Universidad de las Ciencias Informáticas

Facultad 1

Middleware para el applet de PKI “uSafeApp”.

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

Autores: Amanda Cabrera Miralles
Fernando José Enriquez Fernández

Tutores: Msc. Adonis Cesar Legón Campo
Ing. Dayron Almeida Sotolongo

La Habana, Julio 2013

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores del presente trabajo titulado: Aplicación para tarjetas inteligentes en una Infraestructura de Clave Pública y autorizamos a la Universidad de las Ciencias Informáticas y al Centro de Identificación y Seguridad Digital a usarlo en su beneficio.

Para que así conste firmamos la presente a los ____días del mes _____ del año _____.

_____ _____

Amanda Cabrera Miralles Fernando José Enriquez Fernández

_____

Adonis César Legón Campo

AGRADECIMIENTO

Generales: agradecemos a todas las personas que han colaborado con nosotros de alguna u otra manera, en especial a los integrantes del departamento de Tarjetas Inteligentes, tanto profesores como estudiantes. A nuestros tutores, tribunal y oponente, por la ayuda brindada. A la universidad y los profesores que nos formaron. A la Revolución.

Fernando:

Amanda:

A mi mamá y mi papá, por aceptar mis decisiones, apoyarme en todo este tiempo, estar presentes a lo largo de toda mi vida y por hacerme sentir orgullosa de ser su hija.

A mis abuelos, por todo el cariño que siempre me han brindado y su preocupación todos estos años. A mi abuelito Tico, porque hoy estaría muy orgulloso de mí.

A mis hermanos Ander y Adier, que son mi alegría y para quien siempre quiero ser un ejemplo.

A mi familia completa, porque siempre han estado pendientes de mí.

A Kire por compartir mi vida, por estar siempre para mí, por ser especial.

A Ayle por tanto apoyo.

A Abel por su amistad incondicional y todos sus regaños para que siempre sea mejor.

A mis amigos Abel, Charlie, Ayle, Marlon, Pumier, Ledier, Eddy, Dari, Anita, Yoel, Raylith, Audrey, Yeni, Damaris, David, Marlene, Matilde, Yoenia, Ali.

A los profesores, mis amistades que son muchas, a mis amigos y compañeros de aula que me acompañaron en este largo viaje que ha sido la UCI.

A todas las personas que de una forma u otra brindaron su apoyo para la realización de esta investigación.

Por último pero no menos importante a mi tutor y sobre todo a mi compañero de tesis, Fernan gracias por tanta paciencia.

DEDICATORIA

Fernando:

Amanda: A mi familia, porque a ellos les debo el ser humano que soy y son quienes me inspiran y guían en todo momento. En especial a mi mamá y mi papá que son un ejemplo cada día para mí y mis hermanos que son mi luz; a mi Kire por estar siempre, para ellos es este trabajo con todo el amor del mundo.

RESUMEN

En la actualidad es de suma importancia el uso de las tarjetas inteligentes en operaciones de firma digital de documentos y la autenticación con servicios en línea, un ejemplo de ellos son los bancos, debido a la seguridad que ofrece este tipo de dispositivo. El Centro de Identificación y Seguridad Digital (CISED) de la Universidad de las Ciencias Informáticas (UCI), a partir de la experiencia adquirida en el área de las tarjetas inteligentes, ha identificado la necesidad del desarrollo de una aplicación para tarjetas inteligentes en una Infraestructura de Clave Pública que cumpla con los principales estándares asociados a las mismas, su desarrollo permitirá poder hacer uso del *applet* "uSafeApp" para operaciones de firma digital de documentos, operaciones criptográficas y gestión de certificados digitales.

Para dicha aplicación se hace necesario el desarrollo de un middleware que cumpla con el estándar PKCS#11, lo cual permitirá la comunicación con aplicaciones cliente como: aplicaciones de correo, Mozilla Firefox, Acrobat Reader y el *applet* "uSafeApp".

El documento recoge los resultados de la investigación realizada, describiéndose las principales características de los sistemas analizados, la arquitectura y el diseño de la solución propuesta. Además se describen las herramientas, tecnologías utilizadas y los artefactos generados en el proceso de desarrollo.

Palabras claves: firma digital, certificados digitales, infraestructura de clave pública, tarjeta inteligente.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La sociedad ha evolucionado, convirtiéndose la información en un aspecto fundamental de las actividades sociales, culturales y económicas de la misma; se ha convertido en “Sociedad de la Información” y existe un sentimiento generalizado de que cualquier sistema informático es seguro y nadie puede atacar; ni acceder a entornos confidenciales restringidos, es por ello que uno de los aspectos más importantes es la seguridad de la información. La realidad es que la seguridad completa no existe, aunque sistemas PKI (*Public Key Infrastructure* o Infraestructura de clave pública) basándose en el uso de certificados digitales con criptografía de clave pública mejoran este aspecto. Este es el sistema más extendido que permite la centralización de las operaciones relacionadas con la firma electrónica.

Las tarjetas inteligentes proporcionan seguridad y exactitud en la verificación de identidad. Cuando estas se combinan con otros sistemas tecnológicos de identificación tales como: identificación biométrica y certificados digitales, se incrementa la seguridad de los sistemas y se protege la privacidad de la información. Desde su creación, las tarjetas inteligentes ha evolucionado en desarrollo de su tecnología, lo que ha llevado a un aumento en las aplicaciones que se les puedan instalar [1].

Las tarjetas inteligentes se utilizan en una amplia gama de industrias en todo el mundo para soportar aplicaciones de acceso, la identidad, de pago y otros. Día tras día se pueden encontrar nuevos tipos de aplicaciones que hacen uso de ellas. Las aplicaciones fundamentales de las tarjetas inteligentes son: identificación del titular, pago electrónico de bienes o servicios mediante dinero virtual y almacenamiento seguro de información. Además en varios países del mundo se usan como documento oficial de identificación, entre los que están España y Finlandia.

Las tarjetas inteligentes son utilizadas hoy en día para disímiles funciones y una de ellas es incrementar la seguridad de una Infraestructura de Clave Pública. El grado de seguridad de una PKI se incrementa considerablemente con el empleo de las tarjetas inteligentes, puesto que permite la creación y almacenamiento de llaves y certificados digitales en un entorno seguro dentro de la tarjeta. Las tarjetas inteligentes, que forman parte de una Infraestructura de Clave Pública, son ideales para los clientes que requieren los más altos estándares de seguridad como son: las autoridades públicas y las corporaciones que implementan servicios con acceso online, operadores de red e instituciones financieras que cuentan con sistemas de *e-commerce* (Comercio Electrónico) y permiten transacciones seguras utilizando firma digital. Es por ello que

las tarjetas inteligentes son un vínculo vital en la cadena de confianza de la Infraestructura de Clave Pública.

En Cuba la experiencia en cuanto al desarrollo de aplicaciones para tarjetas inteligentes es muy escasa. En el campo de aplicaciones para Infraestructura de Clave Pública no se conocen productos, ni desarrollos nacionales, a excepción de la Universidad de las Ciencias Informáticas. La UCI cuenta con varios centros de desarrollo de software, uno de ellos es el Centro de Identificación y Seguridad Digital (CISED), compuesto por el departamento de Tarjetas Inteligentes y otros. Dicho departamento se dedica a desarrollar aplicaciones, con el fin de obtener productos y servicios que utilicen esta tecnología, como aplicaciones relacionadas con los documentos de viaje electrónicos (norma ICAO), Licencia de conducción (ISO-18013), verificación biométrica y una aplicación de PKI llamada "uSafeApp", que implementa el estándar PKCS#15[1] (*Cryptographic Token Information Format Standard)* de RSA *Laboratories*, cada una de estas aplicaciones posee su *middleware* que cumple con los estándares que aplican a cada caso, pero en el caso de la aplicación PKI, existe un middleware que por sus características de implementación no cumple con el estándar correspondiente para que clientes de correo, navegadores web y herramientas de firma digital puedan utilizar una tarjeta con el *applet* "uSafeApp" para operaciones de firma digital y autenticación.

Partiendo de lo anteriormente expuesto surge como **situación problemática**:

De las soluciones desarrolladas por el departamento de Tarjetas Inteligentes, el *applet* "uSafeApp" no posee un *middleware* que cumpla con los estándares correspondientes para que clientes de correo, navegadores web y herramientas de firma digital puedan usar una tarjeta con el *applet* "uSafeApp", para operaciones de firma digital y autenticación en sitios web, es por ello que se hace necesario para el CISED desarrollar un middleware para el *applet* de PKI (Infraestructura de Clave Pública) "uSafeApp", que cumpla con los principales estándares internacionales que aplican a su vez, el uso de estos dispositivos en operaciones de firma digital y autenticación utilizando certificados digitales.

Derivado de la situación anteriormente expuesta se define el siguiente **problema de la investigación:** ¿Cómo usar el *applet* "uSafeApp" para realizar operaciones de firma digital y autenticación, cumpliendo con los estándares internacionales correspondientes para este tipo dispositivo en una PKI?

[1] Define un estándar que permite a los usuarios de dispositivo criptográficos identificarse con aplicaciones.

Se tiene como **objeto de estudio** de la presente investigación: Las tarjetas inteligentes como dispositivo de seguridad.

Para dar solución al problema existente se ha definido como **objetivo genera**l: Desarrollar un *middleware* que implemente las operaciones necesarias para la comunicación con el *applet* de PKI “uSafeApp”, de manera que se puedan ejecutar operaciones criptográficas que cumplan con el estándar PKCS#11, permitiendo el uso de tarjetas inteligentes en una Infraestructura de Clave Pública.

El **campo de acción** se centra en los middlewar*e* para tarjetas inteligentes en una Infraestructura de Clave Pública.

Tareas de Investigación.

- Análisis de las soluciones existentes sobre el uso de tarjetas inteligentes en una Infraestructura de Clave Pública.
- Análisis del estándar PC/SC relacionado con la comunicación con los lectores de las tarjetas.
- Análisis de los principales aspectos de la norma ISO/IEC 7816 relacionados con la comunicación con tarjetas inteligentes, la estructura para almacenamiento de la información y los modelos de seguridad.
- Análisis del estándar PKCS#15 relacionado con la estructura de la información para un dispositivo criptográfico.
- Análisis del estándar PKCS#11 relacionado con las funcionalidades que brinda un dispositivo criptográfico.
- Análisis de los principales algoritmos de criptografía simétricos y asimétricos, protocolos de autenticación, algoritmos de firma digital y certificados digitales.
- Análisis de elementos del applet de PKI “uSafeApp” tales como: estructura de los objetos de datos y ficheros, máquina de estado principal y comandos que soporta.
- Descripción de las herramientas, tecnologías y metodologías de desarrollo seleccionadas para el desarrollo del middleware.
- Análisis y diseño del middleware, usando patrones de diseño de software.
- Implementación de una aplicación cliente de prueba que permita probar las funcionalidades implementadas en el middleware desarrollado.
- Realización de pruebas de unidad al middleware desarrollado.
- Realización de las pruebas de calidad a la aplicación desarrollada.

Como **estrategia de investigación**; la exploratoria puesto que es necesario hacer una investigación profunda del objeto de estudio en cuestión, mientras el conocimiento no sea el suficiente para alcanzar el objetivo propuesto en el trabajo.

Los siguientes **métodos científicos** sustentarán la investigación:

Métodos teóricos

Analítico–Sintético: Con el uso de este método se podrá realizar un estudio de diferentes soluciones middleware para tarjetas inteligentes en una Infraestructura de Clave Pública existentes en la actualidad y los principales estándares relacionados con las tarjetas inteligentes, además se realizará un estudio del applet "uSafeApp" que permita conocer su funcionamiento y esto propiciará que se pueda crear un middleware que pueda manejar la información de manera segura dentro de la tarjeta. Además se consultará toda la bibliografía necesaria para dar cumplimiento a las tareas de la investigación.

Histórico-Lógico: Permitirá el análisis histórico del proceso de realizar operaciones criptográficas y de gestión de certificados digitales, en tarjetas inteligentes. Este método se emplea durante todo el proceso de diseño y desarrollo de funcionalidades del middleware para el *applet* "uSafeApp".

Método empírico

Observación: Este método es uno de los más utilizados en la presente investigación científica, precisamente por ser un procedimiento sencillo de realizar y que permite percibir directamente, los hechos de la realidad objetiva en el ámbito de la Infraestructura de Clave Pública utilizando tarjetas inteligentes.

Justificación de la Investigación

El fundamento de la presente investigación está dado por la importancia que posee el desarrollo de este tipo de soluciones, ya que con la utilización de una tarjeta inteligente como elemento de seguridad en una Infraestructura de clave pública, aumentaría la seguridad de ésta, al permitir que la llave privada utilizada en las operaciones de firma digital se genere dentro de la tarjeta, por lo que nunca estaría expuesta a un medio inseguro. La investigación permitirá al CISED, el desarrollo de productos y servicios que empleen las tarjetas inteligentes como dispositivo criptográfico, para aumentar la seguridad de una PKI, cumpliendo así, con una de las líneas de investigación del departamento de tarjetas inteligentes.

El trabajo de diploma está estructurado en los siguientes capítulos:

Capítulo 1 Fundamentación Teórica: este capítulo contiene una base teórica para concebir el problema planteado. Se describen los conceptos fundamentales para el

dominio del problema, se muestra el estudio de la aplicación applet de PKI “uSafeApp”, además de hacer referencia a estándares, tecnologías, herramientas y metodologías actuales que se usaron en el presente trabajo de diploma. Se presentan las fases de Exploración y Planificación definidas por la metodología XP para dar solución al problema científico

Capítulo 2 Análisis y Diseño: se identifican las historias de usuarios y los requerimientos no funcionales, se realiza el plan de iteraciones y plan de entregas. Se realiza el diseño de la solución a través de los diagramas de clases y de arquitectura apoyándose en los patrones de diseño y arquitectónicos.

Capítulo 3 Implementación y Pruebas: en este capítulo se realizan los diagramas de componentes y despliegue y se describen las pruebas realizadas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se perfilarán los conceptos fundamentales que servirán de base para el desarrollo del trabajo. Se realizará un análisis de las tecnologías más avanzadas para realizar el proceso de desarrollo del middleware en el marco de las tarjetas inteligentes. Además se caracterizarán los diferentes estándares, herramientas y metodologías con las cuales se desarrollará el diseño de la solución. Se aborda también sobre las diferentes aplicaciones middleware existentes en la actualidad relacionadas con las tarjetas inteligentes integradas a una Infraestructura de Clave Pública, que permitan operaciones de firma digital y autenticación, utilizando certificados digitales y se analiza el estado del arte referente a las herramientas y metodologías con las cuales se desarrollará la aplicación.

