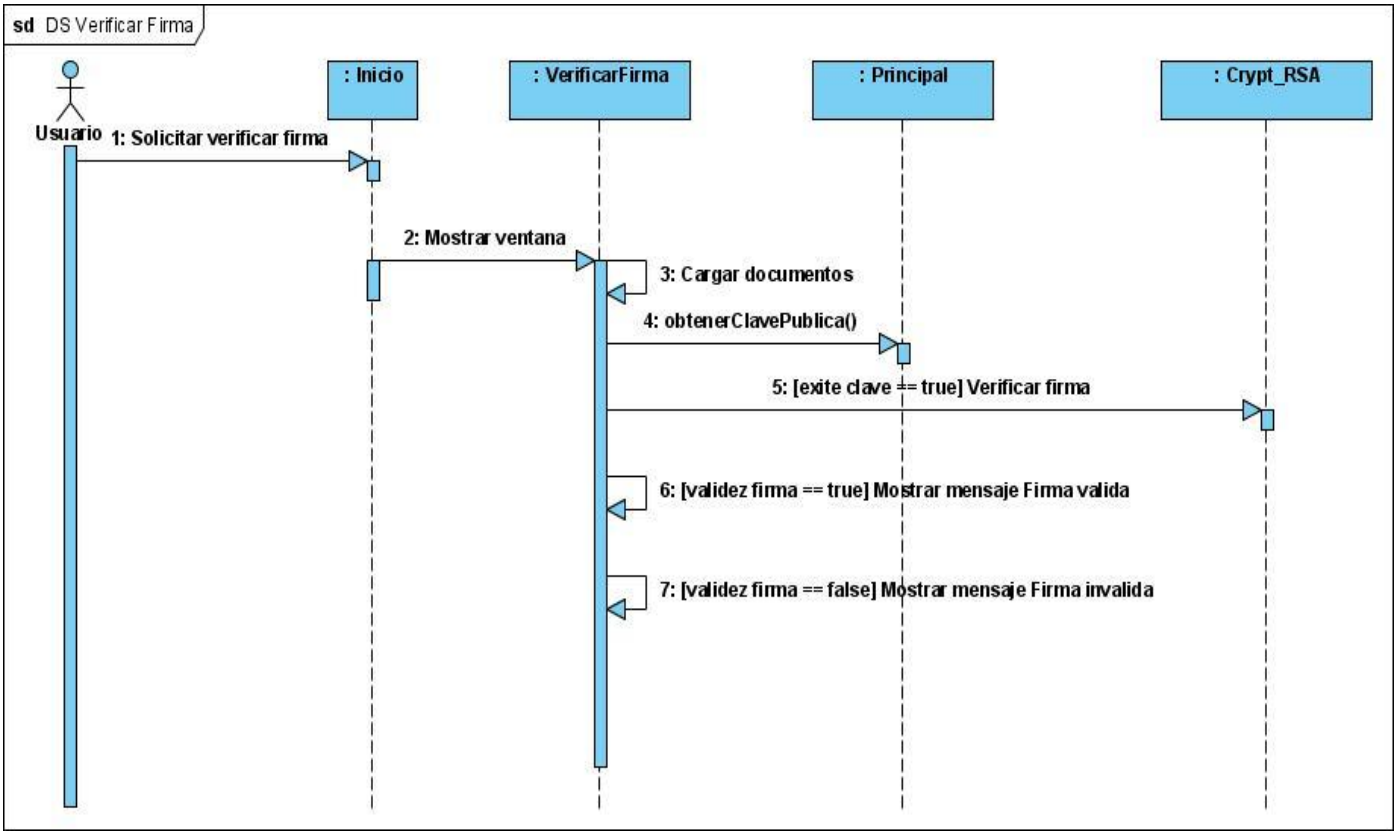


Figura 3.10: *Diagrama de secuencia del CU: Verificar Firma*

3.3 Modelo de implementación

El modelo de implementación describe como los elementos del modelo de diseño, tales como las clases, serán implementas en términos de componentes, ficheros de código fuente, ejecutables y otros. Además muestra como se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje de programación utilizado, y la dependencia que existen entre los componentes.

3.3.1 Diagrama de componentes

Este diagrama muestra las organizaciones y dependencias lógicas entre componentes de software, ya sean componentes de código fuente, binarios o ejecutables. Es empleado para mostrar una vista estática del sistema. A continuación se muestra el diagrama de componentes para el sistema.

