Universidad de las Ciencias Informáticas

Diseño de un Applet y Middleware para gestionar la información biométrica contenida en la Cédula de Identificación Electrónica de la República Bolivariana de Venezuela.

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

Autora:
Yarisel Aguilera Nieblas

Tutor: Ing. Dayron Almeida Sotolongo
Co-Tutor: Ing. Michel Rafael Pérez Costa
Consultante: Msc. Adonis Cesar Legón Campo

La Habana
Junio 2011

"El aspecto fundamental en el cual la juventud debe señalar el camino es precisamente en el aspecto de ser vanguardia en cada uno de los trabajos que le compete."

DECLARACIÓN DE AUTORÍA Declaro que soy la única autora del trabajo titulado: *"Diseño de un Applet y Middleware para gestionar la información biométrica contenida en la Cédula de Identificación Electrónica de la República Bolivariana de Venezuela"* y autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ _____

Nombre del estudiante 1 Nombre del estudiante 2

_____

Nombre del tutor

AGRADECIMIENTOS

A la Revolución por la posibilidad de estudiar y realizarnos como profesionales. A todos los profesores que de una forma u otra influyeron en mi preparación. Al tutor, por su paciencia y dedicación. A los miembros del tribunal y el oponente que influyeron en gran medida en el perfeccionamiento de esta investigación. A mi mamita que siempre ha estado al tanto de mí, brindándome su ayuda y amor, a mi papá que sé que se siente orgulloso. A mi hermanita a quien quiero mucho y a mis sobrinos para quienes espero ser un ejemplo. A mi esposo Arley, gracias por todo su apoyo, paciencia y amor lo que me ha motivado a superarme cada vez más. A mi tía Maricela que me ha acogido en su hogar desde los comienzos de mis estudios en la UCI haciéndome sentir como en mi propia casa.

DEDICATORIA

Primero a mi mayor tesoro, mi esposo Arley, quien con amor y paciencia me ha ayudado todos estos años a ser mejor, siempre será parte importante de mi vida. A mi familia: mis padres, quienes tienen todo el mérito por haberme enseñado desde pequeña a apreciar el conocimiento. A mis tíos; a mis primos y mis abuelos que siempre me han brindado su amor. A mi hermanita a quien siempre he querido como mi segunda mamá y de quien siempre he estado muy orgullosa. A mis amigas, que me han apoyado y que siempre me traerán gratos recuerdos; en especial a Lari, Lora, Yeni, Yanet, Carmen y Mare.

RESUMEN

El objetivo fundamental de esta Tesis se centra en la gestión de la información biométrica de las huellas dactilares en la Cédula de Identificación Electrónica de la República Bolivariana de Venezuela. La motivación de dicho trabajo ha sido la necesidad de una aplicación que al gestionar la información biométrica de las huellas dactilares establezca canal seguro para la transmisión de los datos y los mismos viajen cifrados.

Para lograr este objetivo se realizó un estudio de las principales tecnologías utilizadas en el desarrollo de aplicaciones que se ejecutan en las tarjetas inteligentes, se analizaron soluciones similares y los estándares a los cuales se debe ajustar además se describen las herramientas y los artefactos generados en el proceso de desarrollo.

Obteniéndose el análisis y diseño de una aplicación Applet y su componente middleware con el propósito de vencer las limitaciones existentes en cuanto a la seguridad en el proceso de implantación de la Cédula de Identificación Electrónica.

Palabras Claves

Tarjetas Inteligentes, Applet, Middleware, Cédula de Identificación Electrónica, Estándares.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Desde el comienzo de la civilización, identificar a las personas ha sido crucial para la sociedad, por ello se hizo imprescindible la utilización de mecanismos que permitiesen identificar a cada individuo, los cuales consistían en almacenar datos propios de cada persona, permitiendo así diferenciar unos de otros. Estos mecanismos no se mantuvieron ajenos a los avances que fueron sucediendo con el transcurso de los años, haciéndose cada vez más necesario la aparición de un documento que contara con información más específica de las personas, de ahí que surgiera lo que se conoce hoy como carnet de identidad.

El carné de identidad, oficialmente y según la legislación Documento Nacional de Identidad (DNI) o cédula de identidad, llamada así en países como: Chile, Costa Rica, Bolivia, Venezuela, entre otros, es un documento emitido por una autoridad administrativa competente para permitir la identificación personal de los ciudadanos, ya sea ante las autoridades de orden público, en las votaciones electorales, entre otros.

No todos los países emiten documentos de identidad, ejemplo de ello son los países que poseen un sistema jurídico basado en el derecho anglosajón, tales como Gales e Irlanda; aunque la extensión de la práctica acompañó el establecimiento de sistemas nacionales de registro de la población y la elaboración de los medios de control administrativo del estado.

La República Bolivariana de Venezuela era un país de ciudadanos excluidos que no contaban en los registros y las estadísticas oficiales, hasta que la Constitución de la República Bolivariana de Venezuela (2007), estableció en su Artículo 56 que "toda persona tiene derecho a un nombre propio, al apellido del padre y al de la madre, y a conocer la identidad de los mismos... ". (SAIME, 2011)

Para ello se hacía necesario que el gobierno venezolano diera los primeros pasos y rompiera con las viejas costumbres implantadas por él mismo, trazada esta meta, para el año 2004 nació la *Misión Identidad* para cumplir el mandato de la Constitución y permitir el derecho a la existencia de miles de venezolanos olvidados. De esta forma y por tratarse de un documento de vital importancia para cualquier ciudadano el gobierno nacional inició un plan de cedulación a lo largo de todo el país, con el objeto de prestar un mejor servicio a todos los venezolanos, a través de operativos móviles distribuidos en todo el país reforzando el trabajo realizado por la institución.

Venezuela fue el primer país latinoamericano en expedir un documento de identidad nacional. A finales de 1944 la primera cédula de identidad emitida con el número uno fue entregada al presidente Isaías Medina Angarita.

El documento de identidad venezolano es procesado actualmente por el *Servicio Administrativo de identificación, Migración y Extranjería* (SAIME) (antes *Oficina Nacional de Identificación y Extranjería* (ONIDEX), primeramente DIEX), organismo dependiente del Ministerio del Poder Popular para Relaciones Interiores y Justicia.

La cédula venezolana, a diferencia de otras en Latinoamérica, ha descendido en su calidad en los últimos años. Al iniciarse la *Misión Identidad*, se crea entonces un sistema automatizado en donde estos documentos son impresos en computadoras en menos de cinco minutos, haciendo grandes avances en el proceso de cedulación para toda la población pero reduciendo significativamente la seguridad del mismo.

Con la constante evolución del mundo tecnológico y con vistas a tener una mayor seguridad en cuanto a identificación de cada ciudadano, el gobierno venezolano consideró la utilización de tarjetas inteligentes como documento de identificación (cédula de identificación electrónica), debido a que las mismas son dispositivos activos de almacenamiento de información, que incorporan mecanismos para proteger el acceso a la información mediante claves y algoritmos de cifrado. Además, como las tarjetas inteligentes están basadas en un microprocesador y un sistema operativo que controla todo el flujo de información, cabría la posibilidad de incorporar a este tipo de tarjetas nuevas funcionalidades, brindando así nuevas posibilidades a las diferentes instituciones del país. (Agora SIC, 2000). Al ser Venezuela uno de los primeros países de América Latina en utilizar tarjetas inteligentes en la identificación de sus ciudadanos y para brindar algunos de los servicios que se prestan a las personas por parte de varias entidades, se hace necesario tener mecanismos de seguridad para verificar la validez de posesión del portador de la Cédula de Identificación Electrónica.

La identificación de las personas es un proceso de vital importancia para toda sociedad, ya que existen muchos aspectos como la seguridad, prestación de servicios, control de acceso, comercio electrónico, banca, servicio social y muchos otros que requieren la autenticación de las personas. Como mecanismo de control de acceso, las tarjetas inteligentes hacen que los datos personales y de negocios solo sean accesibles a los usuarios apropiados, asegurando la portabilidad, seguridad y confiabilidad en los mismos.

En la actualidad debido a la continua proliferación de nuevas aplicaciones donde es necesaria una autenticación[1] del usuario que las utiliza y, además, las nuevas posibilidades que se abren dentro de la tecnología de dicho tipo de tarjetas de identificación, se hace necesario que el nuevo sistema de

[1] Acto de establecimiento o confirmación de algo (o alguien) como auténtico, es decir probar que es verdadero.

autenticación esté basado en la identificación biométrica[2], de forma tal que se realice la comprobación de la identidad del titular dentro de la propia tarjeta. La identificación biométrica es esencial, puesto que brinda beneficios, pues las características biológicas no pueden ser prestadas o robadas, e intrínsecamente estas características representan la identidad completa del individuo, es decir se reconoce a una persona por su cuerpo y el mismo es enlazado a una identidad externa establecida.