1.2 Conceptos fundamentales asociados al dominio del problema.

1.2.1 Autenticación

El propósito de la autenticación es verificar la identidad y genuinidad entre las partes de una comunicación. La autenticación requiere que las partes compartan un mismo secreto y que este se pueda verificar a partir de un método de autenticación como:

- Sistemas basados en algo conocido. Ejemplo, un *password* (Unix) o *passphrase* (PGP).
- Sistemas basados en algo poseído. Ejemplo, una tarjeta de identidad, una tarjeta inteligente, dispositivo usb, dongle[2] criptográfico.
- Sistemas basados en una característica física del usuario o un acto involuntario del mismo: Ejemplo, verificación de voz, de escritura, de huellas, de patrones oculares [2].

1.2.2 Criptografía

Arte de escribir en clave secreta o de un modo enigmático. Es la técnica para asegurar la transmisión de información privada que utiliza una escritura convencional secreta, de manera que sea ilegible para cualquiera que no posea la clave de descifrado [3].

Existen dos tipos fundamentales de criptosistemas:

- **Criptosistemas simétricos o de clave secreta:**

Emplean la misma clave k tanto para cifrar como para descifrar. Presentan el inconveniente de que para ser empleados en comunicaciones, la clave k debe estar

[2] Es un pequeño dispositivo de hardware que se puede integrar a un programa y se conecta a un ordenador, normalmente, para autenticar un fragmento de software.

tanto en el emisor como en el receptor, lo cual lleva preguntar cómo transmitir la clave de forma segura [2].

- **Criptosistemas asimétricos o de llave pública:**

Emplean una doble clave (kp, kP). A kp se le conoce como clave privada y kP se conoce como clave pública. Una de ellas sirve para la transformación E de cifrado y la otra para la transformación D de descifrado. En muchos casos son intercambiables, esto es, si empleamos una para cifrar la otra sirve para descifrar y viceversa. Estos criptosistemas deben cumplir además que el conocimiento de la clave pública kP no permita calcular la clave privada kp. Ofrecen un espectro superior de posibilidades, pudiendo emplearse para establecer comunicaciones seguras por canales inseguros puesto que únicamente viaja por el canal la clave pública, que sólo sirve para cifrar o para llevar a cabo autentificaciones [2].

1.2.3 Firma Digital

Es un método que asocia la identidad de una persona o equipo, con un mensaje o documento electrónico, para asegurar la autoría y la integridad del mismo. La firma digital del documento es el resultado de aplicar algoritmos matemáticos, (denominados función hash), a su contenido y así generan una firma digital del documento.

El término firma digital usualmente sólo se utiliza relacionado con algoritmos criptográficos asimétricos debido a que la separación de las llaves pública y privada permite una mayor cantidad de aplicaciones para la firma digital. Sin embargo las firmas basadas en algoritmos simétricos son utilizadas a menudo en la práctica aunque estas solamente permiten comprobar la autenticidad de documentos si la llave utilizada para generar la firma es conocida [4].

Una firma digital presenta una analogía directa con la firma manuscrita, y para que sea equiparable a esta última debe cumplir las siguientes propiedades [4]:

- Va ligada indisolublemente al mensaje. Una firma digital válida para un documento no puede ser válida para otro distinto.
- Sólo puede ser generada por su legítimo titular. Al igual que cada persona tiene una forma diferente de escribir, y que la escritura de dos personas diferentes puede ser distinguida mediante análisis grafológicos, una firma digital sólo puede ser construida por la persona o personas a quienes legalmente corresponde.
- Es públicamente verificable. Cualquiera puede comprobar su autenticidad en cualquier momento, de forma sencilla.

1.2.4 Certificado digital

Es un Documento Digital que permite [5]:

- La identificación al usuario.
- Firmar digitalmente un documento digital.
- Trabajar con documentos digitales firmados digitalmente teniendo certeza respecto del remitente y el destinatario.
- Efectuar transacciones de tipo comercial con total seguridad y sustento legal.
- Mantener la confidencialidad de la información entre el remitente y el destinatario utilizando cifrado.
- Estar seguros de que un documento digital no ha sido alterado.

En síntesis, la utilización de certificados digitales garantiza Autentificación, Integridad, Confidencialidad, No Repudio [5].

Es la certificación electrónica que vincula unos datos de verificación de firma a un signatario y confirma su identidad. La PSC (Prestadores de Servicios de Certificación) certifican que quien firma un documento electrónico, utiliza realmente las claves de quien dice ser, para lo cual han generado previamente la clave pública y privada del firmante, y le han identificado [6].

1.2.5 Middleware

El middleware puede ser considerado como un software de conectividad que permite la interconexión entre diferentes aplicaciones, el acceso a las funcionalidades y a los datos de estas, desde y a través de otros sistemas independientemente de la plataforma. El middleware asociado al applet, que corre sobre el sistema operativo del chip embebido en la tarjeta inteligente funciona como una capa de software intermediario, que se intercala entre las aplicaciones del usuario y la tarjeta [7].

1.2.6 Tarjeta Inteligente

Las tarjetas inteligentes fueron concebidas y patentadas por alemanes, japoneses y franceses en la década del setenta. Estos dispositivos son tarjetas de plástico que contienen un circuito integrado, son semejantes en tamaño y otros estándares físicos a las tarjetas de crédito. Dicho circuito puede ser de solo memoria o contener un microprocesador con un sistema operativo que le permite un grupo de tareas tales como:

1. Almacenar información.
2. Cifrar información.
3. Leer y escribir datos, como una computadora.

Las tarjetas inteligentes con su seguridad hacen que los datos que se consideran personales solo sean accesibles por los usuarios apropiados. Las tarjetas inteligentes aseguran la portabilidad, seguridad y confiabilidad en los datos [8].

Es importante destacar el nivel de seguridad que poseen las tarjetas inteligentes para el almacenamiento de la información, siendo a su vez multi-aplicación, o sea puede contener información de una o varias aplicaciones al mismo tiempo, esto puede ocurrir debido a la característica que poseen, ya que tienen una jerarquía de almacenamiento de la información. Cada una de las aplicaciones contenidas en la tarjeta se protege independientemente [8].

Con el surgimiento de las tarjetas inteligentes se han desarrollado cuantiosas aplicaciones, que hasta hace unos años eran consideradas prácticamente imposibles. En la actualidad se pueden apreciar en diferentes ámbitos, siendo los más comunes:

- Telefonía Móvil Digital (GSM): donde la identificación del titular del número telefónico, la herramienta de cifrado y la base de datos del propietario constituyen las funciones de la tarjeta inteligente.
- Banca: su desarrollo está dado a través de los monederos electrónicos, también para transacciones en el caso de las PKI permitiéndole al usuario cargarlo con una determinada cantidad de dinero y realizar compras teniendo en cuenta el saldo disponible en la tarjeta [4].

Las tarjetas inteligentes son clasificadas de acuerdo a sus capacidades, estructura del sistema operativo, formato e interfaz de comunicación. Esta última está compuesta por otras dos clasificaciones: tarjetas de contacto y tarjetas sin contacto. Las tarjetas de contacto disponen de unos contactos metálicos visibles y debidamente estandarizados [9], y las tarjetas sin contacto (contactless smart card) se comunican usando tecnología RFID[3]. Estas tarjetas solo requieren estar cerca de una antena para completar la transmisión de información (el estándar sugiere distancias de 10 cm para algunos usos y hasta 50 cm para otros).

1.2.7 Applet Java Card.

Los applets son las aplicaciones que corren dentro de una tarjeta Java Card. Dichas aplicaciones interactúan en todo momento con el Java Card Runtime Environment (JCRE) utilizando los servicios que éste brinda, e implementan la interfaz definida en la clase abstracta javacard.framework.Applet.[10, 11].

[3] El RFID o Identificación por radiofrecuencia es un sistema de almacenamiento y recuperación de datos remotos que usa dispositivos denominados etiquetas, transpondedores (dispositivo electrónico utilizado en las comunicaciones inalámbricas) o tags RFID.

1.3 Tecnología de Infraestructura de Clave Pública

Una infraestructura de clave pública (*Public Key Infrastructure*) es la combinación de productos de hardware y software, políticas y procedimientos para proveer un nivel adecuado de seguridad en transacciones electrónicas a través de redes públicas, como Internet.

La infraestructura de clave pública se basa en identificaciones digitales, también conocidas como certificados digitales, los cuales actúan como "pasaportes electrónicos" vinculando a un usuario de firma digital con su clave pública.

Debido a la característica impersonal involucrada en este tipo de tecnología es que se hace necesario contar con medios que garanticen una efectiva identificación y autenticación de los usuarios participantes con el fin de poder lograr el no repudio de las operaciones realizadas. Asimismo, en todo momento debe poder ser garantizada la confidencialidad e integridad de las transacciones que viajan por la red.

1.3.1 Componentes de una Infraestructura de Clave Pública

Una PKI tiene varios componentes como son: usuarios, entidades certificadoras (CA), certificados, y repositorios de certificados. Se encarga de proveer una forma de estructurar estos componentes y define estándares con este propósito. Una forma bastante común, típica o muy utilizada de PKI es la que se muestra en la figura 1 (tomado de [12]). En la imagen la CA raíz denominada *Root* certifica a las CAs del segundo nivel (RA en la figura 1) que pueden representar las CAs de algún país o una región y estas a su vez certifican a las CAs que generan los certificados X.509[4] de organizaciones e individuos. Cuando la autoridad raíz autoriza una RA, genera un certificado X.509 con la llave pública de la RA incluida y firmada por ella, que establece que ha aprobado esa RA. Similar al proceso anterior ocurre en el caso de las demás CAs. Para verificar la legitimidad de un certificado solo se debe seguir la cadena de confianza (verificar cada certificado con la llave pública almacenada en el certificado que emitió la CA superior), hasta encontrar una CA en la cual se confíe [13].

La siguiente imagen muestra una PKI estructurada jerárquicamente.

(a) Estructura de la PKI.

(b) Cadena de certificados.

[4] Estándar que se abunda en los siguientes puntos.

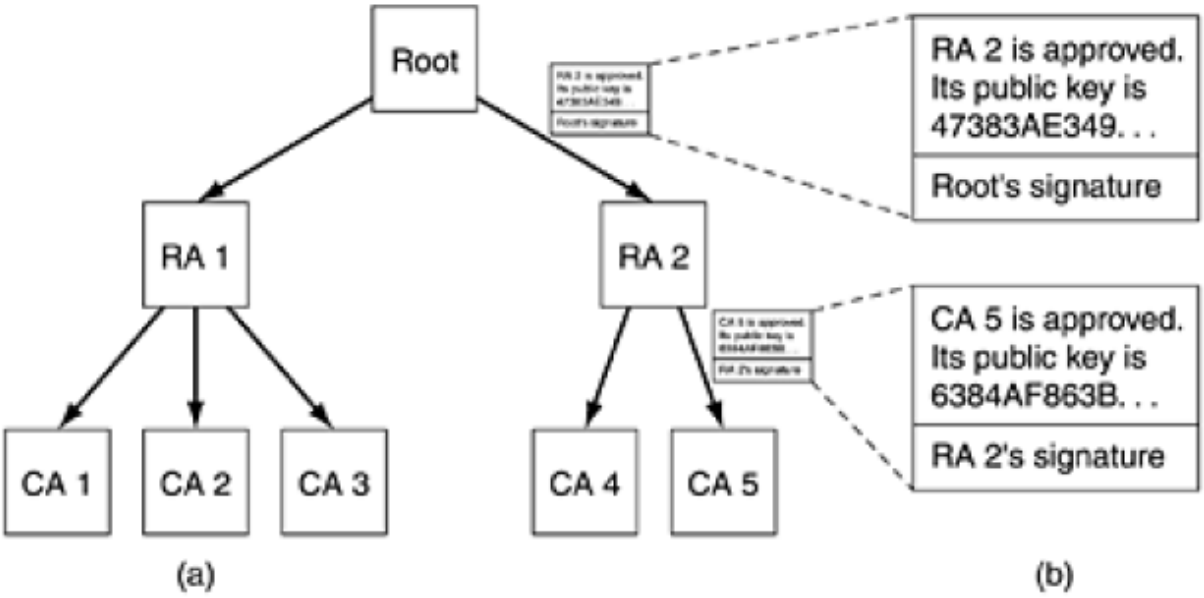

Figura 1 Ejemplo de estructura de PKI [13]

Una PKI bien construida debe proporcionar:

- Autenticidad. La firma digital tendrá la misma validez que la manuscrita.
- Confidencialidad de la información transmitida entre las partes.
- Integridad. Debe asegurarse la capacidad de detectar si un documento firmado ha sido manipulado.
- No repudio, de un documento firmado digitalmente.

1.4 Tecnología en Tarjeta Inteligentes

1.4.1 Java Card.

Esta tecnología permite en tarjetas inteligentes y similares dispositivos incrustados, ejecutar aplicaciones escritas en un subconjunto del lenguaje de programación Java. Ofrece la posibilidad y capacidad al usuario de implementar soluciones aplicadas en distintas esferas de la sociedad y la economía, con elevados niveles de seguridad, ejemplos claros los podemos encontrar en las aplicaciones que permiten la firma digital, la autenticación de usuarios, además en las tarjetas SIM (utilizadas en la telefonía celular), entre otros [14].

La arquitectura básica de una Java Card consiste de Applets, Java Card API, Java Card Virtual Machine (JCVM) / Java Card Runtime Environment (JCRE), y el sistema operativo nativo de la tarjeta.

Aplicaciones de Java Card

Desde el surgimiento de las tarjetas inteligentes el número de aplicaciones para Java Card ha ido aumentando considerablemente y en diversas ramas, ejemplos típicos se mencionan a continuación:

- ElectronicPurse o ElectronicWallet(ePurseoeWallet): esta aplicación es utilizada como dinero, donde luego de haber fijado un monto inicial, se puede realizar consultas, operaciones de débito o crédito. Normalmente para evitar fraudes u otros delitos esta tarjeta lleva asociado algún sistema de seguridad.
- Transacciones Seguras: Las tarjetas inteligentes son menos vulnerables que las tarjetas magnéticas debido a su nivel de seguridad, pues es normal que incluyan un API de criptografía fuerte[5].
- Identificación Digital/Firma Digital: Validar la identidad del portador de la tarjeta, o certificar el origen de ciertos datos es el principal objetivo de esta aplicación. Comúnmente se basa en las primitivas criptográficas del API y/o las que están implementadas en *hardware*.
- Sistemas de Prepago: El propietario de la tarjeta la carga con una cierta cantidad de servicios y según él vaya haciendo uso del servicio, entonces la cantidad de servicios se irá decrementando.
- Health Cards (Tarjetas de Salud): Este sistema es implementado en algunos hospitales para la identificación y almacenamiento de los principales datos de la historia clínica del paciente en tarjetas inteligentes, con el objetivo de proporcionar una mejor atención. En estos momentos la capacidad de almacenamiento es muy pequeña, pero día a día esto puede incrementar, llegando a poder almacenar toda la historia clínica.
- Control de Acceso y de Asistencia: Existen en la actualidad varios sistemas de control de acceso y asistencia implementados sobre tarjetas inteligentes.