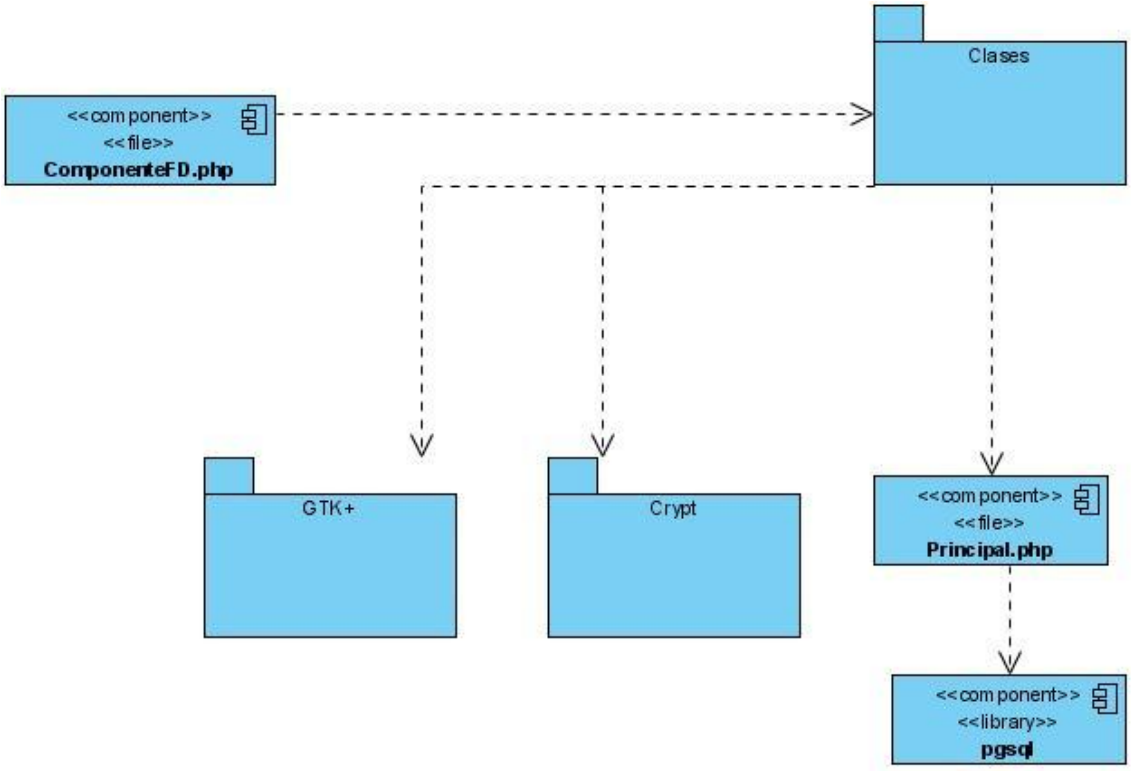


Figura 3.10: *Diagrama de componentes para el sistema*

3.3.2 Diagrama de despliegue

El diagrama de despliegue muestra las relaciones físicas entre los componentes de hardware y software en el sistema final. Es un grafo de nodos unidos por conexiones de comunicación. Un nodo puede reflejar componentes los cuales pueden estar conectados por relaciones de dependencia, posiblemente a interfaces. Teniendo en cuenta las características del sistema el Diagrama de Despliegue quedó como se muestra a continuación.

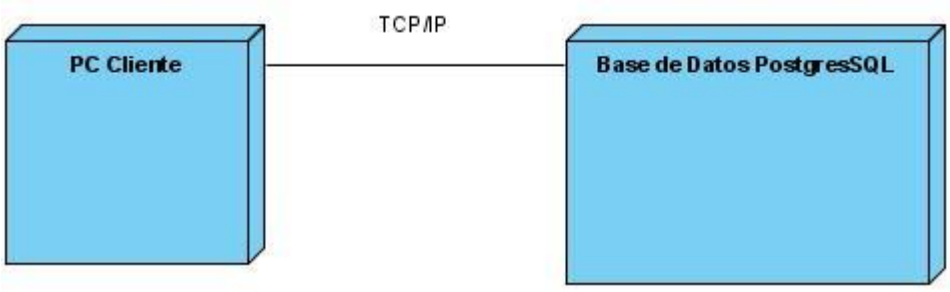


Figura 3.11: *Diagrama de despliegue del sistema*

3.4 Métricas aplicadas para el Modelo de Diseño Orientados a Objetos

Con el objetivo de entender mejor la calidad del producto, evaluar la efectividad del proceso y para mejorar la calidad del trajo llevada a cabo es que se emplean las métricas orientas a objetos. En este epígrafe se hace un análisis de los resultados obtenidos en el diseño del sistema y para esto se aplicarán un conjunto de métricas mostradas a continuación.