Es por ello que la identificación biométrica se utiliza cada vez más para complementar o sustituir los esquemas de autentificación tradicionales tales como los números de identificación personal (PIN) o las claves. De las técnicas de identificación biométrica existentes, las huellas dactilares representan legalmente una de las más utilizadas como prueba fidedigna de identidad, siendo un sistema que además de ser efectivo, es cómodo de aplicar y la autenticación se obtiene rápidamente. (Agora SIC, 2000) De aquí que la técnica de biometría adoptada por el presente trabajo de diploma sea la verificación biométrica por huellas dactilares.

De esta forma, para un sistema de autenticación biométrica, una tarjeta inteligente se podría utilizar como:

• Dispositivo seguro que almacena la identidad del usuario, así como su patrón, sólo dejando leer el patrón por parte del terminal que tiene permiso para ello. Además, la tarjeta, para garantizar la confidencialidad del patrón, puede transmitir el patrón cifrado mediante una clave de sesión, incrementando, por tanto, la seguridad del sistema.

• Además de lo anterior, cabría la posibilidad de pensar en utilizar la Verificación Biométrica como otro sistema más de seguridad dentro de la Tarjeta Inteligente (tal y como se usa, por ejemplo, el PIN). De esta forma, se podría proteger de forma biométrica, no sólo la información almacenada dentro de la tarjeta, sino también determinadas operaciones como, por ejemplo, el débito de un monedero electrónico.(Agora SIC, 2000)

El CryptoManager Applet es una solución que resolvería parcialmente los problemas para el proceso de implantación de la cédula de identificación electrónica en la República Bolivariana de Venezuela, pero es un software propietario y como tal niega a otras personas el acceso al código fuente del programa y el derecho a copiarlo, modificarlo o estudiarlo al ser desarrollado por corporaciones, en este caso Gemalto, la cual posee los derechos de autor sobre el software. Además, en caso de que la compañía fabricante

[2] Sistema de reconocimiento estadístico de patrones que establece la autenticidad de una característica fisiológica o de comportamiento que posee un usuario.

del software propietario se fuese a la banca rota, el soporte técnico desaparecería, al igual que la posibilidad de en un futuro tener versiones mejoradas y la posibilidad de corregir los errores del mismo, y de esta forma los clientes que contrataron licencias para su uso quedarían completamente abandonados a su propia suerte. Esta aplicación establece canal seguro, pero los datos viajan en texto claro, sin embargo para una mayor protección de las operaciones que se realizan con la información se requiere que se establezca canal seguro donde los datos viajen cifrados. Esto permitirá generar una transmisión segura de datos de forma tal que al enviarlos de una computadora a otra estén garantizados en cuanto a que el receptor lo comprenda, sea idéntico al enviado por el emisor y no sea interpretado por personas ajenas a la comunicación pues solo puede ser desencriptado por quien tenga la clave para hacerlo. Con esto se aseguraría la:

- Autenticidad de los usuarios.
- Confidencialidad.
- Integridad.
- No repudio.

Además de esta limitación cabe mencionar que la innovación es derecho exclusivo de la compañía fabricante, si alguien tiene una idea innovadora con respecto a una aplicación propietaria, tiene que elegir entre venderle la idea a la compañía dueña de la aplicación o escribir desde cero su propia versión de una aplicación equivalente, para una vez logrado esto poder aplicar su idea innovadora. Además es ilegal extender una pieza de software propietario para adaptarla a las necesidades particulares de un problema específico. En caso de que sea vitalmente necesaria tal modificación, es necesario pagar una elevada suma de dinero a la compañía fabricante, para que sea ésta quien lleve a cabo la modificación a su propio ritmo de trabajo y sujeto a su calendario de proyectos.

Partiendo de lo anteriormente expuesto surge como **situación problémica**: no existe un software que incluya un mecanismo de verificación biométrica de huellas dactilares, que permita verificar la autenticidad del portador de la Cédula de Identificación Electrónica (CIE), para acceder a la información que se encuentra almacenada dentro de la misma, donde se establezca canal seguro y los datos viajen cifrados, y que se pueda adaptar a las necesidades del proceso de cedulación en la República Bolivariana de Venezuela.

De aquí la interrogante en la que se enmarca el siguiente **problema científico**: ¿cómo gestionar en la Cédula de Identificación Electrónica de la República Bolivariana de Venezuela, la información biométrica de las huellas dactilares?

El **objeto de estudio** de la presente investigación es: el proceso de gestión de la información biométrica de las huellas dactilares en tarjetas inteligentes.

El **campo de acción** se centra en: la gestión de la información biométrica de las huellas dactilares, en la Cédula de Identificación Electrónica de la República Bolivariana de Venezuela.

Para dar solución al problema existente, se ha tomado como **objetivo general**: realizar el análisis y diseño de una aplicación Applet y su componente Middleware para gestionar la información biométrica de las huellas dactilares, en la Cédula de Identificación Electrónica (CIE) de la República Bolivariana de Venezuela donde se establezca canal seguro y los datos viajen cifrados.

Para dar solución a la interrogante se plantea la siguiente **hipótesis**:

Si se implementa una solución multiplataforma para gestionar la información biométrica de las huellas dactilares estableciendo canal seguro donde los datos viajen cifrados, entonces se logrará incrementar la seguridad de los servicios que se brinden con la Cédula de Identificación Electrónica.

Tareas de investigación:

- ✓ Determinar aspectos teóricos conceptuales sobre aplicaciones applet (según tecnología JavaCard) y componentes middleware.
- ✓ Documentar la situación actual de applets implementados en el mundo, relacionados con los mecanismos de autenticación y almacenamiento de huellas dactilares en tarjetas inteligentes.
- ✓ Investigar y seleccionar los estándares (JavaCard, GlobalPlatform, PC/SC, Normas ISO) para la realización de un diseño compatible.
- ✓ Analizar y seleccionar las herramientas que permitan realizar el diseño óptimo de la solución.
- ✓ Desarrollar la documentación de los diagramas de ingeniería de software necesarios para la realización de las fases de Exploración y Planificación.
- ✓ Realizar el estudio de la factibilidad de la solución propuesta.
- ✓ Desarrollar la documentación de los diagramas de ingeniería de software necesarios para el análisis y diseño de la solución; según la metodología de desarrollo XP.

Cuando existe una problemática que está afectando la sociedad, no se tiene una idea clara del asunto en cuestión y la fuente de información más apropiada es la acumulada en la bibliografía existente y la

práctica que se obtiene de los sujetos que están vinculados a la problemática que se investiga, se hace necesario utilizar como **estrategia de investigación** la exploratoria.

Los métodos científicos de investigación constituyen la forma de abordar la realidad, de estudiar la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia y sus relaciones. Estos métodos se pueden clasificar en **métodos teóricos** y **métodos empíricos**.

Los **métodos teóricos** por su parte permiten estudiar las características del objeto de investigación que no son observables directamente, facilitan la construcción de modelos e hipótesis de investigación y crean las condiciones para ir más allá de las características fenomenológicas y superficiales de la realidad, es por ello que los métodos teóricos que están presentes en este trabajo de diploma son: Analítico-Sintético, Hipotético-Deductivo, Análisis Histórico-Lógico.

- **Analítico–Sintético:** mediante este método se va a realizar un análisis de los elementos de la situación problémica, relacionándolos entre sí y vinculándolos como un todo. Además de analizar toda la teoría recopilada a través de los diferentes medios bibliográficos y poder aplicar así estos conocimientos en la práctica de manera que se adquiera una mayor preparación sobre el tema en cuestión.
- **Hipotético-Deductivo:** este método se emplea en cada fase en la cual se obtenga un resultado y se someta a análisis sobre la base de algunos conocimientos adquiridos por resultados anteriores.
- **Histórico-Lógico:** este método posibilita el análisis histórico del proceso de verificación biométrica en las tarjetas inteligentes, centrado en las huellas dactilares de las personas. Se emplea durante todo el proceso de diseño y mejoramiento de las funcionalidades de la aplicación applet y su componente middleware.

Por otro lado los **métodos empíricos** permiten, como parte del plan operativo de la investigación, determinar el método de recolección de datos y tipo de instrumento que se utilizará, es por ello que el método empírico que está presente en el actual trabajo de diploma es: observación.

- **Observación:** este método se puede ver en el proceso de cómo se almacena la huella en el applet que tiene actualmente la Cédula de Identificación Electrónica.