Existen más aplicaciones de Java Card, aunque, algunas son variantes de las que se mencionan anteriormente, también hay otras que satisfacen necesidades específicas de una empresa [14].

Java Card Runtime Environment (JCRE)

El JCRE comprende la máquina virtual de Java Card (JCVM) junto a clases y servicios definidos en el Application Programming Interface (API). Sobre este entorno se ejecutan los applets desarrollados [10].

[5] Término general usado para los sistemas criptográficos aplicados o componentes que se consideran altamente resistente al criptoanálisis.

1.4.2 Estándares relacionados con tarjetas inteligentes

Estándar ISO/IEC- 7816

La tarjeta inteligente más básica cumple los estándares de la serie ISO/IEC 7816. El objetivo de estos es lograr la interoperabilidad entre distintos fabricantes de tarjetas y lectores de las mismas, en lo que respecta a características físicas, comunicación de datos y seguridad. Estos estándares son basados en los ISO 7810 e ISO 7811, los cuales definen características físicas de tarjetas de identificación. Las características de comunicación de las tarjetas sin contacto son definidas en estándares como el ISO/IEC 14443 [9].

Descripción de algunas partes del estándar ISO 7816:

- 7816-1: Características físicas.
- 7816-2: Dimensiones y ubicaciones de los contactos.
- 7816-3: Señales electrónicas y protocolo de transmisión.
- 7816-4: Comandos de intercambio inter-industriales.
- 7816-5: Sistema de numeración y procedimientos de registro.
- 7816-6: Elementos de datos inter-industriales.
- 7816-7: Comandos inter-industriales y consultas estructuradas para una tarjeta.
- 7816-8: Comandos inter-industriales relacionados con seguridad.
- 7816-9: Comandos adicionales inter-industriales y atributos de seguridad.
- 7816-10: Señales electrónicas y respuesta para reiniciar una tarjeta inteligente síncrona.

Estándar ISO/IEC 14443

ISO 14443 es un estándar internacional relacionado con las tarjetas inteligentes sin contacto gestionado conjuntamente por la Organización Internacional de Normalización (ISO) y Comisión Electrotécnica Internacional (IEC).

El estándar ISO 14443 consta de cuatro partes y se describen dos tipos de tarjetas: tipo A y tipo B. Las principales diferencias entre estos tipos son los métodos de modulación, codificación de los planos (parte 2) y el protocolo de inicialización de los procedimientos (parte 3). Las tarjetas de ambos tipos (A y B) utilizan el mismo protocolo de alto nivel (llamado T=CL) que se describe en la parte 4. El protocolo T=CL especifica los bloques de datos y los mecanismos de intercambio [9].

1. ISO/IEC 14443-1: Características físicas.

2. ISO/IEC 14443-2: Energía de radiofrecuencia y señal de interfaz.

3. ISO/IEC 14443-3: Inicialización y anticolisión.

4. ISO/IEC 14443-4: Protocolo de transmisión.

Estándar GlobalPlatform

Este estándar lidera mundialmente el desarrollo en temas de infraestructura de tarjetas inteligentes. Sus descripciones técnicas probadas para las tarjetas, dispositivos y sistemas son consideradas como las normas de la industria para lograr implementaciones interoperables, flexibles y sostenibles por tarjetas que soportan multi-aplicación y multi-actor e implementaciones multi-modelo de negocio [15].

GlobalPlatform es un estándar para la administración de la estructura de las tarjetas inteligentes, comprende la instalación y borrado de las aplicaciones y la administración de otras tareas de las tarjetas. El emisor de la tarjeta tiene el control total sobre el contenido de esta, pero puede permitir a otras instituciones administrar sus propias aplicaciones, esto se logra aplicando protocolos criptográficos, permitiendo a cada institución tener su propia área segura en la tarjeta [15] .

Estándar PKCS

Estándar que se refiere a un grupo importante de normas de criptografía de clave pública, previstos y luego publicados por los laboratorios de RSA de California, en cooperación con desarrolladores del resto del mundo, con la finalidad de apresurar el despliegue de la criptografía de clave pública.

PKCS #15

Estándar de formato de información de dispositivo criptográfico. Define un estándar que admite a los usuarios de dispositivo criptográficos identificarse con aplicaciones independientemente de la implementación de PKCS#11 que estas posean.

Este estándar comprende dos tipos de dispositivos, las tarjetas con circuito integrado y los *tokens* implementados completamente en software, para las tarjetas con circuito integrado desde enero del 2004 existe un estándar basado en el PKCS#15, denominado ISO/IEC 7816-15 [16].

En esta aplicación dicho estándar se utiliza para conformar la estructura de ficheros que permitirá al *middleware* almacenar objetos PKCS#11 (llaves, certificados). El sistema de ficheros definido por el estándar PKCS#15 requiere que la tarjeta soporte los tipos de ficheros definidos en la norma ISO/IEC 7816-4.

Para el estudio de la estructura interna del *applet* “uSafeApp” se analizó la arquitectura (Figura 2) de dicha aplicación para conocer que la capa de Gestión de archivos y comandos, se encarga de procesar los comandos ISO/IEC 7816 enviados a la tarjeta y el manejo de los archivos almacenados en la misma. Esta capa contiene el paquete Sistema de Ficheros ISO/IEC 7816 encargado de procesar los comandos y el Paquete

PKCS#15 encargado de crear la estructura de ficheros del *applet* utilizando el Sistema de Ficheros ISO/IEC 7816.

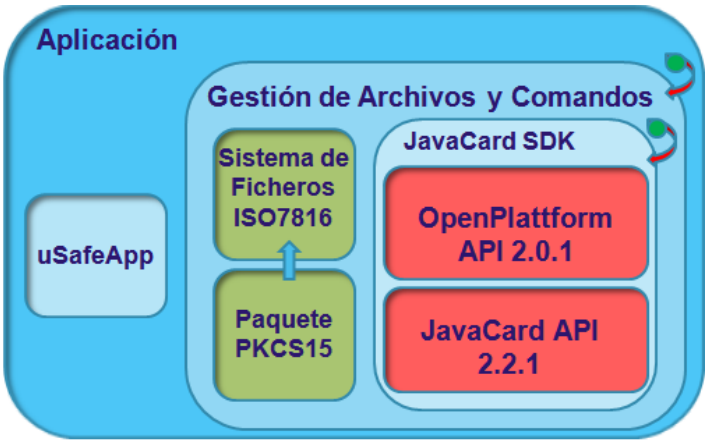

Figura 2 Arquitectura del applet "uSafeApp"[17]

PKCS #11

Estándar *RSA* usado para interactuar con los *tokens* criptográficos. Es el primero de los estándares que se han visto hasta el momento que tiene una auténtica API con funcionalidades concretas que pueden ser invocadas por una aplicación para realizar operaciones en una SmartCard. Este es conocido también como Cryptoki y por extensión a la librería que implementa la interfaz (en Windows será una DLL) también se la conoce con ese nombre.

PKCS#11 no está ideado para la gestión de todo tipo de SmartCards, sino sólo aquellas que tienen capacidades criptográficas (y generalmente de almacenamiento). Este estandar contempla dos visiones diferentes dentro de la criptografía: una primera, enfocada a PKI y a la firma digital (no se soporta el cifrado estándar asimétrico) y, por otro lado, la criptografía simétrica de clave secreta. En este proyecto, sólo se centrará en la primera parte (la de PKI), ya que, en el caso de la criptografía de clave secreta, las operaciones criptográficas no las realiza la tarjeta (ésta sólo almacena la clave) y es la librería de Cryptoki (una DLL software) la que realiza las operaciones [18].

UIT-T X.509

Un certificado es esencialmente una clave pública y un identificador, firmados digitalmente por una autoridad de certificación, y su utilidad es demostrar que una clave pública pertenece a un usuario concreto. El formato de certificados X.509 es el más común y extendido en la actualidad. El estándar X.509 sólo define la sintaxis de los certificados, por lo que no está atado a ningún algoritmo en particular y contempla los siguientes campos:

- Versión.
- Número de serie.

- Identificador del algoritmo empleado para la firma digital.
- Nombre del certificador.
- Período de validez.
- Nombre del sujeto.
- Clave pública del sujeto.
- Identificador único del certificador.
- Identificador único del sujeto.
- Extensiones.
- Firma digital de todo lo anterior generada por el certificador.

Estos certificados se estructuran de forma jerárquica, de manera tal que se podrá verificar la autenticidad de un certificado comprobando la firma de la autoridad que lo emitió, que a su vez tendrá otro certificado expedido por otra autoridad de rango superior. De esta forma vamos subiendo en la jerarquía hasta llegar al nivel más alto, que deberá estar ocupado por un certificador que goce de la confianza de toda la comunidad. Normalmente las claves públicas de los certificadores de mayor nivel se suelen publicar, para que cualquiera pueda verificarlas [8].

El mecanismo que debe emplearse para conseguir un certificado X.509 es enviar la clave pública a la autoridad de certificación, después de habernos identificado positivamente frente a ella. Existen autoridades de certificación que, frente a una solicitud, generan un par llaves pública-privada y las envían al usuario. Es importante destacar que en este caso, si bien tendremos un certificado válido, nuestro certificador podrá descifrar todos nuestros mensajes [2].

Estándar PC/SC

Personal Computer/Smart Card (PC/SC) son un conjunto de especificaciones para facilitar la comunicación entre las tarjetas inteligentes, las computadoras y los lectores de tarjetas inteligentes. Define un API de programación que permite a los desarrolladores trabajar con lectores de diversos fabricantes de una misma manera.

Hasta la fecha existen nueve partes, en las cuales se encuentran detalles sobre la interoperabilidad de los dispositivos, interfaces de programación o información de diseño [19]:

- Parte 1. Introducción y visión general de la arquitectura.
- Parte 2. Requisitos de interoperabilidad para las tarjetas y los lectores.
- Parte 3. Requisitos de interoperabilidad para los lectores.
- Parte 4. Consideraciones de diseño de los lectores e información de referencia.
- Parte 5. Definición de la interfaz del Resource Manager.

- Parte 6. Definición de la interfaz del Service Provider.
- Parte 7. Consideraciones de diseño para el desarrollo de aplicaciones
- Parte 8. Recomendación para la implementación de servicios de seguridad y privacidad con tarjetas inteligentes
- Parte 9. Lectores con capacidades extendidas

1.5 Análisis de soluciones existentes para PKI en tarjetas inteligentes

Durante el estudio de las soluciones existentes relacionadas con las tarjetas inteligentes en una infraestructura de clave pública, se identificaron un conjunto de productos. A continuación se describen algunos de ellos:

- Classic TPC: Tarjeta con aplicación PKI, privativa y compatible con PKCS#15 sobre la tecnología JavaCard y se integra al *middleware* ClassicClient. El proveedor de este producto es Gemalto [20].
- .Net Card: Solución privativa que incluye el *middleware* con PKCS # 11 (Windows, Linux y MAC) y CSP, y las tarjetas PKI y One Time Password (OTP) con tecnología .Net. Se integra a servicios de administración de identidades (Microsoft) y de autenticación. Su proveedor es Gemalto [21] .
- PrivateCard: Tarjeta con aplicación PKI, privativa que integra verificación biométrica con el MatchOnCard de Precise Biometrics. Su proveedor es ARX [22] .
- Java Card PKI applet: Solución libre desarrollada por Wojciech Mostowski que soporta canal seguro GlobalPlatform, tamaño de llave de 1024 RSA, dos PINs y tres llaves.

Después del resumen de las soluciones existentes relacionadas con las tarjetas inteligentes en una infraestructura de clave pública se llegó a la conclusión de que muchas de las analizadas son soluciones privativas, en el caso de la Java Card PKI applet se basó el “uSafeApp” pero le falta el *middleware* PKCS#11 y la solución que ofrece Gemalto: .Net Card, pesar de que la solución incluye un *middleware* con PKCS # 11 ésta es una solución privativa.

1.6 Herramientas y tecnologías de desarrollo

1.6.1 Extreme Programming (XP)

Está diseñada para que el cliente reciba el software que necesita en el tiempo que lo necesita. Su principal objetivo es satisfacer las necesidades de los usuarios y es por ello que se considera un éxito. El trabajo es desarrollado en equipo, preocupándose en

todo momento del aprendizaje de los desarrolladores y estableciendo un buen clima de trabajo. *Extreme Programming*[6] le confiere las facultades a los desarrolladores para que puedan responder con confianza a las necesidades cambiantes de los clientes. Es la metodología más apropiada para entornos volátiles. Este tipo de programación es la adecuada para los proyectos con requisitos imprecisos, muy cambiantes y con un riesgo técnico excesivo.

XP se basa en retroalimentación o *feedback* continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios [23, 24].

Para la selección de la metodología de desarrollo a utilizar, se hace necesario analizar el entorno en el que se construirá el producto. En primer lugar, es necesario destacar que los usuarios finales son los propios desarrolladores del departamento de Tarjetas Inteligentes, lo cual significa que en todo momento se estará interactuando con los mismos.

Los aspectos esenciales que incidieron para la elección de esta metodología son:

- Cantidad de miembros del equipo de desarrollo.
- Necesidad de obtener un producto a corto plazo.
- Presencia del cliente en el grupo de desarrollo.
- Asumir una metodología robusta implica una cantidad excesiva de roles y gran volumen de información generada durante todo el ciclo de vida del proyecto
- Los requisitos asociados a la aplicación pueden sufrir cambios debido al propio proceso de desarrollo del departamento de Tarjeta Inteligentes lo cual arrojará nuevas necesidades y por ende nuevas funcionalidades a incluir en el sistema.

1.6.2 UML (*Unified Modeling Language*)

De los lenguajes de modelado de sistemas de software el más conocido y utilizado actualmente es el Lenguaje Unificado de Modelado. Es un lenguaje de modelado visual que se usa para visualizar, especificar, construir y documentar un sistema de software. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. Además de ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo [25].

[6] La programación extrema o *Extreme Programming* (XP) es una metodología de desarrollo de la ingeniería de software.

1.6.3 Altova UModel

El Altova UModel es una herramienta CASE (Ingeniería de Software Asistida por Computadora) que es el punto de entrada para el desarrollo de software exitoso. Se utiliza para crear e interpretar diseños software mediante la potencia del estándar UML.