3.4.1 Métrica Tamaño de clase (TC)

Esta métrica fue propuesta por Lorenz y Kidd y se centra en el conteo de las operaciones y atributos para cada clase individual y los valores promedio para el sistema como un todo.

Para calcular el tamaño de una clase se debe determina el total de operaciones que se encapsulan dentro de esta, ya sean operaciones privadas o heredadas de la instancia, también se determina el número de atributo, tanto privados como heredados que contiene la instancia.

Si los valores para TC son altos, quiere decir que la clase debe tener bastante responsabilidad, lo que reduciría la reutilización de dicha clase y complicaría la implementación y las pruebas. Además de esto se puede calcular los promedios para el número de atributos y operaciones de clase y cuanto menor sea este promedio las clases podrán ser más reutilizables dentro del sistema.

En la actualidad no existe una medida fija para determinar el umbral para los parámetros de calidad en el diseño de sistemas, aunque la mayoría de los especialistas plantean umbrales como se muestra en la siguiente tabla.

Números de Operaciones y Atributos	
TC	**Umbral**
Pequeño	<=20
Medio	>20 y <=30
Grande	>30

Tabla 3.1 *Umbrales para la Métrica TC*

A continuación se muestra el resultado de la aplicación de esta métrica al sistema.

No.	Nombre de la clase	Cantidad de atributos	Cantidad de operaciones	Tamaño de la clase
1	AutenticarUsuario	3	2	Pequeño
2	CambiarClaves	3	7	Pequeño
3	FirmarDocumento	2	4	Pequeño
4	GenerarClaves	2	5	Pequeño
5	Inicio	0	3	Pequeño
6	VerificarFirma	6	10	Pequeño
7	Principal	7	10	Pequeño
8	Crypt_RSA_ErrorHandler	2	6	Pequeño
9	Crypt_RSA	8	15	Medio
10	Crypt_RSA_KeyPair	7	12	Pequeño
11	Crypt_RSA_Key	7	15	Medio
12	Crypt_RSA_MathLoader	0	1	Pequeño
13	Crypt_RSA_Math_BCMath	1	18	Pequeño
14	Crypt_RSA_Math_BigInt	1	16	Pequeño
15	Crypt_RSA_Math_GMP	1	16	Pequeño

Tabla 3.2 *Clases del sistema y su tamaño.*

En la tabla se muestra que 13 de las 15 clases tienen un tamaño Pequeño las otras 2 poseen un tamaño Medio lo que evidencia un buen resultado en la aplicación de la métrica seleccionada.

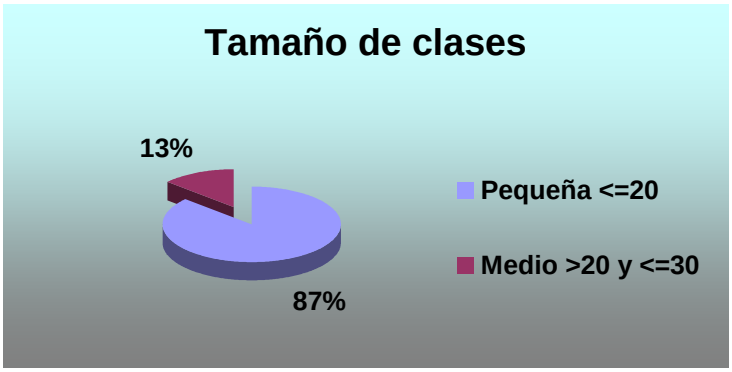


Figura 3.13 *Porcientos de clases por tamaños.*

3.5.2 Métrica Árbol de profundidad de herencia (APH)

Compuesta en el conjunto de métricas propuesta por Chidamber y Kemerer, comúnmente nombrada serie de métricas CK.

Esta métrica se define como la máxima longitud del nodo a la raíz del árbol. A medida que el valor del APH crece, es posible las clases de más bajos niveles hereden más métodos, trayendo consigo que puedan ocurrir dificultades cuando se intenta predecir el comportamiento de una clase, también conduce a una mayor complejidad en el diseño, aunque cuando un sistema presenta altos valores de APH es mayor el número de métodos que se reutilizarían.

Después de haberse aplicado esta métrica se obtuvo un valor de APH igual a 2, lo que nos expresa que la solución propuesta es de poca complejidad debido al bajo valor obtenido.