El presente documento consta de tres capítulos:

Capítulo 1 Fundamentación Teórica del Applet Secure Fingerprint Manager: este capítulo contiene una base teórica para entender el problema planteado. Se describen los conceptos fundamentales para el

dominio del problema, se muestra el estudio de aplicaciones applets y middleware además de hacer referencia a las tecnologías, herramientas y metodologías actuales que se usaron.

Capítulo 2 Propuesta y análisis del Applet Secure Fingerprint Manager: se presentan las fases de Exploración y Planificación definidas por la metodología XP para dar solución al problema científico. Se seleccionan las herramientas y tecnologías para desarrollar la solución. Se identifican las historias de usuarios y los requerimientos no funcionales, se realiza el plan de iteraciones y plan de entregas.

Capítulo 3 Diseño del Applet Secure Fingerprint Manager: se da cumplimiento a los planes trazados a través de las fases: Iteraciones a primera liberación donde cada historia de usuario es expresada en tareas de ingeniería y se realizan los diagramas de diseño correspondientes para la solución del problema planteado según la metodología de desarrollo XP.

Capítulo 1: Fundamentación Teórica del Applet Secure Fingerprint Manager

1.1 Introducción

No importa donde usted vaya, hoy en día, es muy probable que en algún momento alguien le solicite ver su identidad (ID) o Cédula de Identidad. Hoy, la verificación de identidad se solicita a diario en una variedad de situaciones conocidas, como cuando una persona desea obtener servicios de salud, cuando entra a un edificio público o a una oficina corporativa, o para subirse a un avión. Las organizaciones que necesitan verificar identificaciones por lo general encuentran problemas sobre la privacidad y la protección de información personal, es por ello que muchos países han adoptado la tecnología de tarjetas inteligentes como solución a muchos de estos problemas de seguridad. Con razón de hacer de las tarjetas inteligentes dispositivos aún más seguros, se hace necesaria la utilización de mecanismos de verificación biométrica por huellas dactilares, a fin de lograr una mayor seguridad para el portador de la CIE de la República Bolivariana de Venezuela.

En el presente capítulo se perfilarán los conceptos fundamentales que servirán de base para el desarrollo del trabajo. Se realizará un estudio sobre las aplicaciones (applets) existentes en la actualidad que gestionan huellas dactilares, así como las tecnologías más avanzadas para realizar este proceso 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.

1.2 Conceptos fundamentales asociados al dominio del problema

1.2.1 Cédula de Identificación Electrónica

La cédula de identidad es un documento que acredita la identidad de una persona. Es de carácter personal e intransferible, y constituye el documento principal de identificación para los actos civiles, mercantiles, administrativos y judiciales, y para todos aquellos casos en los cuales su presentación sea exigida por la ley. El gobierno de la República Bolivariana de Venezuela, con el fin de mejorar la eficiencia de los servicios de identificación de ciudadanos y responder a las necesidades que impone la creciente informatización del mundo moderno, concibe un nuevo documento de identificación, la Cédula de

Identificación Electrónica, la cual está constituida por un chip que tiene alojado los applets que permiten la gestión de la información que se almacena.

1.2.2 Applet

Es un componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo un navegador web. El applet debe ejecutarse en un contenedor, que lo proporciona un programa anfitrión, mediante un plugin, o en aplicaciones como teléfonos móviles que soportan el modelo de programación por applets.

Los applets, según el contexto del trabajo, son aplicaciones implementadas utilizando la tecnología JavaCard y son ejecutadas dentro de las tarjetas inteligentes. (thefreedictionary, 2010). El Applet Secure Fingerprint Manager que gestionará la información contenida en la CIE, se ejecutará en el contexto del JavaCard Runtime–Environment (JCRE). Este Applet será el encargado de ejecutar las operaciones que maneja, mediante los comandos APDU (Application Protocol Data Units) que le son enviados, retornando los resultados mediante APDU de respuestas.

1.2.3 Middleware

Software que funciona como una conversión o capa de traducción. Soluciones middleware personalizadas se han desarrollado durante décadas para permitir a una aplicación comunicarse con otra, ya sea que se ejecuta en una plataforma diferente o viene de un proveedor distinto. (thefreedictionary, 2010)

El middleware implementado sirve para aislar las operaciones de interacción con el applet contenido en la CIE, brindando una interpretación más adecuada de las funcionalidades que realiza.

1.2.4 Tarjetas Inteligentes

Las tarjetas inteligentes están teniendo cada vez más aceptación como la credencial de preferencia para controlar el acceso físico con seguridad. Esta tecnología tiene su origen en la década del 70 cuando inventores de Alemania, Japón y Francia inscribieron las patentes originales

Las tarjetas inteligentes son tarjetas de plástico similares en tamaño y otros estándares físicos a las tarjetas de crédito que llevan estampadas un circuito integrado. Este circuito puede ser de sola memoria o contener un microprocesador (CPU) con un sistema operativo que le permite una serie de tareas como:

1. Almacenar información.
2. Encriptar información.
3. Leer y escribir datos, como un ordenador.

Como mecanismo de control de acceso las tarjetas inteligentes hacen que los datos personales y de negocios solo sean accesibles a los usuarios apropiados, por lo que esta tarjeta asegura la portabilidad, seguridad y confiabilidad en los datos.

Las tarjetas de identificación inteligentes basadas en estándares pueden ser usadas para fácilmente autenticar la identidad de una persona, determinar el nivel de acceso adecuado y admitir físicamente al portador de la tarjeta a un servicio, o a un establecimiento. Es además un sistema portátil y seguro de almacenamiento de la información. Su característica de poseer una jerarquía en el almacenamiento de la información, permite que la tarjeta inteligente pueda ser multi-aplicación, es decir, que la misma tarjeta física tenga información de, por ejemplo, una aplicación de monedero electrónico y una aplicación de datos sanitarios del titular. Además, la jerarquía se extiende a la arquitectura de seguridad, por lo que permite que la información de las dos aplicaciones se proteja independientemente, siendo imposible para el banco observar los datos sanitarios, y para la empresa de salud los datos bancarios.

La tecnología empezó a ser aceptada y a bajar sustancialmente su coste. Esto facilitó que las tarjetas no sólo se utilizasen en programas de crédito, sino también en muchos otros sectores: fidelización de clientes, seguros sanitarios, identificación de socios de clubs, etc.

La aparición de las tarjetas inteligentes ha hecho posibles numerosas aplicaciones hasta hace poco sólo soñadas. En la actualidad se pueden ver en múltiples entornos, siendo los más activos:

•Telefonía Móvil Digital (GSM): donde la tarjeta sirve como identificación del titular del número de teléfono, herramienta de cifrado y base de datos del titular (donde se guardan sus opciones, sus números de teléfono, sus mensajes cortos, etc.)

•Banca: su gran potencial se ha intentado desarrollar mediante el denominado Monedero Electrónico, pudiendo cargarlo con un determinado importe y hacer compras frente al saldo existente en la tarjeta.

En muchos países esta tecnología se está utilizando en la salud y un ejemplo de esto es en Francia, en 1998 introdujeron su primera tecnología de tarjetas inteligentes para la atención médica, la llamada “carte vitale”. La tarjeta contiene todas las transacciones médicas del paciente, un registro al que tiene acceso cualquier médico u hospital que lo esté atendiendo.

Por otro lado las tarjetas inteligentes se pueden clasificar atendiendo a su capacidad, estructura del sistema operativo, el formato y por sus características de interfaz. En cuanto a esta última clasificación las tarjetas inteligentes pueden ser tarjetas de contacto y sin contacto. Las tarjetas de contacto disponen de unos contactos metálicos visibles y debidamente estandarizados (ISO 7816), mientras que las tarjetas sin contacto no tienen los contactos metálicos visibles. Las tarjetas de contacto, por tanto, deben ser insertadas en una ranura de un lector para poder operar con ellas. A través de estos contactos el lector alimenta eléctricamente a la tarjeta y transmite los datos oportunos para operar con ella conforme al estándar.

1.2.4.1 Tarjetas inteligentes de identificación sin contacto

Las tarjetas sin contacto funcionan mediante tecnología de radiofrecuencia e incorporan una antena interior que, mediante una señal de radio, genera la electricidad necesaria para activar el chip. Cuando en una tarjeta de contactos se producen fallos de funcionamiento, casi siempre se deben al deterioro en la superficie de contacto o a la suciedad adherida a los mismos en las tarjetas sin contactos los problemas técnicos antes mencionados no ocurren, debido claro está, a que carecen de contactos, además no hay necesidad de introducir la tarjeta en un lector, lo que significaría una gran ventaja en sistemas de control de accesos donde se necesita abrir una puerta u otro mecanismo, puesto que la autorización de acceso puede ser revisada sin que se tenga que sacar la tarjeta del bolsillo e introducirla en un terminal.