Permite modelar diseños de aplicaciones y generar código Java o C# a partir de los diagramas, o realizar ingeniería inversa de programas existentes a diagramas UML claros y precisos. Corrige el código generado o los modelos y produce automáticamente nuevos diagramas o regenera el código. De cualquier manera, UModel permite mantener el proyecto sincronizado y al día. También se soportan hiperenlaces entre diagramas UML y desde diagramas a ficheros externos o sitios Web [26, 27].

Se utiliza esta herramienta por su eficiencia, fácil manejo, hace que sea práctico para todos los programadores y gestores de proyecto, además que por su rica interfaz visual y usabilidad superior permite disminuir la curva de aprendizaje de UML, potenciando su productividad y maximizando los resultados esperados.

1.6.4 Qt-Creator

Para la selección de la herramienta Qt-Creator como tecnología *middleware* a utilizar, se hace necesario analizar el entorno en el que se construirá la solución, además de las restricciones de diseño que ésta posee.

Qt Creator es un Entorno Integrado de Desarrollo o IDE (esto es, editor + compilador + depurador) bastante completo, moderno, potente, fácil de manejar, eficiente, abierto y gratuito, que permite el desarrollo rápido de aplicaciones en entornos MS Windows, Mac OS y Linux. Este es una biblioteca multiplataforma ampliamente usada para desarrollar aplicaciones con interfaz gráfica de usuario, así como también para el desarrollo de programas sin interfaz gráfica, como herramientas para la línea de comandos y consolas para servidores [28].

Es desarrollada como un software libre y de código abierto a través de Qt Project, donde participa tanto la comunidad, como desarrolladores de Nokia, Digia y otras empresas. Esta herramienta utiliza el lenguaje de programación C++ de forma nativa, adicionalmente puede ser utilizado en otros lenguajes de programación a través de *bindings*[7]. También es usada en sistemas informáticos empotrados para las máquinas que se desplazan por la acción de un motor, aeronavegación y aparatos domésticos como frigoríficos.

[7] Un *binding* es una adaptación de una biblioteca para ser usada en un lenguaje de programación distinto de aquél en el que ha sido escrita.

Funciona en todas las principales plataformas, y tiene un amplio apoyo. El API de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL, así como uso de XML, gestión de hilos, soporte de red, una API multiplataforma unificada para la manipulación de archivos y una multitud de otros para el manejo de ficheros, además de estructuras de datos tradicionales.

1.6.5 Lenguaje C++

El lenguaje de programación C++ fue creado en los años 80 por Bjarne Stroustrup basando en el lenguaje C. Es un lenguaje orientado a objetos al que se le añadieron características y cualidades de las que carecía el lenguaje C haciéndolo increíblemente versátil. Usándolo se pueden programar las aplicaciones más simples y las más complicadas, incluso sistemas operativos.

Se selecciona este lenguaje por ser portable, es decir, un programa con el código escrito en C++, se podrá compilar en cualquier sistema operativo o sistema informático con sólo algunos cambios en el código fuente. Es un lenguaje multi-nivel, que puede ser usado tanto para programar directamente el hardware (en dependencia del sistema operativo), como para crear aplicaciones [29, 30]. Además que las mejores soluciones para middleware PKI en tarjetas inteligentes se encuentran en dicho lenguaje y se tiene como restricción de implementación, debido a que las aplicaciones cliente de las que se mencionan anteriormente, solo pueden utilizar librerías o módulos compilados como binarios “nativos” que implementen el estándar PKCS#11.

1.7 Propuesta y selección de herramientas

Dada la necesidad de implementar un *middleware* para el *applet* de PKI “uSafeApp” para tarjetas inteligentes en una Infraestructura de Clave Pública, se determina que esta se va a desarrollar utilizando el IDE de desarrollo Qt-Creator, permitiendo implementar un *middleware* que hace uso del *applet* “uSafeApp” para realizar las operaciones de firma digital y autenticación, utilizando certificados digitales. Para interpretar las necesidades del sistema y diseñar la solución propuesta se utilizará la metodología ágil XP por ajustarse a las condiciones del proyecto y el equipo. Se modelará además utilizando la herramienta Altova UModel, definida por el departamento de tarjetas inteligentes, cumpliendo de esta forma con los requisitos que se hacen necesarios para la implementación de la solución: Middleware para el *applet* de PKI “uSafeApp”.

1.8 Conclusiones Parciales

- El análisis de las soluciones existentes sobre el uso de tarjetas inteligentes en una Infraestructura de Clave Pública, permitió identificar los principales

elementos de seguridad y los estándares asociados a este tipo de tecnologías, facilitando su comprensión.

- El análisis las soluciones existentes relacionadas con las tarjetas inteligentes en una infraestructura de clave pública evidenció de que muchas de las soluciones analizadas son privativas o no tienen un *middleware* que implemente el estándar PKCS#11.
- Para el desarrollo de la solución se utilizará como lenguaje de modelado UML, como herramienta CASE Altova UModel, como lenguaje de programación C++, como IDE de desarrollo Qt-Creator y como metodología de desarrollo de software Extreme Programming (XP).

CAPÍTULO 2: ANÁLISIS y DISEÑO

2.1 Introducción

La solución a desarrollar, debe ser fruto de un análisis correcto y una comprensión de todos los elementos que se relacionan en correspondencia con el tema de los componentes que interactúan en la solución, profundizándose en el estudio de todas las características que posibiliten desarrollar los mismos. En este capítulo se hace un estudio de los principales aspectos en los que se enmarca el problema concluyendo que se debe realizar una modelación de dominio, identificando para esto las entidades principales que se tendrán y las relaciones que se establecen entre ellas. La propuesta de solución será modelada haciendo uso de la metodología de desarrollo de software XP, las necesidades del sistema serán especificadas mediante requerimientos funcionales y no funcionales.

En este capítulo se expone el avance de la solución durante las fases iniciales de Planificación y Diseño, donde se plantean de manera precisa las historias de usuarios, al mismo tiempo que el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas, se exploran las posibilidades de la arquitectura del sistema; además se establece la prioridad de cada historia de usuario y los programadores realizan una estimación del esfuerzo necesario de cada una de ellas.

2.2 Descripción del Sistema

El sistema está diseñado con tres componentes.

Una aplicación cliente que se encarga de interactuar con el usuario y de solicitar al *middleware* las funcionalidades requeridas por el usuario tales como: firmar un documento, verificación de pin, generar llave, entre otras.

El *middleware* “uSafeMdw” actúa de intermediario entre las aplicaciones cliente (cliente de correo, Mozilla Firefox, Acrobat Reader) y el *applet* “uSafeApp”, enviando y recibiendo los APDU para la comunicación con el *applet*.

El *applet* “uSafeApp” recibe los comandos APDU enviados por el *middleware* y envía un APDU respuesta al *middleware* de la funcionalidad solicitada.

2.3 Modelo de Dominio

Se realizó un modelo de dominio haciendo uso de la herramienta Altova UModel[8], como punto de partida para el diseño del sistema, este se utiliza para mostrar de manera visual los principales conceptos que se manejan, ayudando así a los usuarios,

[8] Herramienta UML.

desarrolladores e interesados a utilizar un vocabulario común para poder entender el contexto en que se desarrolla la solución. En la figura 3 se muestra el diagrama de clases del modelo de dominio.

2.3.1 Diagrama de Clases del Modelo de Dominio

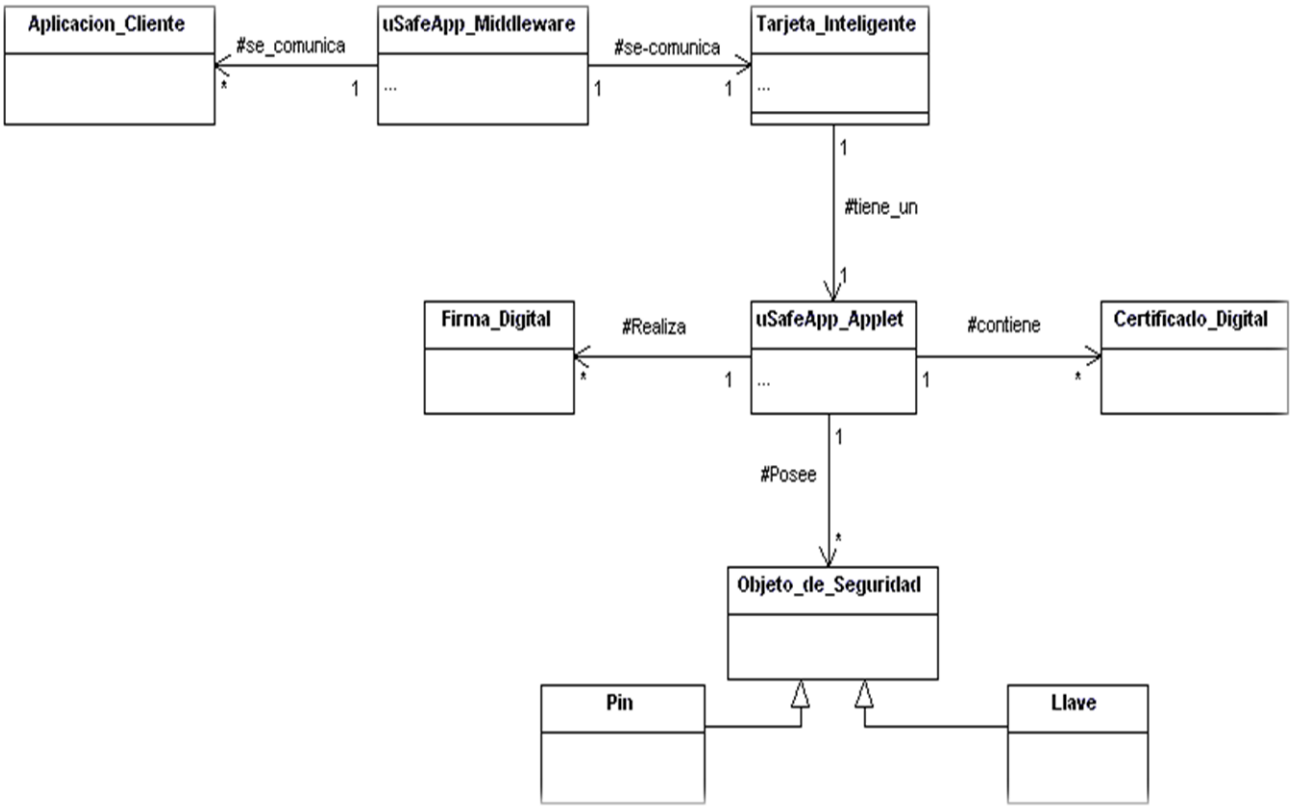

Figura 3 Diagrama de clase del modelo de dominio

2.3.2 Glosario de Conceptos del Modelo de Dominio

Aplicación Cliente: Encargado de abstraer al desarrollador del envío y recepción de los comandos que se intercambian entre el *middleware* y el *applet*.

Certificado_Digital: Es la combinación de la llave pública que ha sido firmada por la autoridad de certificación, la firma digital de esa llave pública y algunos parámetros adicionales. El formato del certificado digital está regido por el estándar *UIT-T X.509*.

Firma_Digital: Esquema matemático que es utilizado para garantizar la autenticidad de documentos electrónicos o mensajes transmitidos electrónicamente.

Llave: Es una secuencia de caracteres, que puede contener letras, dígitos y símbolos, y que es convertida en un número, utilizada por los algoritmos de cifrado para codificar y decodificar mensajes.

Pin (Número de Identificación Personal): es un valor numérico usado para identificarse y poder acceder a la información de la tarjeta.

Tarjeta Inteligente: Es un dispositivo de plástico similar en tamaño y otros estándares físicos a las tarjetas de crédito, presenta un circuito integrado, el mismo puede ser de sólo memoria o contener un microprocesador con un sistema operativo que le permita una serie de funcionalidades:

UsafeApp_Applet: Aplicación *Java Card* para tarjetas inteligentes, que cumple con los principales estándares internacionales asociados, y con la capacidad de realizar operaciones criptográficas y de gestión de certificados digitales, en una Infraestructura de Clave Pública

uSafeApp_Middleware: Encargado de enviar un APDU con la instrucción que se desea realizar.

2.4 Especificaciones de los requerimientos de Software

2.4.1 Requerimientos Funcionales

Los Requerimientos funcionales especifican acciones que el sistema debe ser capaz de realizar, sin tomar en consideración ningún tipo de restricción física [31]. Es decir, especifican el comportamiento de entrada y salida del sistema y surgen de la razón fundamental de la existencia del producto.

Se definieron los siguientes requisitos para la aplicación:

- RF_1 Obtener lectores disponibles.
- RF_2 Inicializar conexión.
- RF_3 Enviar comandos APDU de la tarjeta.
- RF_4 Establecer canal seguro.
- RF_5 Verificar PIN.
- RF_6 Cambiar PIN.
- RF_7 Generar par de llaves.
- RF_8 Obtener par de llaves.
- RF_9 Realizar firma.
- RF_10 Obtener certificados digitales.
- RF_11 Cargar certificados digitales.
- RF_12 Realizar desconexión.

2.4.2 Requerimientos no Funcionales

Los requerimientos no funcionales son las propiedades o cualidades que el producto debe tener para que este sea atractivo, usable, rápido y confiable [31].

Requerimientos de software:

- Para la realización de las pruebas funcionales a los componentes se requiere que estén instalados en la PC cliente los drivers del lector de tarjetas.
- En la PC cliente se podrá utilizar Microsoft Windows en cualquiera de sus versiones.

Requerimientos de hardware recomendados para el desarrollador:

- Lector de tarjetas que cumpla con el estándar PC/SC versión 1.0 o superior.
- Procesador de 1 GHz de 32 bits o 64 bits.
- 1 GB de memoria RAM en el sistema.

Requerimientos de diseño e implementación:

- Lenguaje de programación: *C++.*
- IDE de desarrollo a utilizar: Qt-Creator.

2.5 Planificación

El propósito de la fase de planificación es establecer un acuerdo entre los clientes y desarrolladores sobre el menor tiempo en que la mayor cantidad de Historias de Usuarios puedan ser realizadas. Para ello se realizó la planificación con el objetivo de maximizar el valor del software producido a partir de la puesta en producción de las características más importantes lo antes posible, se realiza una estimación del esfuerzo requerido para la implementación de las HU y se define el tiempo de entrega y cada iteración.

2.5.1 Historia de Usuario

Las historias de usuarios se utilizan en la metodología XP para describir los requisitos de una forma menos detallada y desde la perspectiva del cliente. La forma de tratar estas historias es muy flexible, pudiéndose romper en cualquier momento, cambiar o añadir nuevas. Cada historia de usuario debe ser comprensible y estar delimitada, de forma que los programadores puedan realizarla en unas semanas.