3.6 Resultados de las pruebas de unidad

Las pruebas de caja negra se aplican a la interfaz del software, centrándose en los requisitos funcionales del sistema. Estas permiten obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Entre sus objetivos se encuentran encontrar errores en funciones incorrectas o ausentes, en la interfaz de usuario, en estructuras de datos o en accesos a base de datos externas, en el rendimiento y errores de inicialización y terminación.

Al aplicarse los casos de pruebas (ver Casos de Pruebas en Anexos Tablas 1-14) al componente de firma digital se arrojaron resultados satisfactorios. Donde se demostró lo siguiente:

- El sistema responde a las funcionalidades descritas en la descripción de los CU.
- Se logra el objetivo de cada caso de uso en su flujo básico y están correctamente estructurados los pasos que corresponden al mismo así como a los del flujo alterno.

3.7 Conclusiones

En este capítulo se realizó el diseño de los diagramas de clases y diagramas de secuencia por cada uno de los procesos que forman parte del software que se presenta, y se obtuvo el diagrama de componentes y de despliegue. También se aplicaron métricas para la validación del diseño propuesto, en el cual se determinó que este no muestra una alta complejidad estructural, permitiendo que las pruebas no sean complejas y afecten en mínimo de tiempo algún cambio. Por último se diseñaron las pruebas para los casos de usos en las cuales los resultados fueron satisfactorios y se demostró que el producto responde a las necesidades y propósitos del cliente.

CONCLUSIONES

A partir del desarrollo del componente de firma digital se arriba a las siguientes conclusiones:

- Se realizó el modelo del dominio del sistema el cual permitió relacionar todos los conceptos del mismo.
- Se especificaron los requisitos funcionales y no funciones con los que debía cumplir el sistema.
- Se realizó el diseño correspondiente para el sistema obteniéndose un grupo de artefactos que serán utilizados durante la implementación de la solución.
- Se desarrolló y evaluó el componente de firma digital dándole cumplimiento al objetivo planteado.

RECOMENDACIONES

- Continuar implementando el sistema para integrarlo a una PKI que garantice la manipulación y emisión de certificados digitales.
- Mejorar la interfaz gráfica de usuario.
- Incorporar al sistema nuevos algoritmos de clave pública desarrollados en Cuba por el MININT (Ministerio del Interior).
- Que la firma se adjunte al documento en vez de ir en un archivo aparte.
- Que se puedan firmar documentos pdf.

BIBLIOGRAFÍA

Albaladejo, Xavier. [En línea] [Citado el: 10 de enero de 2009.] http://www.proyectosagiles.org/beneficios-de-scrum.

Booch, G., Rumbaugh, J. y Jacobson, I. 2000. *El Lenguaje Unificado de Modelado.* 2000.

—. 2005. *El Proceso Unificado de desarrollo de Software.* 2005.

DI Management. [En línea] [Citado el: 25 de noviembre de 2008.] http://www.di-mgt.com.au/rsa_alg.html.

ETSI. [En línea] [Citado el: 4 de febrero de 2009.] http://www.etsi.org.

Extreme Programming. [En línea] [Citado el: 17 de febrero de 2009.] http://www.extremeprogramming.org/.

González Duque, Raúl. *Python para todos.*

H. Canós, José, Letelier, Patricio y Penadés, Mª Carmen. 2003. *Metodologías Ágiles en el Desarrollo de Software.* 2003.

Kriptópolis. [En línea] [Citado el: 27 de enero de 2009.] http://www.kriptopolis.com.

Martínez Equihua, Saúl. 2007. *Biblioteca digital: Conceptos, recursos y estándares.* 2007.

Mattocks, Scott. 2006. *Pro PHP-GTK.* 2006.

Menezes, A. J. y Van Oorschot, P. C. y Vanstone, S. A. *Handbook of applied cryptography.*

Menezes, A. J., Van Oorschot, P. C. y Vanstone, S. A. 2004. *Guide to elliptic curve cryptography.* 2004.

Miller, V. 1985. *Use of elliptic curves in cryptography.* 1985.

Pango. Pango. [En línea] [Citado el: 10 de febrero de 2009.] http://www.pango.org.

RSA Laboratories. [En línea] [Citado el: 20 de diciembre de 2008.] http://www.rsa.com/rsalabs/node.asp?id=2146.

SPARXSYSTEMS. [En línea] [Citado el: 21 de enero de 2009.] http://www.sparxsystems.com.ar.

The PHP Group. PHP. [En línea] [Citado el: 20 de enero de 2009.] http://www.php.net/manual/es/.

ANEXOS

1 GLOSARIO