Las tarjetas sin contacto son leídas con rapidez y facilidad a una distancia de hasta diez centímetros, lo que permite utilizar las aplicaciones en entornos exigentes como el transporte público:

En los Países Bajos todo el transporte público, trenes, autobuses, metros y tranvías, comparten ahora una sola: OV-Chipkaart, es decir, una tarjeta inteligente, para hacer los pagos mucho más fáciles. Esta tarjeta simplifica el uso del transporte público en los Países Bajos. La tarjeta cuenta con un chip electrónico que se pasa a la entrada de cualquier estación de metro, autobús o tren. El precio correcto basado en la distancia recorrida se descuenta automáticamente de la tarjeta inteligente.

Actualmente hay disponibles tres tipos diferentes de esta tarjeta inteligente (OV-Chipkaart):

1) La **tarjeta inteligente personal**, que sólo puede utilizarla una misma persona, dirigida para residentes y turistas que van a usar el transporte público con demasiada frecuencia. Debe contener una foto y los datos personales del viajero.

2) La **tarjeta inteligente anónima**, puede ser compartida y no contiene información personal de los viajeros.
3) La **tarjeta inteligente desechable**, una tarjeta simple que se limita exclusivamente al precio de un solo viaje.

En Estados Unidos la autoridad municipal de transporte de San Francisco, ha comenzado a emitir tarjetas inteligentes gratuitas (TransLink sin contacto) a los pasajeros jóvenes y personas de la tercera edad, según The San Francisco Chronicle. Las tarjetas TransLink, pueden utilizarse en casi todos los medios de transportación en el área de la bahía, incluyendo BART, AC Transit, Caltrain, Golden Gate Transit y el Ferry, además de Muni.

En Malasia la Tarjeta Inteligente Multipropósito Mykad está siendo utilizada a escala nacional (18 millones de tarjetas) para manejar en una sola tarjeta: identificación personal, licencia de conducir, tarjeta de seguro, pago (ePurse) para transporte público e información de viajero.

1.3 Tendencias tecnológicas

1.3.1 Tecnología en tarjetas inteligentes

1.3.1.1 JavaCard

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

Numerosos campos de aplicación de la Internet y de las tarjetas inteligentes, especialmente el comercio electrónico, requieren que datos y recursos sean protegidos por medio de mecanismos de seguridad sofisticados. El lenguaje de programación Java representa una respuesta práctica a las cuestiones de movilidad y seguridad sobre Internet. Actualmente, se puede afirmar que Java se ha impuesto como un estándar de facto en estos dominios de aplicación donde las exigencias de seguridad son muy altas.

Sun Microsystems Inc. publicó la definición de un nuevo miembro de las tecnologías Java, llamada JavaCard, que está orientada a la programación de tarjetas inteligentes. JavaCard fue diseñado de tal forma que ciertas construcciones de Java consideradas como demasiado complejas o no aplicables para la programación de tarjetas inteligentes no son incorporadas y por otro lado se agregan facilidades específicas para el manejo de transacciones con tarjetas inteligentes (atomicidad de un grupo de

operaciones, objetos persistentes, etc.). Las políticas de seguridad de Java (conocidas como el sandbox model) que prohíben cualquier interacción entre objetos de diferentes applets fueron modificadas, y en algunos casos, debilitadas: JavaCard, por ejemplo, permite que un objeto sea compartido por diferentes applets.

Dentro de la categoría de SmartCards con microprocesador se encuentran las llamadas JavaCards o Java SmartCards. Una JavaCard es una SmartCard capaz de ejecutar programas desarrollados en Java. Originalmente, las aplicaciones para SmartCards eran escritas en lenguajes específicos de los proveedores de SmartCards (normalmente el ensamblador del microprocesador utilizado, o eventualmente en C). El lema "escribe una vez, corre en cualquier lado" hizo que Java fuese una solución a este problema. (Di Giorgio, 1998) La primera JavaCard en salir al mercado fue producida por Schlumberger. (Giorgio, 1997)

En pocas palabras, una JavaCard es una tarjeta con microprocesador que puede ejecutar programas (llamados applets) escritos en un subset[3] del lenguaje Java. Los componentes principales dentro de una JavaCard son el microprocesador y las memorias. La arquitectura básica de una JavaCard consiste de applets, JavaCard API, JavaCard Virtual Machine (JCVM) / JavaCard Runtime Environment (JCRE), y el sistema operativo nativo de la tarjeta.

JavaCard brinda al usuario la capacidad de programar aplicaciones que se ejecutan en la tarjeta de modo que ésta tenga una funcionalidad práctica en un dominio de aplicación específico (pe. identificación, pago, etc.). Esta tecnología se usa ampliamente en las tarjetas SIM (utilizadas en teléfonos móviles GSM) y en tarjetas monedero electrónico.

La primera tarjeta Java fue presentada en 1997 por varias empresas entre las que estaban Axalto (división de tarjetas de Schlumberger y Gemplus. Los productos JavaCard están basados en la Plataforma JavaCard cuyas especificaciones han sido desarrolladas por Sun Microsystems. La última versión de la plataforma JavaCard es la especificación 2.1.1, liberada por Sun en el 2000.

La máquina virtual corre sobre el sistema operativo de la tarjeta, y se puede ver como una abstracción del mismo. Sobre la máquina virtual se encuentra el JavaCard Framework, que es un conjunto de clases necesarias para el funcionamiento del JCRE y JavaCard API, así como clases utilitarias comerciales o

[3] Es un subconjunto.

extensiones propietarias del fabricante de la tarjeta. Finalmente, haciendo uso de este ambiente se encuentran los applets. (Chan) (Di Giorgio, 1998)

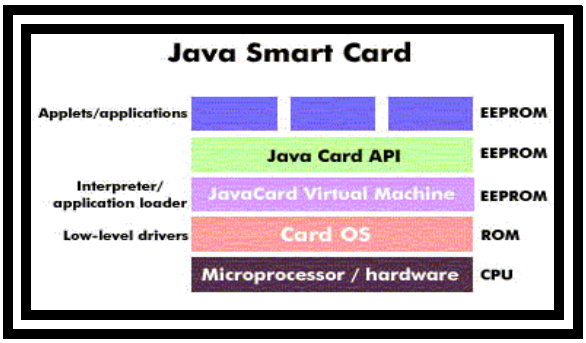

Figura 1: Arquitectura de una Smart Card

Las principales características de esta tecnología son la portabilidad y la seguridad.

Portabilidad

JavaCard tiene por objetivo la definición de un estándar de tarjeta inteligente que permite al applet funcionar en diferentes tarjetas inteligentes, de forma muy parecida a cómo un applet de Java se ejecuta en diferentes ordenadores. Al igual que en Java, esto se consigue utilizando la combinación de una máquina virtual (la máquina virtual de JavaCard), y unas librerías cuya API está especificada. La portabilidad, en todo caso, sigue siendo obviada en muchos casos por cuestiones de tamaño de la memoria, el rendimiento y tiempo de ejecución (por ejemplo para los protocolos de comunicación o algoritmos criptográficos).

Seguridad

La tecnología JavaCard fue desarrollada originalmente con el propósito de asegurar la información sensible almacenada en las tarjetas inteligentes. La seguridad está determinada por diversos aspectos de esta tecnología:

- Applet. El applet es una máquina de estados que sólo procesa los comandos recibidos a través del dispositivo lector, enviando y respondiendo con códigos de estado y datos.
- Separación de applets. Las distintas aplicaciones están, además, separadas unas de otras por un cortafuego que limita el acceso y control de los elementos de datos de un subprograma a otro.
- Encapsulación de datos. Los datos se almacenan en la aplicación y las aplicaciones JavaCard se ejecutan en un entorno aislado (la tarjeta de Java VM), separada del sistema operativo y del equipo en que se lee la tarjeta.

- Criptografía. En esta plataforma están implementados los algoritmos criptográficos más comúnmente utilizados como DES, 3DES, AES, RSA (incluyendo el uso de criptografía de curva elíptica). Otros servicios como la firma electrónica o la generación de claves de intercambio también están soportados.