Tabla 1 HU_1 Obtener lectores disponibles

Historia de Usuario.	
Número: HU_1.	**Usuario:**
Nombre historia: Obtener lectores disponibles.	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Baja.
Puntos estimados: 2.	**Iteración asignada:** 1.

Programador responsable: Amanda Cabrera Miralles, Fernando José Enríquez Fernández.
Descripción: Utilizando la función SCardListReaders de la winscard.dll se obtiene los lectores disponibles.
Observaciones: Si el usuario quiere realizar alguna operación y la tarjeta no está en estado: conectada en el lector; el sistema notifica un error.

Tabla 2 HU_1 Inicializar conexión

Historia de Usuario.	
Número: HU_2.	Usuario:
Nombre historia: Inicializar conexión.	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Medio.
Puntos estimados: 2.	Iteración asignada: 2.
Programador responsable: Amanda Cabrera Miralles, Fernando José Enríquez Fernández.	
Descripción: Utilizando la función SCardConnect de la winscard.dll se realiza la conexión con la tarjeta inteligente.	
Observaciones: Si el usuario quiere realizar la operación de conectar y la tarjeta no está en estado conectada en el lector, el sistema notifica un error.	

Tabla 3 HU_3 Enviar comandos APDU de la tarjeta

Historia de Usuario.	
Número: HU_3.	**Usuario:**
Nombre historia: Enviar comandos APDU de la tarjeta.	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Baja.
Puntos estimados: 2.	**Iteración asignada:** 1.
Programador responsable: Amanda Cabrera Miralles, Fernando José Enríquez Fernández .	
Descripción: Utilizando la función SCardTransmit de la winscard.dll se realiza el envío de comandos APDU con la tarjeta inteligente.	
Observaciones: Si la tarjeta no está en estado conectada, el sistema notifica un error. En caso de que la tarjeta sea retirada del lector el sistema notificará.	

Tabla 4 HU_4 Establecer canal seguro

Historia de Usuario	
Número: HU_4.	**Usuario:**
Nombre historia: Establecer canal seguro.	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Baja.
Puntos estimados: 2.	**Iteración asignada:** 1.
Programador responsable: Amanda Cabrera Miralles, Fernando José Enríquez Fernández .	
Descripción: Se envían comandos APDU para obtener los datos para ejecutar la autenticación mutua simétrica, en la fase de aplicación.	
Observaciones: Si la tarjeta no está en estado conectada en el lector se lanza una excepción. Si el comando APDU enviado es incorrecto el sistema lanza una excepción. Si la autenticación mutua es incorrecta el sistema lanza una excepción.	

Tabla 5 HU_5 Verificar Identidad mediante PIN

Historia de Usuario	
Número: HU_5.	**Usuario:**
Nombre historia: Verificar Identidad mediante pin.	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Baja.
Puntos estimados: 2.	**Iteración asignada:** 1.
Programador responsable: Amanda Cabrera Miralles, Fernando José Enríquez Fernández.	
Descripción: Una vez autenticado, el usuario procede a entrar su PIN de identificación.	
Observaciones: En caso de que el PIN sea incorrecto se le notifica al usuario.	

Tabla 6 HU_6 Cambiar PIN.

Historia de Usuario	
Número: HU_6	**Usuario:**
Nombre historia: Cambiar PIN.	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Baja.
Puntos estimados: 2.	**Iteración asignada:** 1.
Programador responsable: Amanda Cabrera Miralles, Fernando José Enríquez Fernández.	
Descripción: El usuario confirmará su PIN de identificación y luego el nuevo introducirá el nuevo PIN.	
Observaciones: En caso de que el PIN de identificación no sea correcto se le notifica al usuario.	

Tabla 7 HU_7 Generar par de llaves.

Historia de Usuario	
Número: HU_7	**Usuario:**
Nombre historia: Generar par de llaves.	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Baja.
Puntos estimados: 2.	**Iteración asignada:** 1.
Programador responsable: Amanda Cabrera Miralles, Fernando José Enríquez Fernández.	
Descripción: Una vez autenticado el usuario, se conforma el comando GeneratePublicKey PairCommand para generar el par de llaves en la tarjeta.	
Observaciones: Si el usuario no se ha autenticado correctamente el sistema notificará al usuario.	

Tabla 8 HU_8 Obtener par de llaves.

Historia de Usuario	
Número: HU_8.	**Usuario:**

Nombre historia: Obtener par de llaves.	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Baja.
Puntos estimados: 2.	**Iteración asignada:** 1.
Programador responsable: Amanda Cabrera Miralles, Fernando José Enríquez Fernández.	
Descripción: Una vez autenticado el usuario se conforma el comando GetDataTLVCommand para obtener el par de llaves en la tarjeta.	
Observaciones: Si el usuario no se ha autenticado correctamente el sistema notificará al usuario.	

Tabla 9 HU_9 Realizar Firma

Historia de Usuario	
Número: HU_9.	**Usuario:**
Nombre historia: Realizar Firma.	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Baja.
Puntos estimados: 2.	**Iteración asignada:** 1.
Programador responsable: Amanda Cabrera Miralles, Fernando José Enríquez Fernández.	
Descripción: Una vez que el usuario escoge el documento que desea firmar se conforma el comando PSOSignCommand para realizar la firma en la tarjeta.	
Observaciones: Si el usuario no se ha autenticado correctamente el sistema notificará al usuario.	

Tabla 10 HU_10 Obtener certificado

Historia de Usuario	
Número: HU_10.	**Usuario:**
Nombre historia: Obtener certificado.	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Baja.

Puntos estimados: 2.	**Iteración asignada:** 1.
Programador responsable: Amanda Cabrera Miralles, Fernando José Enríquez Fernández .	
Descripción: Se obtiene el certificado digital del *applet* mediante el identificador de dicho certificado.	
Observaciones: Si el usuario no se ha autenticado correctamente el sistema notificará al usuario. Si en el *applet* no encuentra el certificado digitar requerido se notificará al usuario.	

Tabla 11 HU_11 Cargar certificados digitales

Historia de Usuario	
Número: HU_11	**Usuario:**
Nombre historia: Cargar certificados digitales.	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Baja.
Puntos estimados: 2.	**Iteración asignada:** 1.
Programador responsable: Amanda Cabrera Miralles, Fernando José Enríquez Fernández.	
Descripción: Se muestra la lista de certificados digitales en la tarjeta.	
Observaciones: Si el usuario no se ha autenticado correctamente el sistema notificará al usuario.	

Tabla 12 HU_12 Realizar desconexión

Historia de Usuario	
Número: HU_12	**Usuario:**
Nombre historia: Realizar desconexión.	
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Baja.
Puntos estimados: 2.	**Iteración asignada:** 1.
Programador responsable: Amanda Cabrera Miralles, Fernando José Enríquez Fernández.	
Descripción: Utilizando la función SCardDisconnect de la winscard.dll se realiza la	

desconexión con la tarjeta inteligente.
Observaciones: Si el usuario quiere realizar la operación de desconectar y la tarjeta no está en estado conectada en el lector, el sistema notifica un error.

2.5.2 Estimación de Esfuerzo

En la metodología XP, la creación del sistema se divide en tres etapas. Normalmente, los proyectos suelen tener más de tres etapas, cada una de las cuales toma el nombre de iteración. Es por ello que esta metodología se dice que es iterativa. La duración ideal de una iteración está entre una y tres semanas.

Para cada una de las iteraciones se establecen un conjunto de historias que se van a llevar a cabo y un módulo, dando como resultado de cada iteración la entrega de dicho módulo, habiendo superado éste las pruebas de aceptación establecidas por el cliente de forma que verifique los requisitos.

Los desarrolladores estiman cuanto tiempo es necesario para implementar cada una de las historias de usuarios. Si ese tiempo supera en alguna las tres semanas se debe desglosar en otras más pequeñas. Si es inferior a una semana, la historia de usuario ha descendido a un nivel de detalle excesivo y habrá que combinarla con otra. Esto se hace concluida la fase de exploración. Como se ha dicho anteriormente, este valor es estimado y se irá acercando a la realidad con el transcurso de las iteraciones. En la presente solución se han identificado 12 historias de usuarios y quedan definidas las dos iteraciones con un total de 22 semanas de duración; de ser necesaria otra iteración se valoraría.

Para el desarrollo de la aplicación se realizó una estimación de esfuerzo por cada una de las historias de usuario identificadas por el cliente. Las estimaciones de esfuerzo asociado a la implementación se determinan utilizando como medida la cantidad de puntos. Se le dan valores de 1 a 3 puntos donde un punto equivale a una semana de programación [23, 32]. Especificando un tiempo estimado para la elaboración de cada una en base a una semana de 5 días y un día de 8 horas (Ver Anexo 1: Estimación de Esfuerzo).

2.5.3 Plan de Entrega

Después de tener ya definidas las historias de usuarios es necesario crear un plan de publicación. En este plan los desarrolladores y clientes establecen los tiempos de implementación ideales de las historias de usuario y la prioridad con la que serán implementadas. Al final se propone una fecha para entregar el producto, con el tiempo esta fecha puede variar, pero lo más seguro es que esta se acerque mucho a la

planificación. El plan de entrega especifica exactamente qué historias de usuario serán implementadas en cada entrega del sistema y sus prioridades, de modo que también permita conocer con exactitud qué historias de usuario serán implementadas en la próxima liberación [23]. Para la elaboración del plan de entrega de este proyecto y aplicando los parámetros de desarrollo bajo la metodología xp, se establece el tiempo calendario de acuerdo a un mes de 4 semanas y una semana de 5 días y un día de 8 horas. (Ver Anexo 2: Plan de entrega).

2.5.4 Plan de Iteraciones

La planificación en iteraciones y el diseño iterativo permite mantener discusiones continuas acerca de los problemas y avances realizados en las tareas, fomentando así la comunicación entre las partes involucradas y ayudando a la optimización de tiempo y esfuerzo empleado por el correcto desarrollo del proyecto [24, 32]. Para la elaboración del plan de iteración de este proyecto, se leyeron y detallaron las historias de usuarios, a las que luego se le asignaron tareas en cada una de las iteraciones que se van a llevar a cabo, dando como resultado de cada iteración la entrega de un módulo, habiendo superado éste las pruebas de aceptación establecidas por el cliente de forma que verifique los requisitos (Ver Anexo 3: Plan de Iteraciones).

2.5.5 Tareas de ingeniería

Todo el trabajo de la iteración es expresado en tareas de programación, las cuales se realizan para especificar las acciones llevadas a cabo por los programadores en cada historia de usuario, ya que no ofrecen el nivel de detalle requerido para saber qué implementar. Según el Plan de Iteraciones las historias de usuario se agruparon en tres iteraciones. (Ver Anexo 4: Tareas de Ingeniería).

2.6 Diseño

La metodología XP hace especial énfasis en los diseños simples y claros. Sólo se diseñan aquellas historias de usuario que el cliente ha seleccionado para la iteración actual. Kent Beck dice que en cualquier momento el diseño adecuado para el software es aquel que: supera con éxito todas las pruebas, no tiene lógica duplicada, refleja claramente la intención de implementación de los programadores y tiene el menor número posible de clases y métodos

2.6.1 Metáfora

Cada proyecto XP es guiado por una metáfora global. Estas ayudan a cualquier persona a entender el objetivo del programa, principalmente al equipo ya que aporta un contexto para entender los elementos básicos y sus relaciones proporcionando integridad conceptual. Es de vital importancia que el cliente y los desarrolladores estén

de acuerdo y conozcan la metáfora a usar para así poder trabajar y discutir en los mismos términos de una forma más precisa y de dominio de todos.

El *middleware* que funcionará como componente para la comunicación entre la tarjeta inteligente y las aplicaciones: cliente de correo, Acrobat Reader, Mozilla Firefox, cumpliendo con el estándar PKCS # 11, este permitirá poder hacer uso del *applet* "uSafeApp" con el objetivo de poder utilizar cada una de sus funcionalidades: firma digital y generación de llaves. El sistema confirmará que el usuario es el real portador de la tarjeta a través de la comparación del PIN (que es para proteger las operaciones de cifrado que requiera el uso de llaves privadas) introducido por este y el que se encuentra almacenado en la tarjeta, a partir de entonces se procede a la obtención de los datos almacenados en la misma que permitirán poder hacer uso de sus funcionalidades.

2.6.2 Definición de las tarjetas CRC

Tabla 13 Tarjeta CRC clase SCReaderWrapper

Nombre de la Clase: SCReaderWrapper	
Funcionalidades: - initialize - VerifyResponse - VerifyPin - PutDataPin - PutDataPuk - selectApplication - InicializeSecureChannel - OpenSecureChannel - setContext - listReaders - getReader - getConnected - connect - disconnect - transmit - getScreader - Sign	**Clases Asociadas:** - SmartCardUtil - SCardExeption - APDU - CommandAPDU - SelectCommad - GetDataCommad - VerifyPINCommad - MSESetCommand - ResponseAPDU - CustomException - SecureChannel - SecureChannelConfiguration
Descripción: Clase encargada de la conexión, desconexión, enviar APDU, establecer el canal seguro, listar los lectores, inicializar el canal seguro, verificar PIN, verificar las respuestas APDU.	

Tabla 14 Tarjeta CRC de la clase SmartCardUtil

Nombre de la Clase: SmartCardUtil	
Funcionalidades: - StringtoArrayByte - convert_string_to_byte_ptr - SubByteArray - ByteArrayToHexString - GenerateRandomData - DESPadding - ByteArrayListConcat - SymmetricCipher - RetailMAC - CompareByteArray	**Clases Asociadas:** - SCardExeption - APDU - ResponseAPDU - CustomException - cryptopp562
Descripción: Clase encargada de la conversión de datos y encriptar.	

Tabla 15 Tarjeta CRC de la clase SecureChannel

Nombre de la Clase: SecureChannel	
Funcionalidades: - MutualAuthentication - VerifyCardMutualAuthenticateResponse	**Clases Asociadas:** - SCardExeption - APDU - ResponseAPDU - CustomException - SmartCardUtil - MutualAuthenticationCommand - SCReaderWrapper - SecureChannelConfiguration
Descripción: Clase encargada de efectuar la autenticación mutua simétrica guardar los datos del canal seguro.	

Tabla 16 Tarjeta CRC de la clase APDU

Nombre de la Clase: APDU	
Funcionalidades: - getBuffer - setBuffer - toString	**Clases Asociadas:** - CustomException
Descripción: Clase encargada de conformar el apduBuffer.	

2.6.3 Diagrama de Clases

Los diagramas de clases sirven para representar la estructura estática de un sistema incluyendo una colección de elementos estáticos, tales como clases y relaciones [33]. Representan las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. Sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia y de uso. Un diagrama de clases está compuesto por los siguientes elementos: Clase: atributos, métodos y visibilidad. Relaciones: Herencia, Composición, Agregación, Asociación y Uso [34].