1.3.1.2 Aplicaciones de JavaCard

El número de aplicaciones para JavaCards, y SmartCards en general, va en un aumento constante, y abarca áreas muy diversas. Algunos ejemplos típicos se citan a continuación:

- Monedero Electrónico: esta aplicación se utiliza como dinero electrónico. Se puede fijar un monto de dinero inicial, sobre el cual se puede realizar operaciones de débito, crédito o consulta, y puede ser utilizado para el pago o cobro de servicios o bienes. Típicamente lleva asociado algún sistema de seguridad (por ejemplo un PIN), para evitar la posibilidad de fraude.
- Transacciones Seguras: ya sea a través de cajeros automáticos o de Internet, las SmartCards proveen un nivel de seguridad muy superior al de las tarjetas magnéticas comunes o los sistemas basados en contraseñas o cookies, ya que es normal que incluyan un API de criptografía fuerte.
- Identificación Digital / Firma Digital: este tipo de aplicaciones se utiliza para validar la identidad del portador de la tarjeta, o para poder certificar el origen de ciertos datos. Normalmente se basan fuertemente en las primitivas criptográficas del API y/o las que están implementadas en hardware.
- Sistemas de Prepago: en estos sistemas, un cliente "carga" su tarjeta con una cierta "cantidad" de dinero, el cual va siendo decrementado a medida que el cliente hace uso del servicio. El servicio puede variar desde telefonía celular hasta TV cable, pasando por acceso a sitios web o transporte público.
- Tarjetas de salud: en algunos hospitales ya se está implementando un sistema de identificación de pacientes y almacenamiento de los principales datos de la historia clínica de los mismos en SmartCards para agilizar la atención. Actualmente la capacidad de almacenamiento es muy limitada, pero en un futuro se podría almacenar toda la historia clínica (incluidas radiografías y similares) en una SmartCard.
- Control de Acceso y de Asistencia: ya hay varios sistemas de control de acceso y asistencia implementados en base a SmartCards o a iButtons. (Chan) (Sun Microsystem, 1999)

Existen más aplicaciones de las SmartCards y JavaCards, algunas son variantes de las mencionadas anteriormente, mientras otras satisfacen necesidades específicas de una empresa.

1.3.1.3 JavaCard API 2.0 y 2.1

La última versión de la plataforma JavaCard es la especificación 2.1.1, liberada por Sun en el 2000, y actualmente hay algunas implementaciones en el mercado o en vías de ser comercializadas. (Gemplus) (Schlumberger)

Existen además otros sistemas basados en la especificación 2.0, como los iButtons (Dallas Semiconductor), que ya están ampliamente difundidos. La principal innovación que se introdujo en la versión 2.1 es la facilidad de compartir objetos entre applets.

1.3.1.4 JavaCard Runtime Environment (JCRE)

El JCRE comprende la máquina virtual de JavaCard (JCVM) junto a las clases y servicios definidos en el Application Programming Interface (API). Sobre este ambiente se ejecutan los applets que se desarrollan.

La JCVM se diferencia principalmente de una JVM normal en que el tiempo de vida de la misma es igual al tiempo de vida de la tarjeta, por lo cual los objetos mantienen sus estados entre dos sesiones con una terminal (CAD[4]). Es responsabilidad del JCRE garantizar este comportamiento. Cuando se retira la tarjeta del CAD, se asume que se está ejecutando en un ciclo de reloj de período infinito. (Sun Microsystem, 1999) Otras diferencias entre ellas son las limitaciones en los tipos de datos manejados y los requerimientos de hardware para la ejecución. (Sun Microsystem, 1999)

Para poder comprender como funciona una JavaCard, hay que tener en cuenta que al realizar la especificación de la plataforma, Sun se apegó al estándar ISO 7816, el cual establece, entre otras cosas, la forma de comunicación entre una SmartCard y un CAD. De acuerdo al ISO 7816, el intercambio de información y comandos entre la tarjeta y el CAD se realiza a través de APDUs (Application Protocol Data Units), los cuales son paquetes de información con un formato específico. (Sun Microsystems, 1999.) (Chan) (Giorgio, 1997)

De acuerdo al estándar, las SmartCards nunca inician la comunicación con el CAD, sino que sólo responden a los comandos que éste le envía. Se define dos tipos de APDU, los llamados COMMAND

[4] Tarjeta de dispositivo de aceptación

APDU, que son los que envía el CAD a la tarjeta, y los RESPONSE APDU, que son los que envía la tarjeta al CAD como respuesta a un COMMAND APDU.

La siguiente tabla describe el formato de ambos tipos de APDU.

Command APDU						
Encabezado obligatorio				**Cuerpo opcional**		
CLA	INS	P1	P2	LC	Data Field	LE
Response APDU						
Cuerpo opcional				**Cola obligatoria**		
Data Field				SW1		SW2

Tabla 1. Comando APDU

Campo	Tamaño	Descripción
CLA	**1bytes**	Clase de instrucción. Indica la estructura y el formato para una categoría de COMMAND y RESPONSE APDU.
INS	**1bytes**	Código de instrucción. Especifica la instrucción del comando.
P1	**1bytes**	Parámetros de la instrucción. Proveen más información sobre la instrucción.
P2	**1bytes**	
LC	**1bytes**	Número de bytes en el Data Field del APDU.
Data Field	**LC bytes**	Secuencia de bytes con información.
LE	**1bytes**	Cantidad máxima de bytes esperados como respuesta.

Figura 2: Estructura del comando APDU

1.3.1.5 JavaCard Applet

Los applets son las aplicaciones que corren embarcadas en una JavaCard. Dichas aplicaciones interactúan en todo momento con el JCRE utilizando los servicios que éste brinda, e implementan la interfaz definida en la clase abstracta javacard.framework.Applet, la cual deben extender. (Sun Microsystem, 1999)

Se puede decir que un applet comienza su ciclo de vida al ser correctamente cargado en la memoria de la tarjeta, link - editada y preparada para su correcta ejecución. (Sun Microsystems, 1999)

Una vez registrada en el JCRE con el método register (), un applet está en condiciones de ejecutarse. Este applet normalmente existe durante el resto de vida de la tarjeta.

La clase javacard.framework.Applet define cuatro métodos públicos que son utilizados por el JCRE para hacer funcionar las aplicaciones.

- Método install (byte(), short, byte)

Este método es invocado por el JCRE antes de crear una instancia del applet en la tarjeta. La implementación usual de este método es llamar al constructor de la clase, que normalmente es privado, crear todos los objetos que el applet necesitará para su ejecución, y por último registrar el applet con el método register().

- Método select ()

Este método es invocado por el JCRE como consecuencia de la recepción a un SELECT APDU. Este APDU, contiene el Application IDentifier (AID) del applet a seleccionar.

El AID es una secuencia de entre 5 y 16 bytes, que identifica de forma única una aplicación para SmartCards, de acuerdo al ISO 7816, y es la misma ISO la que otorga los AIDs.

- Método process (APDU)

Cuando llega un APDU el JCRE invoca este método del applet seleccionado, pasándole como parámetro el COMMAND APDU recibido. Dentro de este método, el applet identifica el comando asociado al APDU y los parámetros, si los hay, y los procesa de acuerdo al protocolo que se haya definido para la interacción entre el applet y la aplicación terminal.

- Método deselect ()

Este método es invocado por el JCRE para avisar al applet que está actualmente seleccionado y que va a dejar de estarlo. Esto sucede cuando el JCRE recibe un SELECT APDU (aún cuando el AID del applet a seleccionar coincida con el del applet seleccionado). Esto brinda al applet la oportunidad de realizar las tareas de limpieza que sean necesarias para quedar en un estado consistente.

La tecnología JavaCard combina parte del lenguaje de programación Java con un entorno de ejecución optimizado para SmartCards y similares. El objetivo de la tecnología JavaCard es llevar los beneficios del desarrollo de software en Java al mundo de las SmartCards.

1.3.2 Tecnología Biométrica

1.3.2.1 Identificación Biométrica

Según el Diccionario de la Real Academia Española, se define *biometría* como "estudio mensurativo o estadístico de los fenómenos o procesos biológicos". (Álvarez Reyes) Esta definición se hace más específica cuando se utiliza el término de biometría dentro del campo de la identificación de personas. Se podría decir en este caso, que biometría es la ciencia por la que se puede identificar a una persona basándose en sus características biofísicas o de comportamiento. Expuesto en forma de ejemplos, es la ciencia que consigue reconocer a una persona mediante una imagen de su rostro o mediante la impresión de su huella dactilar.

Existen diferentes técnicas de identificación biométricas, pero este trabajo sólo se enfocará en las huellas dactilares. Su uso data de tiempos muy antiguos: las huellas dactilares se usaron en tablillas de arcilla para transacciones comerciales en la antigua Babilonia. Hace ya unos 5000 años, los chinos eran conscientes de la individualidad de las huellas, y la colocación de éstas de forma muy cuidadosa en los objetos manufacturados plantea la posibilidad de que lo utilizasen como forma de firmar sus creaciones. A finales del siglo XVIII y principios del XIX, el inglés Thomas Bewick, es considerado el primer caso oficial de firma mediante la huella dactilar, al utilizar esa técnica para firmar sus libros.

Sir Francis Galton, a finales del siglo XIX realizó estudios muy detallados sobre la huella dactilar, estudiando su estabilidad, unicidad y morfología. Sus trabajos, complementados por los de Vucetich, Henry, Hershel y Faulds (cada uno de forma independiente), consiguieron que la identificación por huella fuera aceptada y se convirtiera en el método de identificación biométrica más utilizado por la policía mundial. (Álvarez Reyes) Pero la identificación de huellas digitales no se quedó como un método sólo forense, sino que se utiliza oficialmente para fines comerciales; como un seguro y efectivo método de autenticación en numerosos campos, incluido el financiero, médico, el comercio electrónico y las aplicaciones de control de entrada.

La identificación por huella dactilar es, sin lugar a duda, la más estudiada y probada. Existen numerosos estudios científicos que avalan la unicidad de la huella de una persona y, lo que es más importante, la estabilidad con el tiempo, la edad, etc. En estos aspectos es una técnica que lleva mucha ventaja con respecto a otras, debido a su siglo de existencia.

La autenticación o verificación trata de responder a la pregunta: ¿es éste sujeto la persona que dice ser? En este esquema de funcionamiento, el usuario, al que se le toman sus características biométricas, también comunica su identidad. De esta forma, el sistema se encarga de comparar las características extraídas, con el patrón del usuario indicado. Si la comparación supera un determinado umbral de parecido, se considera que el usuario es el indicado, rechazando la comparación en caso contrario. El patrón, es almacenado en un sistema portátil de información como puede ser una tarjeta inteligente. (Agora SIC, 2000)

1.3.2.2 Match on Card

Match on Card (MoC) es una tecnología que realiza comparaciones de huellas dactilares en tarjetas. Las tecnologías biométricas han ido fortaleciendo los mecanismos de autenticación al comparar la plantilla biométrica almacenada con la plantilla biométrica capturada al momento de la comparación (plantilla en vivo). En el caso de las tarjetas inteligentes, esta comparación se hace dentro de la tarjeta lo cual requiere una capacidad de procesamiento interno que dependerá de la complejidad de la información biométrica a comparar (huellas dactilares, iris, tomografía facial, geometría de la mano, etc.) y de los algoritmos usados.

El proceso biométrico utilizando la tecnología MoC se divide en dos funciones a realizar: *El enrolamiento y la verificación* de las huellas digitales en la tarjeta. (Precise Biometrics) La plantilla biométrica es almacenada dentro de la tarjeta, que también realiza la comparación con la plantilla en vivo. Por tanto, se necesita una capacidad de procesamiento interno como la del microprocesador de una tarjeta inteligente, lo cual conlleva al uso de un sistema operativo que ejecute las aplicaciones de comparación necesarias. (Mendoza, 2008) (Precise Biometrics, 2005)

1.3.3 Estándares relacionados con tarjetas inteligentes

1.3.3.1 Estándar ISO/IEC- 7816

ISO/IEC - 7816 es un estándar internacional relacionado con las tarjetas de identificación electrónicas, en especial las tarjetas inteligentes, gestionado conjuntamente por la Organización Internacional De Normalización (ISO) y Comisión Electrotécnica Internacional (IEC). Se trata de una extensión de la ISO 7810 e ISO 7811, los cuales definen características físicas de tarjetas de identificación. La tarjeta

inteligente más básica cumple los estándares de la serie ISO 7816. El objetivo 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.

1.3.3.1.1 Descripción de las diferentes partes de esta norma (ISO/IEC 7816).

- ✓ **7816-1: Características físicas.**

En el estándar ISO/IEC 7816-1 se definen los siguientes tamaños para tarjetas inteligentes:

- **ID 000**: el de las tarjetas SIM usadas para teléfonos móviles GSM. También acostumbran a tener este formato las tarjetas SAM (Security Access Module) utilizadas para la autenticación criptográfica mutua de tarjeta y terminal.
- **ID 00**: un tamaño intermedio poco utilizado comercialmente.
- **ID 1**: el más habitual, tamaño tarjeta de crédito.

- ✓ **7816-2: Tarjetas con contactos - Dimensiones y localización de los contactos.**

Creado en 1988, actualizado en 1999, modificado en 2004.

- ✓ **7816-3: Señales electrónicas y protocolo de transmisión.**
- ✓ **7816-4: Comandos de intercambio inter-industriales.**

ISO/IEC 7816-4:2005 es independiente de la tecnología de la interfaz física (no sólo se aplica a tarjetas con contactos). Se aplica a las tarjetas de contactos, de proximidad, radiofrecuencia, etc.

- ✓ **7816-5: Sistema de numeración y procedimientos de registro.**

Desde su resumen, ISO/IEC 7816-5 define cómo usar un identificador de aplicación para determinar la presencia y/o realizar la recuperación de una aplicación en una tarjeta.

- ✓ **7816-6: Elementos de datos inter-industriales.**

Desde su resumen, especifica los elementos de datos utilizados para el intercambio basado en las tarjetas de circuito integrado, con contactos y sin contactos. Se da el identificador, el nombre, la descripción, el formato, la codificación y el diseño de cada elemento de datos y define los medios de recuperación de éstos desde la tarjeta.

- ✓ **7816-7: Comandos inter-industriales y consultas estructuradas para una tarjeta.**
- ✓ **7816-8: Comandos inter-industriales relacionados con seguridad.**

Desde su resumen, especifica los comandos de las tarjetas (ya sea con o sin contactos) que pueden utilizarse para operaciones criptográficas. Estos comandos son complementarios, y sobre la base de los comandos enumerados en la norma ISO/IEC 7816-4.

- ✓ **7816-9: Comandos adicionales inter-industriales y atributos de seguridad.**

Desde su resumen, especifica los comandos de las tarjetas (con contactos y sin contactos) para la gestión de ficheros, por ejemplo la creación y borrado de ficheros. Estos comandos abarcan todo el ciclo de vida de la tarjeta y, por consiguiente, algunos comandos pueden ser usados antes de que la tarjeta haya sido expedida a su titular o después de que ésta haya caducado.

- ✓ **7816-10: Señales electrónicas y respuesta para reiniciar una tarjeta inteligente síncrona.**
- ✓ **7816-11: Verificación de la identidad personal a través de métodos biométricos.**

Desde su resumen, especifica el uso de los comandos y de los datos relacionados con la verificación de la identidad de una persona a través de los métodos biométricos en las tarjetas de circuito integrado. Los comandos utilizados se definen en la norma ISO/IEC 7816-4. Los datos se definen parcialmente en esta norma y en parte importados de la norma ISO/IEC 19785-1

- ✓ **7816-12 Tarjetas con contactos. Interfaz eléctrica USB y procedimientos operativos.**

Desde su resumen, especifica las condiciones de funcionamiento de una tarjeta de circuito integrado a través de una interfaz USB

ISO/IEC 7816-12:2005 proporciona dos protocolos para controlar las transferencias. Se trata de soportar el protocolo T=0 (versión A) o utilizar la transferencia de APDU (versión B). ISO/IEC 7816-12:2005 proporciona los diagramas de estado para la interfaz USB-ICC para cada una de las transferencias (transferencias a granel, el control de las transferencias versión A y versión B)

- ✓ **7816-13: Comandos de administración de aplicaciones en múltiples aplicaciones entorno.**
- ✓ **7816-15: Aplicación de información criptográfica.**

Desde su resumen, especifica una aplicación que contiene información sobre la funcionalidad criptográfica. Por otra parte, ISO/IEC 7816-15:2004 define una sintaxis común (en ASN.1) y el formato de codificación de la información y los mecanismos para compartir esta información cuando sea apropiado

1.3.3.2 Estándar GlobalPlatform

GlobalPlatform es el líder mundial en el desarrollo inteligente de las infraestructuras de la tarjeta. Sus especificaciones técnicas probadas para las tarjetas, dispositivos y sistemas se consideran como el estándar de la industria para lograr implementaciones interoperables inteligente, flexible y sostenible para la tarjeta que soporte multi-aplicación y multi-actor e implementaciones multi-modelo de negocio.

GlobalPlatform es una organización enfocada a gestionar una infraestructura estandarizada para el desarrollo y despliegue de tarjetas inteligentes. Proporciona un conjunto de especificaciones universalmente reconocidas e implementadas, junto con configuraciones de mercado, aplicación de esas especificaciones y documentos de apoyo. Cubriendo toda la infraestructura de tarjetas inteligentes (las tarjetas, dispositivos y sistemas) estos documentos técnicos ofrecen una plataforma tecnológica dinámica y completa para el desarrollo de programas de tarjetas inteligentes, para poder establecer una conexión segura con la misma y administrar sus aplicaciones.