En el diagrama de clases que se muestra en la imagen, la clase SCReaderWrapper es la clase controladora encargada de comunicarse con el *applet* “uSafeApp” mediante el lector, sus funciones son el envió de APDUs, iniciar la comunicación, establecer el canal seguro con el *applet* “uSafeApp”. Al mismo tiempo que se encarga de verificar PIN. Utiliza la clase SmartCardUtil, que a su vez usa las clases de la librería criptográfica cryptopp, que brinda utilidades para convertir datos y encriptar. También utiliza la clase SecureChannel que permite efectuar la autenticación mutua simétrica y guarda los datos del canal seguro, usa la clase SecureChannelConfiguration y la clase SecureSession. Además la clase SCReaderWrapper hace uso de la clase SCardException encargada de gestionar las excepciones que tengan que ver con la tarjeta inteligente, además utiliza la clase CustomException para el tipo de excepciones comunes, utiliza la clase APDU encargada de crear el buffer APDU, de esta hereda la clase CommandAPDU encargada de crear los comandos APDU, de esta heredan los distintos tipos de APDU como son SelectCommand, GetDataCommand, VerifyCommand, MutalAuthenticateCommand, entre otros.

Figura 4 Diagrama de clase del middleware "uSafeMdw"

2.6.4 Arquitectura del Sistema

El *middleware* implementa los procesos definidos en los requerimientos de la solución. Este actúa como un componente que opera como una capa que abstrae, que aísla al usuario de la comunicación directa con las operaciones que realiza el *applet* “uSafeApp”.

Para poder desarrollar el middleware, se pasa por una fase de diseño del mismo, definiendo en este, la arquitectura que tendrá para un posterior desarrollo. En la siguiente figura se muestra la arquitectura que tendrá el middleware.

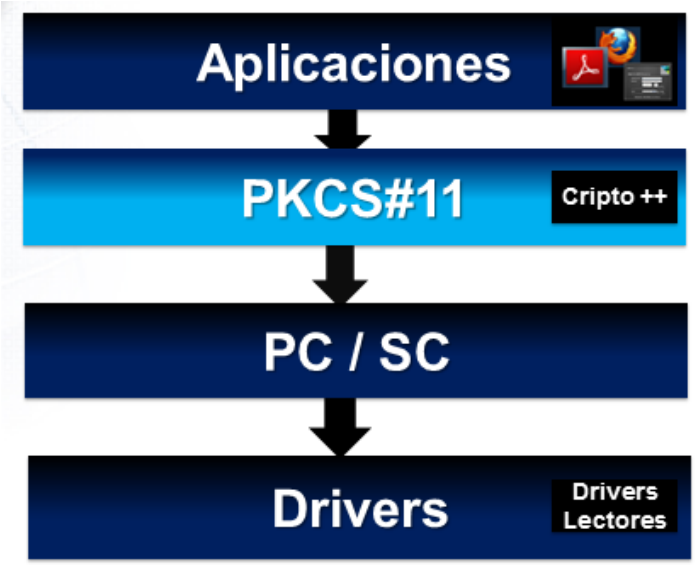

Figura 5 Arquitectura del Sistema

La arquitectura que se muestra, representa las distintas capas que necesita la solución para su funcionamiento. La capa de aplicaciones, donde estas pueden ser: navegador Mozilla Firefox, cliente de correo o Acrobat Reader, entre otras, esta capa accede a la capa inferior PKCS #11 que es la capa que se implementa como solución, en este caso, esta capa haciendo uso de Cripto++ para las operaciones criptográficas, se comunica con la capa PC/CS para la comunicación con los lectores y esta a su vez se comunica con los Drivers para que el lector pueda comunicarse con la tarjeta.

2.6.5 Patrones de Diseño

Los patrones de diseño son las soluciones a los principales problemas comunes en el desarrollo de software. Estos brindan una solución ya probada y documentada a un problema en el desarrollo del software que están sujetos a contextos similares, identifican Clases, Roles, Colaboraciones y la distribución de responsabilidades [35, 36].

Se utilizaron para el diseño de la solución los patrones GRASP, los cuales describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable.

Creador: Permite decidir cuáles serán las clases creadoras de otras clases. La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia, es útil contar con un principio general para la asignación de las responsabilidades de creación. Este patrón se utiliza en las clases entidades, cuando estas se instancian y crean instancias de clases contenidas en las mismas, ejemplo: La clase SCReaderWrapper se encarga de crear objetos de tipo SmartCardUtil para realizar operaciones auxiliares.

Alta cohesión: permite que una clase contenga responsabilidades moderadas en un área funcional, y colabora con las otras para llevar a cabo las tareas, incrementa la claridad y facilita la comprensión del diseño. Se pone de manifiesto en la clase SelectCommand que se encarga únicamente de crear un comando de tipo select.

2.7 Conclusiones Parciales

- La realización del modelo de domino determinó las bases brindar una visión más clara de los componentes y los conceptos asociados a la solución propuesta, así como las relaciones entre estos, que son determinados luego del estudio de las herramientas, tecnologías y elementos tangibles asociados al dominio de la solución.
- Los principales artefactos logrados fueron las Historias de Usuario, obteniendo un enfoque más claro y objetivo de los requerimientos del cliente. También se generó el Plan de Iteraciones, y el Plan de Entrega, determinado en 22 semanas como tiempo estimado para la realización de la solución, divididas en 2 iteraciones, y estas iteraciones se expresaron en tareas de programación, para ofrecer el nivel de detalle requerido para saber qué deberían de implementar los desarrolladores.
- La realización de una selección de los patrones de diseño tanto para implementar la solución como para diseñarla a nivel de principios generales, mostró a partir de los diagramas de clase que se brinda una solución ya probada y documentada para dar solución al problema planteado.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1 Introducción

A partir del marco de trabajo definido en la sección anterior, se define a continuación la evolución del procedimiento, durante las fases de Iteraciones a primera liberación y producción, además de desarrollar las pruebas del sistema para verificar la implementación de las historias de usuarios, ya que las pruebas funcionales permiten verificar que el sistema en desarrollo satisface a éstas. Se explica también el diseño de la solución así como los principales componentes y sus relaciones.

3.2 Diagrama de Componentes

En los diagramas de componentes se muestran los elementos de diseño de un sistema de software. Un diagrama de componentes permite visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces. Permite describir un diseño que se implemente en cualquier lenguaje o estilo. Solo es necesario identificar los elementos del diseño que interactúan con otros elementos del diseño a través de un conjunto restringido de entradas y salidas [37].

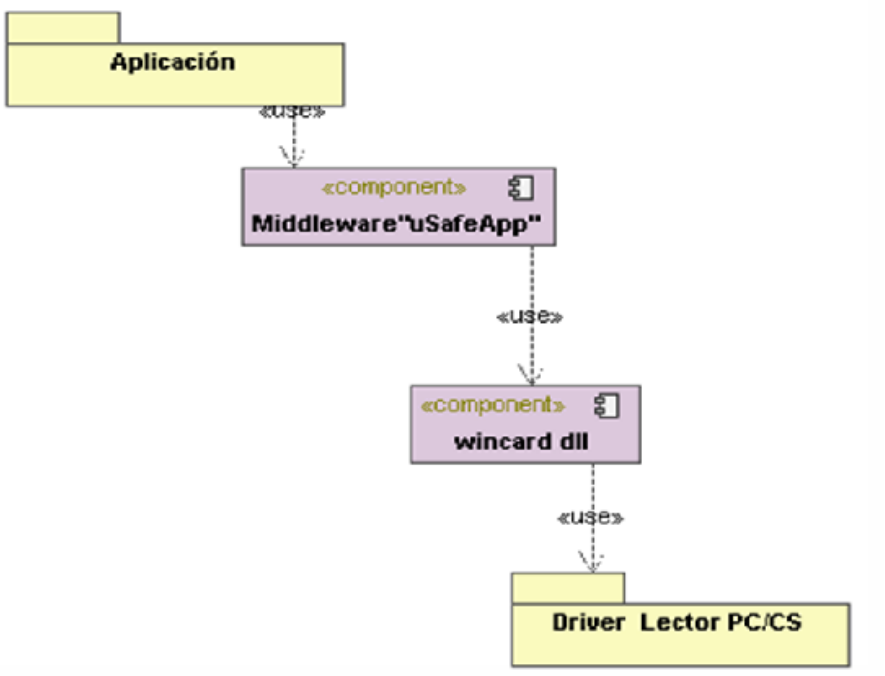

Figura 6 Diagrama de componentes del sistema

Aplicación: Componente encargado de la comunicación con el *middleware.*

Middleware: Componente encargado de enviar y recibir todos los comandos APDU entre el *applet* PKI "uSafeApp" y la aplicación para integrar las funcionalidades del *applet* "uSafeApp".

Winscard.dll: DLL del sistema operativo Windows que contiene las funciones para la conexión y envío de comandos APDU a la tarjeta.

Driver Lector PC/SC: Driver necesario para la comunicación entre el lector el sistema.

3.3 Diagrama de Despliegue

Los diagramas de despliegue permiten modelar la disposición física o topología de un sistema. Muestra el hardware usado y los componentes instalados en el hardware, las conexiones físicas entre el hardware y las relaciones entre componentes [38]. Para la muestra de las relaciones que se establecen entre los componentes de software y hardware, se realiza una representación mediante nodos estrechamente conectados.

La figura 7 muestra el despliegue físico de la solución.

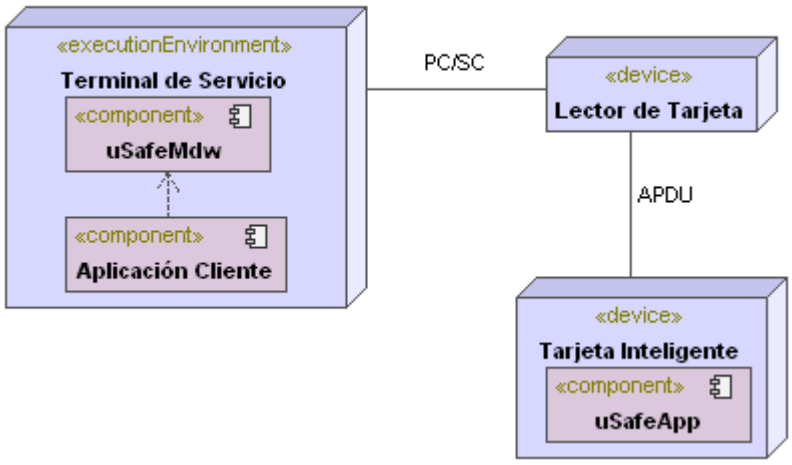

Figura 7 Diagrama de despliegue del sistema

Descripción de los Nodos:

Terminal de servicio: Este nodo contiene los componentes uSafeMdw y Aplicación cliente que permite a través del lector comunicarse con la tarjeta, pedante el uso del estándar PC/SC.

Lector de tarjeta: Este nodo representa el dispositivo que hará lectura de la tarjeta y con el que se podrá acceder al applet "uSafeApp".

Tarjeta Inteligente: Este nodo representa el dispositivo donde se encuentra el applet "uSafeApp", al cual se podrá acceder mediante el protocolo de comunicación APDU.

3.4 Pruebas

Para la fase de prueba el equipo de desarrollo se centró en los pedidos del cliente, refinando el diseño y el código continuamente. El perfeccionamiento de código sólo fue posible a través de un grupo de pruebas unitarias automatizadas que aseguraron la ejecución correcta del sistema en todo el período de desarrollo.

Frecuentemente cuando se desarrolla, siempre se realizan una gran cantidad de pruebas para verificar que el código esté correcto. Estas pruebas normalmente tienen que ser realizadas varias veces y se ven afectadas por los cambios que se introducen conforme se va desarrollando.

XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código (diseñada por los programadores) y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida (diseñadas por el cliente final).

3.4.1 Pruebas de Aceptación

Mediante la planificación y en base a las especificaciones de las historias de usuarios, se crea las pruebas de aceptación, también denominadas pruebas de funcionalidad, las mismas son constantes y constituyen uno de los pilares básicos de la metodología XP, permitiendo reducir el número de errores e incrementar la calidad del producto. A continuación se muestran los casos de pruebas de la solución.

Tabla 17 HU1_CP1_Obtener lectores disponibles.

Caso de prueba de aceptación	
Código de caso de prueba: HU1_CP1.	Nombre de la historia de usuario: Obtener lectores disponibles.
Responsable de la prueba: Amanda Cabrera Miralles.	
Descripción de la prueba: Prueba de funcionalidad para comprobar la conexión con el lector	
Condiciones de ejecución: Debe existir al menos un lector disponible	
Entrada/Pasos de ejecución: Seleccionar un lector, en caso que exista más de uno, se escoge el primero o se puede elegir el que se desea.	
Resultado esperado: Queda seleccionado el lector para la comunicación.	
Evaluación de la prueba: Prueba Satisfactoria.	

Tabla 18 HU2_CP2_Inicializar conexión.

Caso de prueba de aceptación	
Código de caso de prueba: HU2_CP2	Nombre de la historia de usuario: Inicializar conexión.

Responsable de la prueba: Fernando José Enríquez Fernández
Descripción de la prueba: Prueba de funcionalidad para comprobar la conexión.
Condiciones de ejecución: Debe existir un lector y una tarjeta, conectados y la conexión de la aplicación tiene que estar abierta.
Entrada/Pasos de ejecución: Conectar la tarjeta.
Resultado esperado: Queda establecida la conexión con la tarjeta
Evaluación de la prueba: Prueba Satisfactoria.

Tabla 19 HU3_CP3_ Enviar comandos APDU de la tarjeta.

Caso de prueba de aceptación	
Código de caso de prueba: HU3_CP3	Nombre de la historia de usuario: Enviar comandos APDU de la tarjeta.
Responsable de la prueba: Amanda Cabrera Miralles.	
Descripción de la prueba: Prueba de funcionalidad para verificar el envió de APDU a la tarjeta.	
Condiciones de ejecución: Debe existir un lector y una tarjeta, conectados.	
Entrada/Pasos de ejecución: Se realiza una operación en la aplicación.	
Resultado esperado: Se espera respuesta satisfactoria de dicha operación.	
Evaluación de la prueba: Prueba Satisfactoria.	

Tabla 20 HU4_CP4_Establecer canal seguro.

Caso de prueba de aceptación	
Código de caso de prueba: HU4_CP4	Nombre de la historia de usuario: Establecer canal seguro.
Responsable de la prueba: Fernando José Enríquez Fernández	
Descripción de la prueba: Prueba de funcionalidad para comprobar el intercambio de información de forma segura entre el *Middleware* y el *Applet*.	
Condiciones de ejecución: Debe existir un canal de intercambio de información segura.	
Entrada/Pasos de ejecución: Realizar operaciones en la aplicación.	
Resultado esperado: Queda establecido el canal seguro.	
Evaluación de la prueba: Prueba Satisfactoria.	