Las tarjetas, dispositivos y sistemas GlobalPlatform, son interoperables, independientemente de la tecnología del proveedor y la flexibilidad de su infraestructura técnica, garantizan que pueda responder a las necesidades básicas en el instante del despliegue inicial.

1.3.3.3 Estándar PC/SC

PC / SC (en inglés Personal Computer/SmartCard) es un estándar para las tarjetas inteligentes de acceso en plataformas Windows (incluido en Windows 2000). En particular se define un API[5] de programación, que permite a los desarrolladores trabajar de forma uniforme con lectores de tarjetas de distintos fabricantes (que cumplan con la especificación). El API de PC/SC está incorporada en sistemas Microsoft Windows 200x/XP y Microsoft Windows NT/9x. También hay una implementación libre, de código abierto, llamada PC/SC Lite (proyecto MUSCLE) para sistemas operativos GNU Linux.

La principal ventaja de PC / SC es que las aplicaciones no tienen que estar al tanto de los detalles relativos al lector de tarjetas inteligentes con el fin de comunicarse con la tarjeta inteligente. Por otra parte, la aplicación puede funcionar con cualquier lector que cumpla con este estándar.

[5] interfaz de programación de aplicaciones.

Cabe destacar que Microsoft ha desempeñado un papel importante en la elaboración de estas normas, debido al hecho de que las plataformas de Windows son los únicos que ofrecen un gestor de recursos de tarjetas inteligentes.

La especificación se divide en 10 partes que contienen los requisitos detallados de interoperabilidad de dispositivos compatibles, información de diseño, interfaces de programación y otras. Estas partes son:

- ✓ 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 conectados.
- ✓ Parte 4. Consideraciones de diseño e información de referencia de los lectores.
- ✓ 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.
- ✓ Parte 10. Lectores con capacidades de entrada de PIN de seguridad.

1.3.3.4 Estándar PKI

Una infraestructura de clave pública (PKI) es un conjunto de aplicaciones y servicios que permite utilizar la criptografía de clave pública (certificados) de una forma fácil y efectiva. Se basa en la criptografía de clave pública, su uso más común se plasma en la firma digital, ésta es ideal para prestar servicios como la autenticación de usuarios permitiendo:

- ✓ Asegurarse de la identidad de un usuario como responsable de documentos o para garantizar el acceso a servicios distribuidos en la red, ya que sólo él puede conocer su clave privada, evitando así la suplantación.
- ✓ No repudio, para impedir que una vez firmado un documento el signatario se retracte o niegue haberlo redactado.
- ✓ La integridad de la información para prevenir la modificación deliberada o accidental de los datos firmados durante su transporte, almacenamiento o manipulación.
- ✓ El acuerdo de claves secretas para garantizar la confidencialidad de la información intercambiada.

PKI se puede utilizar para:

- Gestión de claves: permite crear, revisar o revocar claves, así como gestionar niveles de confianza.
- Publicación de claves: una vez creadas las claves, PKI permite difundir la clave pública, así como localizar las claves públicas de otros usuarios, junto con su estado (clave revocada, etc.).
- Utilización de claves: una vez recuperada una clave, PKI facilita el uso de la misma.

La Infraestructura de Llave Pública o PKI:

Es la integración de:

a) La Criptografía de llave pública o asimétrica, usada para la firma digital

b) La Criptografía de llave simétrica usada para cifrar

c) El Hash

d) La Gestión de los pares de Llaves Público / Privados (El no compromiso de la llave privada a través de un procedimiento de distribución segura de llaves)

1.3.4 Tecnologías de desarrollo

1.3.4.1 JCardManager

El JCardManager es una aplicación cliente genérico basado en terminal. Esta herramienta permite la comunicación con los applets y probarlos sin necesidad de desarrollar la aplicación propia cliente. El JCardManager puede comunicarse con tarjetas reales y simuladores de tarjeta.

Las principales funciones de la JCardManager son:

- El acceso a la GxpConverter para convertir archivos de clase de formatos listos para ser cargados en una tarjeta o el GSE (simulador de tarjeta).
- La gestión de la capa de OCF comunicaciones.
- El acceso al Editor de archivos de implementación para el despliegue de registro y reproducción, lo que permite un paso de carga, la instalación y selección de applets.
- Realización de tarjeta de autenticación de terminales.
- La carga y la instalación de los applets en la tarjeta.
- Gestión de las características de cifrado de la tarjeta.
- Visualización y grabación de un seguimiento de toda la actividad entre el programa y el objetivo.
- Viendo el contenido de una tarjeta.

- El envío de comandos de alto nivel de APDU o comandos definidos por el usuario a un objetivo y la observación de las respuestas.
- Grabación y reproducción de secuencias de comandos para automatizar la emisión de secuencias de comandos.

1.3.4.2 Developer Suite

Developer Suite proporciona un conjunto de herramientas de desarrollo para crear y depurar applets JavaCard. Te permite trabajar desde dentro de Eclipse para desarrollar applets. Este además facilita el desarrollo de soluciones inalámbricas para los desarrolladores de Java, la gestión del desarrollo directamente, en un ambiente familiar (Java Entorno de desarrollo integrado (IDE)). Por lo tanto, es un desarrollador sin problemas que comprende las particularidades de JavaCard. Además brinda un ambiente favorable para el diseño y la implementación de applets y posibilita simular las funcionalidades de los applets antes de ser instalados en las tarjetas inteligentes. (Gemalto)

Developer Suite viene con una simulación de extremo a extremo Suite:

- Tarjeta de Simuladores de tarjetas SIM, tarjetas USIM y tarjetas R-UIM, y las tarjetas NFC SCWS.
- Simuladores de móviles 2G, 3G y CDMA.
- Simuladores Server para 2G, 3G y SCWS OMA.

1.3.4.3 Metodologías de desarrollo

Desarrollar un buen software depende de un sinnúmero de actividades y etapas, donde el impacto de elegir la mejor metodología para un equipo en un determinado proyecto, es trascendental, para el éxito del producto.

A continuación se detallan los dos grandes enfoques, tanto metodologías tradicionales y metodologías ágiles, las primeras recalcan el uso exhaustivo de documentación durante todo el ciclo del proyecto, mientras que las segundas ponen vital importancia en la capacidad de respuesta a los cambios y al mantener una buena relación con el cliente para llevar al éxito el proyecto.

1.3.4.3.1 Metodologías Tradicionales

Las metodologías tradicionales se centran en la documentación, planificación y los procesos (plantillas, técnicas de administración, revisiones, etc.). Otra de las características importantes dentro de este enfoque, son los altos costos al implementar un cambio y la falta de flexibilidad en proyectos donde el entorno es volátil. Entre las principales metodologías tradicionales tenemos a RUP, que centra su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo del proyecto.

1.3.4.3.1.1 Rational Unified Process

RUP es un proceso formal: Provee un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales (respetando cronograma y presupuesto).
Fue desarrollado por Rational Software, y está integrado con toda la suite Rational de herramientas. Puede ser adaptado y extendido para satisfacer las necesidades de la organización que lo adopte. Es un producto que se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso, y utiliza UML como lenguaje de notación. Incluye artefactos, que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc. y además incluye roles, que no es más que el papel que desempeña una persona en un determinado momento (una persona puede desempeñar distintos roles a lo largo del proceso).

1.3.4.3.2 Metodologías Ágiles

Los métodos ágiles, nacen como una respuesta a los problemas de las metodologías tradicionales y se basan en dos aspectos puntuales, el retrasar las decisiones y la planificación adaptativa; permitiendo potenciar aún más el desarrollo de software a gran escala. Estas metodologías ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan. Nos lo proponen porque para muchos clientes esta flexibilidad será una ventaja competitiva y porque estar preparados para el cambio significa reducir su coste.
Como resultado de esta nueva teoría se crea un Manifiesto Ágil cuyas principales ideas son:

- ✓ Los individuos y las interacciones entre ellos son más importantes que las herramientas y los procesos empleados.

- ✓ Es más importante crear un producto software que funcione que escribir documentación exhaustiva.
- ✓ La colaboración con el cliente debe prevalecer sobre la negociación de contratos.
- ✓ La capacidad de respuesta ante un cambio es más importante que el seguimiento estricto de un plan.

Entre los principales métodos ágiles tenemos el XP (eXtreme Programming), Scrum, entre otras.

1.3.4.3.2.1 Extreme Programming

Es la más destacada de los procesos ágiles de desarrollo de software formulada por Kent Beck. La programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos. Son estrategias de desarrollo de software que promueven prácticas que son adaptativas en vez de predictivas, centradas en las personas o en los equipos, iterativas, orientadas hacia prestaciones y hacia la entrega, de comunicación intensiva, y que requieren que el negocio se involucre en forma directa.