Tabla 21 HU5_CP5_ Verificar PIN.

Caso de prueba de aceptación	
Código de caso de prueba: HU5_CP5	Nombre de la historia de usuario: Verificar PIN.
Responsable de la prueba: Amanda Cabrera Miralles.	
Descripción de la prueba: Prueba de funcionalidad para realizar la verificación del PIN.	
Condiciones de ejecución: Debe existir un lector y una tarjeta, conectados.	
Entrada/Pasos de ejecución: Introducir PIN, dar clic en botón: Autenticarse.	
Resultado esperado: Se muestra un mensaje para informar que la autenticación ha sido satisfactoria.	
Evaluación de la prueba: Prueba Satisfactoria.	

Tabla 22 HU7_CP7_Generar par de llaves.

Caso de prueba de aceptación	
Código de caso de prueba: HU7_CP7	Nombre de la historia de usuario: Generar par de llaves.
Responsable de la prueba: Amanda Cabrera Miralles.	
Descripción de la prueba: Prueba de funcionalidad para verificar la generación de llaves.	
Condiciones de ejecución: Debe existir un lector y una tarjeta conectados a la aplicación, además debe estar autenticado el usuario correctamente.	
Entrada/Pasos de ejecución: Se realiza una operación en la aplicación.	
Resultado esperado: Se espera respuesta satisfactoria de dicha operación.	
Evaluación de la prueba: Prueba Satisfactoria	

Tabla 23 HU9_CP9_Realizar firma.

Caso de prueba de aceptación	
Código de caso de prueba: HU9_CP9	Nombre de la historia de usuario: Realizar firma.
Responsable de la prueba: Amanda Cabrera Miralles.	
Descripción de la prueba: Prueba de funcionalidad para verificar realización de firma.	

Condiciones de ejecución: Debe existir un lector y una tarjeta conectados a la aplicación, además debe estar autenticado el usuario correctamente.
Entrada/Pasos de ejecución: Se realiza una operación en la aplicación.
Resultado esperado: Se espera respuesta satisfactoria de dicha operación.
Evaluación de la prueba: Prueba Satisfactoria

Tabla 24 HU12_CP_ Realizar desconexión.

Caso de prueba de aceptación	
Código de caso de prueba: HU12_CP12	Nombre de la historia de usuario: Realizar desconexión.
Responsable de la prueba: Fernando José Enríquez Fernández	
Descripción de la prueba: Prueba de funcionalidad para comprobar la desconexión con la tarjeta.	
Condiciones de ejecución: Debe existir un lector y una tarjeta conectados a la aplicación y tiene que estar iniciada la conexión.	
Entrada/Pasos de ejecución: desconectar la tarjeta.	
Resultado esperado: Queda cerrada la conexión.	
Evaluación de la prueba: Prueba Satisfactoria.	

3.4.2 Pruebas Unitarias

Las pruebas de unidad se crean por los programadores antes de empezar a codificar lo cual hará más sencillas y efectivas las pruebas finales. Se corren reiteradamente a lo largo de todo el proyecto, asegurando siempre el funcionamiento correcto de cada componente por individual antes de realizar su integración. Evitan las ambigüedades y los requerimientos quedan afinados en la prueba. Una funcionalidad está terminada cuando pasa todas sus pruebas de unidad.

Método de Caja Blanca - Camino Básico

El método de caja blanca se basa en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y/o bucles.

Para comprobar los resultados de la solución se utilizó el método del camino básico, propuesto por Tom McCabe, permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia de programa.

La complejidad se puede calcular de tres formas:

1.- El número de regiones del grafo de flujo

2.- Aristas - Nodos + 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)

El valor de V(G) nos 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.

Caso de prueba: HU_4 Establecer Canal Seguro.

```
void SCReaderWrapper::OpenSecureChannel()
{
 // Obteniendo CardSerialNumber
 ByteArray* tag = new ByteArray(2, 0x01); (1)
 GetDataCommand* getData0101Command = new GetDataCommand(GlobalPlatform,
tag); (1)
 ResponseAPDU* getData0101Response = new ResponseAPDU(); (1)
 transmit(*getData0101Command,*getData0101Response); (2)
 VerifyResponse(getData0101Response); (3)
 //CSN
 ByteArray* cardSerialNumber = util->SubByteArray(getData0101Response-
>getData(), 3, 8); (4)
 // Obteniendo ChipSerialNumber
 ByteArray* tag1 = new ByteArray(); (4)
 tag1->push_back(0x9F); (4)
 tag1->push_back(0x7F); (4)
 GetDataCommand* getData9F7FCommand = new GetDataCommand(GPlatform, tag1);
(4)
 ResponseAPDU* getData9F7Response = new ResponseAPDU();(4)
 transmit(*getData9F7FCommand,*getData9F7Response);(5)
 VerifyResponse(getData9F7Response);(6)
 //SN.ICC
 std::vector<byte>* chipSerialNumber= util->SubByteArray(getData9F7Response-
>getData(),13,8);(7)
 if(!AplicationSelect)(8)
 selectApplication();(9)
 if(!_secureChannnel) (10)
 InitializeSecureChannel();(11)
 //efectuar la autenticación mutua simétrica, en la fase de aplicación.
```

```
 _secureChannnel-
>MutualAuthentication(cardSerialNumber,chipSerialNumber);(12)
}(13)
```

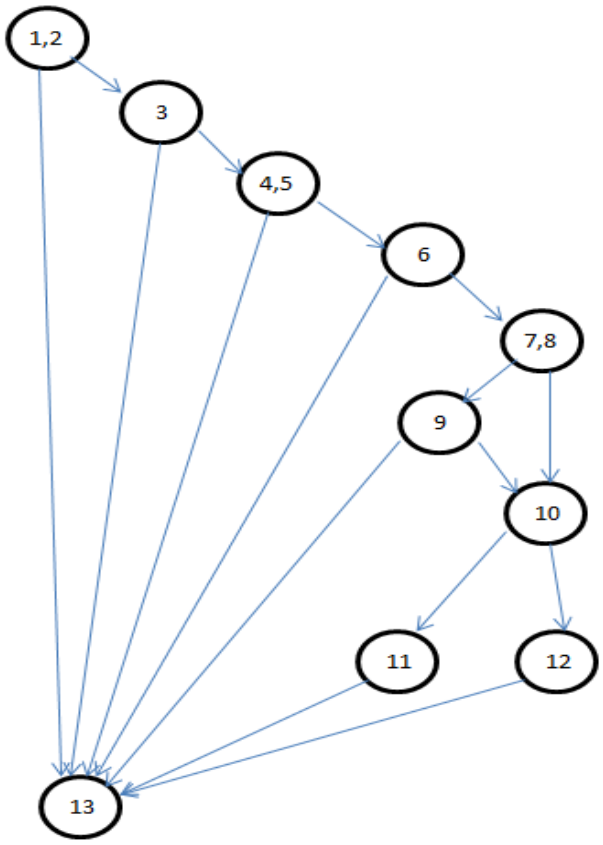

Figura 8 Grafo de flujo HU_4 Establecer Canal Seguro

Complejidad ciclomática.

V (G): *Número de regiones del grafo.*

V (G) = A – N + 2

V (G) = P + 1

A: Número de aristas del grafo.

N: Número *de n*odos.

P: Número de nodos predicados.

A= 16

N= 10

P= 7

V (G) = 8

Caminos:

1, 2, 13

1, 2, 3, 13

1, 2, 3, 4, 5, 13

1, 2, 3, 4, 5, 6, 13

1, 2, 3, 4, 5, 6, 7, 8, 9, 13

1, 2, 3, 4, 5, 6, 7, 8, 9,10, 11, 13

1, 2, 3, 4, 5, 6, 7, 8, 9,10, 12, 13

Casos de prueba para cada camino**.**

Camino: 1, 2,13.

Entrada: Datos para la autenticación mutua.

Salida: Excepción especificando que no se puede enviar el comando APDU y por qué.

Precondiciones: Ninguna.

Camino: 1, 2, 3, 13.

Entrada: Datos para la autenticación mutua.

Salida: Excepción especificando un error en la respuesta APDU de la tarjeta.

Precondiciones: Ninguna.

Camino: 1, 2, 3, 4, 5, 13.

Entrada: Datos para la autenticación mutua.

Salida: Excepción especificando que no se puede enviar el comando APDU y por qué.

Precondiciones: Ninguna

Camino: 1, 2, 3, 4, 5, 6, 13.

Entrada: Datos para la autenticación mutua.

Salida: Excepción especificando un error en la respuesta APDU de la tarjeta.

Precondiciones: Ninguna.

Camino: 1, 2, 3, 4, 5, 6, 7, 8, 9, 13.

Entrada: Datos para la autenticación mutua.

Salida: Excepción especificando un error a la hora de seleccionar la aplicación.

Precondiciones: Ninguna.

Camino: 1, 2, 3, 4, 5, 6, 7, 8, 9,10, 11, 13.

Entrada: Datos para la autenticación mutua.

Salida: Excepción especificando un error en el hora de inicializar el canal seguro.

Precondiciones: Ninguna.

Precondiciones: Ninguna.

Camino: 1, 2, 3, 4, 5, 6, 7, 8, 9,10, 12, 13.

Entrada: Datos para la autenticación mutua.

Salida: Excepción especificando un error en el hora de efectuar la autenticación mutua simétrica.

Precondiciones: Ninguna.

3.4.3 Análisis de los Resultados de las Pruebas

Tras la ejecución de las diferentes pruebas de software se alcanzaron los resultados que se recogen en la gráfica de la figura 9.

Para un total de 60 pruebas realizadas durante todo el ciclo del proyecto, dividido en 4 iteraciones, se detectaron de 70 no conformidades. Durante la primera iteración se realizaron 20 pruebas de unidad y 15 pruebas de aceptación, detectándose 27 no conformidades a las cuales se le dieron solución. En la segunda iteración se realizaron 10 pruebas de unidad y 8 de aceptación, mostrándose 22 no conformidades las mismas fueron resueltas en su totalidad. En la tercera iteración se realizaron 8 pruebas de unidad y 5 de aceptación, detectándose 13 no conformidades. En la última iteración se realizaron 3 pruebas de unidad y 3 de aceptación, detectándose 8 no conformidades. En las cuatro iteraciones efectuadas las no conformidades detectadas, en su mayoría respondían a errores de bajo impacto en el correcto funcionamiento de la aplicación y todas tuvieron solución un tiempo máximo de 6 días. Posteriormente se realizó una iteración adicional para verificar que no existieran no conformidades, la misma arrojó un resultado de cero no conformidades.

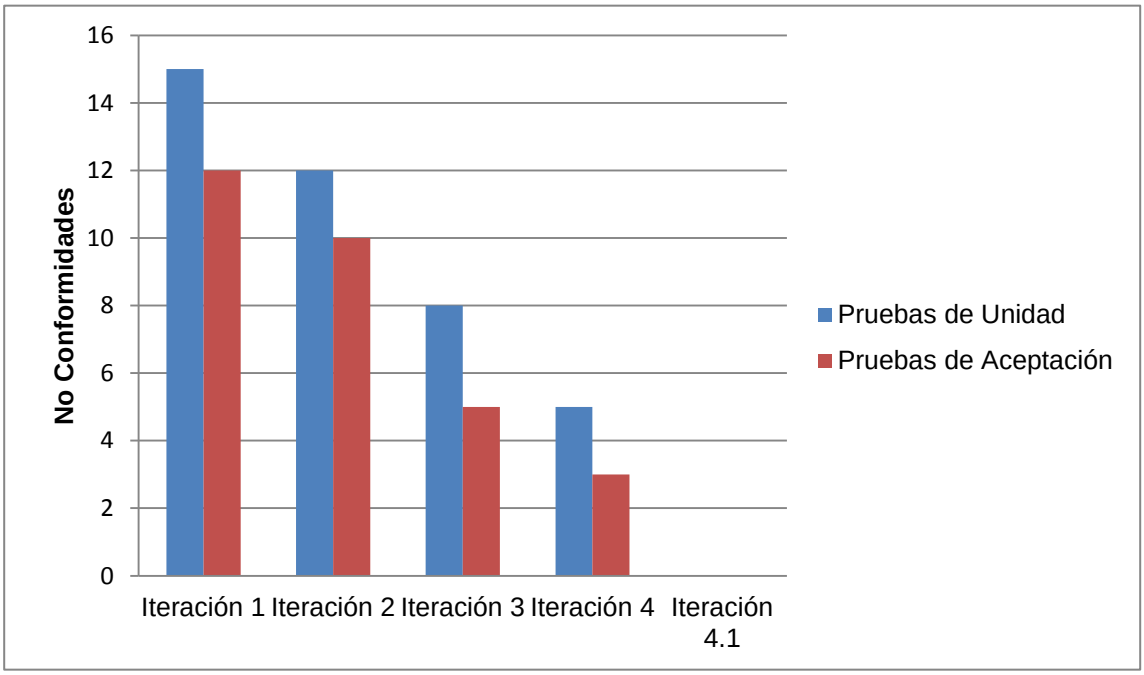

Figura 9 Resultados de las pruebas

3.5 Conclusiones Parciales

- Con el diseño del diagrama de despliegue se mostró las distribución física que tendría la solución, brindando una distribución completa del acople de los distintos componentes de hardware y software del sistema.
- Con la distribución de los paquetes y componentes a partir del diagrama de componentes los mismos proporcionan y utilizan a través de las interfaz de la aplicación cliente que se implementó.
- Las pruebas unitarias posibilitaron evaluar las respuestas de funcionalidades implementadas, en cada una de las iteraciones realizadas. Las pruebas realizadas al sistema tuvieron un resultado satisfactorio dando así cumplimiento a los requisitos propuestos.

CONCLUSIONES GENERALES

Una vez completado el trabajo de diploma “Middleware para el *applet* PKI “uSafeApp””, se concluye que:

- El análisis de los principales sistemas que implementan aplicaciones *middleware* para entornos PKI en tarjetas inteligentes, ha posibilitado una mejor compresión del problema.
- El análisis de diferentes herramientas, tecnologías, lenguajes y metodologías ha permitido definir cuáles utilizar en el desarrollo del sistema.
- La especificación de los requisitos que debe cumplir el sistema han posibilitado aplicar el patrón arquitectónico n-capas para describir la estructura general del sistema.
- La aplicación de patrones de diseño ha posibilitado una mejor organización y un mejor rendimiento durante la implementación del sistema.
- El desarrollo del *middleware* “uSafeMdw” ha permitido hacer uso del *applet* PKI “uSafeApp” realizando operaciones de firma digital y autenticación, cumpliendo con los estándares internacionales correspondientes para este tipo dispositivo en una PKI.
- La ejecución de las pruebas han posibilitado validar el correcto funcionamiento del sistema.

RECOMENDACIONES

Tras haber finalizado el desarrollo del *middleware* “uSafeMdw” en su primera versión se recomienda:

- El *applet* “uSafeApp” pueda soportar otros algoritmos de criptografía asimétrica, específicamente algoritmos basados en curvas elípticas, para así aumentar sus posibilidades de uso en ambientes reales en la actualidad.
- Integrar el middleware con otras herramientas de firma que se desarrolló en el departamento de seguridad digital para firmar pdfs y otros formatos de documentos con el objetivo de obtener un producto de firma digital totalmente desarrollado en la UCI e integrado con tarjetas inteligentes Java Card.

REFERENCIAS BIBLIOGRÁFICAS

1. Damien Deville, A.G., Gilled Grimaud, Smart Card Operating System Past, Present and Future. Febrero 2003.

2. Lucena López, M.J., Criptografía y Seguridad en Computadores. 1999.

3. Farlex, T.t.D.b., Criptografía in Diccionario Manual de la Lengua Española Vox. © 2007 Larousse Editorial, S.L. Diccionario Enciclopédico Vox 1. © 2009 Larousse Editorial, S.L. 2013.

4. Effing, W.a.R., Wolfgang, Smart Card Handbook Third Edition: John Wiley & Sons Ltd. 2002.

5. S.A., C.D., Certificado Digital. . Version 1.0, 2010.

6. Tecnysis, SEGURIDAD INFORMÁTICA. 2010.

7. Farlex, T.f.D.b., Middleware, in The American Heritage® Dictionary of the English Language, Fourth Edition copyright ©2000 by Houghton Mifflin Company. Updated in 2009. Published by Houghton Mifflin Company. All rights reserved. 2009.

8. John Wiley, S.L., Smart Card Handbook. Third Edition ed. 2002.

9. webiec. ISO/IEC 2007INTERNATIONAL STANDARD ISO/IEC 7816-2, Identification cards, Integrated circuit cards, Part 2: Cards with contacts, Dimensions and location of the contacts. 2007; Available from: http://webstore.iec.ch/preview/info_isoiec7816-2%7Bed2.0%7Den.pdf.

10. Microsystems, S., Java Card 2.1.1 Runtime Environment (JCRE) Specification. 2000, Sun Microsystems, Inc.: 901 San Antonio Road Palo Alto, CA 94303 USA.

11. Microsystem, S., Conozca más sobre la tecnología Java. 2012.

12. Tanenbaum, A., S .Computer Network Tanenbaum. 4 ed ed.

13. Tanenbaum, A., Computer Networks 4ta edición ed. 2003: Pearson Education.

14. Perovich, D., L. Rodríguez, and M. Varela, Proyecto de Taller V (2000) Programación de JavaCards, in Instituto de Computación Facultad de Ingeniería. 2001, Universidad de la República.

15. GlobalPlatform, Card Specification Versión 2.1.1. 2003.

16. Laboratories, R., PKCS #15 v1.0: Cryptographic Token Information Format Standard. 1999.

17. Yoan Crespo, K.R., Aplicación para tarjetas inteligentes en una Infraestructura de Clave Pública, Universidad de las Ciencias Informáticas. 2012.

18. Laboratories, R., PKCS #11 v1.0: Cryptographic Token Interface Standard. 1999.

19. Alliance, S. About SmartCard. 2013; Available from: http://www.smartcardalliance.org/.

20. Gemalto. IDClassic 300. 2013; Available from: http://www.gemalto.com/products/classic_tpc/.

21. Gemalto. IDPrime .NET. 2012; Available from: http://www.gemalto.com/products/dotnet_card/.

22. ARX. 2013; Available from: http://www.arx.com.

23. Alliance, A. Metodologia XP. Fases. Domingo, 11 Diciembre 2011 16:31 Available from: http://metodologiasagiles.herobo.com/index.php/es/2011-12-05-16-09-55/metodologia-xp/planificacion.

24. Batalla, L., et al., Extreme Programming (XP). Gestión de Software, 2006.

25. Ivar Jacobson, J.R., Grady Booch, El lenguaje unificado de modelado. Manual de referencia. 2000.

26. Softpedia. Altova UModel Enterprise Edition 2010 2010; Available from:

http://www.softpedia.com/get/Programming/Other-Programming-Files/Altova-UModel.shtml.

27. http://www.altova.com/umodel.htm. ALTOVA UModel. 2010.

28. WikiLibros. Programación con Qt4. 2010 Modificada por última vez el 17 sep 2010, a las 21:43.

29. OrganiZATOR. Curso C++. 2010; Available from: http://www.zator.com/Cpp/E1_2.htm.

30. WikiLibros, Programación en C++. Capítulo 1 2013, Modificada por última vez el 15 may 2013, a las 16:42.

31. Software, Flujo de trabajo Captura de requisitos. Modelo de Negocio, D.C.d.I.d. Software, Editor. 2004: Ciudad de la Habana, Universidad de Ciencias Informáticas.

32. Beck, K., “Extreme Programming Explained. Embrace Change”. 1999: Pearson Education.

33. Zapata, M.A., Diagrama de clases, in 3.- Diseño estructural. 2006, Máster Bases de Datos e Internet.

34. Caamal, R., Diagrama de clases. Material académico, 2012.

35. Larman, C., UML y Patrones Introducción al análisis y diseño orientado a objetos. 2002: Prentice-Hall.

36. Kicillof, C.R.N., Estilos y Patrones en la Estrategia de Arquitectura de Microsoft, in Versión 1.0. 2004, UNIVERSIDAD DE BUENOS AIRES: Argentina.

37. Msdn. Diagramas de componentes de UML: Referencia. 2012.

38. Alva, E.R., Diagramas de Componentes y Despliegue. Arquitectura de Software II, 2008.

BIBLIOGRAFÍA CONSULTADA

- Altova, Inc. 2010. ALTOVA. [En línea] 2010. http://www.altova.com/umodel.htm.
- ARX. 2012. ARX. [En línea] 2012. http://www.arx.com.
- Biznestehnologia Ltd. 2007.7.CORBA. Millennium ERP: a competitive ERP solution for the new millennium / CORBA (eng).UralWES, 2007. http://www.millennium-group.ru/index/lang/eng/parent_id/30/level/1.
- BlueZ. PKCS#15 an implementation for Open Platform Java Cards.IBM Corporation. 2003. 2003, Vol. Rev2.01.
- GARCERANT. Modelo de Dominio. [En línea] 2010. Disponible en: http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.
- KARL.Middleware.2011.Disponible en: http://www.buenastareas.com/ensayos/Middleware/2031177.html.

GLOSARIO DE TÉRMINOS

APDU: *Unidades de Datos en la Capa de Aplicación.*

API: *Programación de Aplicaciones de Interfaz* es un lenguaje y un formato de mensaje usado por un programa de aplicación para comunicarse con el sistema operativo o algún otro programa de control.

Applets: son aplicaciones implementadas utilizando la tecnología Java Card y son ejecutadas dentro de las tarjetas inteligentes.

ISO: *Organización Internacional para la Normalización* es el organismo encargado de promover el desarrollo de normas internacionales de fabricación, comercio y comunicación para todas las ramas industriales a excepción de la eléctrica y la electrónica.

Java Card: es una tecnología que permite ejecutar de forma segura pequeñas aplicaciones Java (Applets) en tarjetas inteligentes y similares dispositivos empotrados.

Tarjeta Inteligente: tarjeta que contiene un pequeño microprocesador, que es capaz de hacer diferentes cálculos, guardar información y manejar programas, que están protegidos a través de mecanismos avanzados de seguridad.

.

ANEXOS

Anexo 1: Estimación de Esfuerzo

Tabla 25 Estimación de Esfuerzo.

No	Historia de Usuario	Tiempo Estimado		
		Semanas Estimadas	**Días Estimados**	**Horas Estimadas**
1	Obtener lectores disponibles.	1	5	40
2	Inicializar conexión.	1	5	40
3	Enviar comandos APDU de la tarjeta.	1.5	5.5	44
4	Establecer canal seguro entre el *middleware* "uSafeMdw" y el *applet* "uSafeApp".	2.5	10.5	84
5	Verificar PIN.	2.5	10.5	84
6	Cambiar PIN.	2.5	10.5	84
7	Generar par de llaves.	1.5	5.5	44
8	Obtener par de llaves.	1.5	5.5	44
9	Realizar firma.	2.5	10.5	84
10	Obtener certificados digitales.	2.5	10.5	84
11	Cargar certificados digitales.	1.8	6	48
12	Realizar desconexión.	1	5	40
	Total	**21.8**	**90**	**720**

Anexo 2: Plan de Entregas

Tabla 26 Plan de Entregas.

No	Historia de Usuario	Semanas Estimadas	Iteración Asignada	
			1	2
1	Obtener lectores disponibles.	1	x	
2	Inicializar conexión.	1	x	
3	Enviar comandos APDU de la tarjeta.	1.5	x	
4	Establecer canal seguro entre el *middleware* "uSafeMdw" y el *applet* "uSafeApp".	2.5	x	
5	Verificar PIN.	2.5	x	
6	Cambiar PIN.	2.5	x	
7	Generar par de llaves.	1.5		x
8	Obtener par de llaves.	1.5		x
9	Realizar firma.	2.5		x
10	Obtener certificados digitales.	2.5		x
11	Cargar certificados digitales.	1.8		x
12	Realizar desconexión.	1		x

Anexo 3: Plan de Iteraciones

Tabla 27 Plan de Iteraciones.

Iteraciones	No HU	Nombre de Historia de Usuario	Duración (semanas)	Inicio	Fin
1	1	Obtener lectores disponibles.	8.5	Noviembre 2012	Febrero 2013
	2	Inicializar conexión.			
	3	Enviar comandos APDU de la tarjeta.			
	4	Establecer canal seguro entre el *middleware* “uSafeMdw” y el *applet* “uSafeApp”.			
	5	Verificar PIN.			
	6	Cambiar PIN.			
2	7	Generar par de llaves.	10.6	Marzo 2013	Mayo 2013
	8	Obtener par de llaves.			
	9	Realizar firma.			
	10	Obtener certificados digitales.			
	11	Obtener lectores disponibles.			
	12	Realizar desconexión.			

Anexo 4: Tareas de Ingeniería

Tabla 28 Tareas de Ingeniería.

Iteración	Historia de Usuario	Tarea
1	Obtener lectores disponibles.	- Detectar lector conectado. - Mostrar lectores disponibles. - Seleccionar lector. - Elaborar prototipo de interfaz. - Diseño de la tarjeta CRC SCReaderWrapper. - Implantación método ListReaders() del componente *middleware* “uSafeMdw”. - Documentación del caso de prueba Inicializar conexión.
	Inicializar conexión.	- Seleccionar *applet “*uSafeApp*”.* - Diseño de la tarjeta CRC SCReaderWrapper. - Implantación método Connect() del componente *middleware* “uSafeMdw”. - Realización de prueba de unidad al método Connect(). - Documentación del caso de prueba Inicializar conexión
	Enviar comando APDU a la tarjeta.	- Establecer comunicación con la tarjeta inteligente - Diseño de la tarjeta CRC SCReaderWrapper. - Diseño de la tarjeta CRC SmartCardUtil. - Diseño de la tarjeta CRC APDU. - Implantación método trasmit() del componente *middleware* “uSafeMdw”. - Realización de prueba de unidad al método trasmit().
	Establecer canal seguro.	- Establecer canal seguro. - Encriptar la información que se envía de la tarjeta al *applet* y viceversa. - Diseño de la tarjeta CRC SCReaderWrapper. - Diseño de la tarjeta CRC SmartCardUtil. - Diseño de la tarjeta CRC SmartCardUtil. - Implantación método verifyResponse() del middleware “uSafeMdw”. - Realización de prueba de unidad al método verifyResponse(). - Documentación del caso de prueba Inicializar conexión

	Verificar identidad mediante PIN.	- Obtener los datos introducidos por el usuario. - Realizar verificación de PIN. - Elaborar prototipo de interfaz. - Diseño de la tarjeta CRC SCReaderWrapper. - Diseño de la tarjeta CRC SmartCardUtil. - Diseño de la tarjeta CRC SmartCardUtil. - Implantación método verifyResponse()del middleware “uSafeMdw”. - Realización de prueba de unidad al verifyResponse (). - Documentación del caso de prueba Inicializar conexión.
	Cambiar PIN.	- Modificar ID de usuario. - Elaborar prototipo de interfaz. - Diseño de la tarjeta CRC SCReaderWrapper. - Diseño de la tarjeta CRC SmartCardUtil. - Diseño de la tarjeta CRC SmartCardUtil. - Implantación método OpenSecureChannel() del componente *middleware* “uSafeMdw” - Realización de prueba de unidad al métodoOpenSecureChannel(). - Documentación del caso de prueba Cambiar PIN.
2	Generar par de llaves.	- Elaborar prototipo de interfaz. - Diseño de la tarjeta CRC SCReaderWrapper. - Diseño de la tarjeta CRC SmartCardUtil. - Diseño de la tarjeta CRC SmartCardUtil. - Implantación método MutualAuthenticate() del componente *middleware* “uSafeMdw”. - Realización de prueba de unidad al método MutualAuthenticate(). - Documentación del caso de prueba Generar par de llaves.
	Obtener par de llaves.	- Implantación método VerifyPin()del componente *middleware* “uSafeMdw”. - Realización de prueba de unidad al método VerifyPin(). - Documentación del caso de prueba Generar par de llaves.
	Realizar par de firma.	- Elaborar prototipo de interfaz

		- Permitir cargar documentos. - Permitir Firma de documentos. - Diseño de la tarjeta CRC SCReaderWrapper. - Diseño de la tarjeta CRC SmartCardUtil. - Diseño de la tarjeta CRC SmartCardUtil. - Implantación método privatekeygenerator() del componente *middleware* "uSafeMdw". - Realización de prueba de unidad al métodoprivatekeygenerator().
	Obtener certificados digitales.	- Obtener información del *applet "*uSafeApp*".* - Elaborar prototipo de interfaz. - Diseño de la tarjeta CRC SCReaderWrapper. - Diseño de la tarjeta CRC SmartCardUtil. - Diseño de la tarjeta CRC SmartCardUtil.
	Obtener lectores disponibles.	- Elaborar prototipo de interfaz - Listar Lectores. - Diseño de la tarjeta CRC SCReaderWrapper. - Diseño de la tarjeta CRC SmartCardUtil. - Diseño de la tarjeta CRC SmartCardUtil.
	Realizar desconexión.	- Desconectar la tarjeta del lector.