1.3.4.3.2.2 Scrum

Scrum define un marco para la gestión de proyectos y está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en:

- ✓ El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente.
- ✓ Las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.

1.3.4.4 Comparación entre RUP y XP

RUP

- Intenta conseguir el objetivo común por medio del orden y documentación exhaustiva.
- Está pensado para proyectos y equipos de trabajo más grandes en cuanto a tamaño y duración del mismo.
- La primera fase, llamada inicio es más lenta que en la metodología XP, ya que puede tener varias iteraciones.
- En la Fase de "Elaboración" se presentan los casos de uso y se establecen con claridad los requerimientos del sistema.
- La fase de desarrollo implementa iteraciones.

XP

- Disminución en la documentación, mejorando los procesos de comunicación directa e inmediata entre las personas que intervienen en el proceso.
- Trabajo por equipos.
- Se implementa mejor en proyectos cortos y equipos más pequeños.
- En un proyecto XP la primera fase es llamada "Exploración" y es una fase mucho más rápida que la primera fase en RUP.
- La segunda fase de XP la "Planificación".
- XP está diseñado con los programadores en mente, con facilitar su trabajo y por tal razón define todo el desarrollo completo incluyendo pruebas e integración.

1.3.4.5 Comparación entre Scrum y XP

- Semejanzas

Tanto Scrum como Programación Extrema (XP) requieren que los equipos completen algún tipo de producto potencialmente liberable al final de cada iteración. Estas iteraciones están diseñadas para ser cortas y de duración fija. Este enfoque en entregar código funcional cada poco tiempo significa que los equipos Scrum y XP no tienen tiempo para teorías. No persiguen dibujar el modelo UML perfecto en una herramienta CASE, escribir el documento de requisitos perfecto o escribir código que se adapte a todos los cambios futuros imaginables. En vez de eso, los equipos Scrum y XP se enfocan en que las cosas se hagan. Estos equipos aceptan que puede que se equivoquen por el camino, pero también son conscientes

de que la mejor manera de encontrar dichos errores es dejar de pensar en el software a un nivel teórico de análisis y diseño y sumergirse en él, ensuciarse las manos y comenzar a construir el producto.

- Diferencias

Scrum es una metodología ágil que es aplicable si desea "una mayor visibilidad, la participación de mejores negocios, la colaboración en equipo, y un claro proceso de establecimiento de prioridades." (Kniberg, 2007). Es un método de la participación de los accionistas del proyecto de toda la organización, y mantenerse al día con las cambiantes necesidades en lugar de tratar de luchar contra ellos.

Scrum y XP se centran en diferentes aspectos del desarrollo. Scrum en la gestión de proyectos, XP en la ingeniería. Es posible utilizar ambos al mismo tiempo. Si usted está más centrado en la creación de un código limpio, estable, debería utilizar XP. Si están más centrados en gestionar con éxito su proyecto, la metodología a utilizar sería Scrum.

1.3.4.4 UModel Altova

UModel se utiliza para crear e interpretar diseños de software mediante la potencia del estándar UML 2.1. Dibuja el diseño de la aplicación y puede generar código para Java o C# a partir de planos, así como que permite realizar ingeniería inversa de programas existentes a diagramas UML claros y precisos para abarcar rápidamente su arquitectura software. Incluso, con la utilización de UModel se puede corregir el código generado o los modelos y completar la ronda produciendo automáticamente nuevos diagramas o regenerando el código.

1.3.4.5 Plataforma.NET

1.3.4.5.1 Microsoft Visual Studio

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión net 2002). Así se

pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

Este entorno de desarrollo integrado presenta varias versiones:

- Microsoft Visual Studio 6

Se lanzó en 1998 y fue la última versión en ejecutarse en la plataforma Win9x. Esta versión fue la base para el sistema de desarrollo de Microsoft para los siguientes 4 años, en los que Microsoft migró su estrategia de desarrollo al .NET Framework.

Visual Studio 6.0 fue la última versión en que *Visual Basic* se incluía de la forma en que se conocía hasta entonces; versiones posteriores incorporarían una versión muy diferente del lenguaje con muchas mejoras, fruto de la plataforma .NET.

- Microsoft Visual Studio .NET (2002)

En esta versión se produjo un cambio sustancial, puesto que supuso la introducción de la plataforma .NET de Microsoft. .NET en una plataforma de ejecución intermedia multilenguaje, de forma que los programas desarrollados en .NET no se compilen en lenguaje máquina, sino en un lenguaje intermedio (CIL - Common Intermediate Language) denominado Microsoft Intermediate Language (MSIL). Visual Studio .NET 2002 supuso también la introducción del lenguaje C#, un lenguaje nuevo diseñado específicamente para la plataforma .NET, basado en C++ y Java.

- Microsoft Visual Studio .NET 2003

Visual Studio .NET 2003 supone una actualización *menor* de Visual Studio .NET. También se añade soporte con el fin de escribir aplicaciones para determinados dispositivos móviles, ya sea con ASP.NET o con el .NET Compact Framework.

- Microsoft Visual Studio 2005

Visual Studio 2005 se empezó a comercializar a través de Internet a partir del 4 de Octubre de 2005 y llegó a los comercios a finales del mes de Octubre en inglés. En castellano no salió hasta el 4 de Febrero de 2006. La actualización más importante que recibieron los lenguajes de programación fue la inclusión de tipos genéricos, similares en muchos aspectos a las plantillas de C++. Con esto se consigue encontrar muchos más errores en la compilación en vez de en tiempo de ejecución, incitando a usar comprobaciones estrictas en áreas donde antes no era posible.

- Microsoft Visual Studio 2008

El nuevo framework (.Net 3.5) está diseñado para aprovechar las ventajas que ofrece el nuevo sistema operativo "Windows Vista" a través de sus subsistemas "Windows Communication Foundation" (WCF) y "Windows Presentation Foundation" (WPF). El primero tiene como objetivo la construcción de aplicaciones orientadas a servicios mientras que el último apunta a la creación de interfaces de usuario más dinámicas que las conocidas hasta el momento.

- Microsoft Visual Studio 2010

Visual Studio 2010 es la versión más reciente de esta herramienta, acompañada por .NET Framework 4.0. Entre sus más destacables características, se encuentran la capacidad para utilizar múltiples monitores, así como la posibilidad de desacoplar las ventanas de su sitio original y acoplarlas en otros sitios de la interfaz de trabajo.

1.3.4.5.2 Framework .NET

.NET es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

.NET podría considerarse una respuesta de Microsoft al creciente mercado de los negocios en entornos Web, como competencia a la plataforma Java de Sun Microsystems y a los diversos framework de desarrollo web basados en PHP. Su propuesta es ofrecer una manera rápida y económica, a la vez que segura y robusta, de desarrollar aplicaciones o como la misma plataforma las denomina, soluciones permitiendo una integración más rápida y ágil entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo.

La plataforma .NET de Microsoft provee un extenso conjunto de soluciones predefinidas para necesidades generales de la programación de aplicaciones, y administra la ejecución de los programas escritos específicamente con la plataforma. Esta solución es el producto principal en la oferta de Microsoft, y pretende ser utilizada por la mayoría de las aplicaciones creadas para la plataforma Windows.

El Framework de .Net es una infraestructura sobre la que se reúne todo un conjunto de lenguajes y servicios que simplifican enormemente el desarrollo de aplicaciones. Mediante esta herramienta se ofrece un entorno de ejecución altamente distribuido, que permite crear aplicaciones robustas y escalables.

1.4 Applet que gestiona huellas dactilares

1.4.1 CryptoManager Applet

El CryptoManager es el punto de partida para el paquete de cifrado. Se usa para inicializar el subsistema, así como para buscar llaves, fichas y otros. Este está diseñado para desarrolladores que trabajan con información biométrica y que realizan comparación de huellas dactilares. Además el CryptoManager provee información, con la cual es posible para los clientes, configurar sus propias condiciones de acceso en orden de proteger sus datos. Este applet necesita como requisitos de sistema los siguientes:

Requisitos de hardware

- Tarjeta Inteligente (SmartCard) y escáner de huellas dactilares.
- La aplicación JavaCard debe ejecutar las APIs biométricas (BioMatch™ C or Precise BioMatch™ J).

Requisitos de software

- Se precisa de BioMatch™ Pro Toolkit 2.0 para generar y trabajar con las plantillas compatibles de BioMatch.

1.4.1.1 Arquitectura del CryptoManager

El CryptoManager es un applet de servidor que controla las funcionalidades biométricas presente en la tarjeta, tales como:

 Proporcionar una API externa (borde de la tarjeta), permitiendo que un host pueda utilizar la biometría presente en la tarjeta.

 Proporcionar una API interna, haciendo posible que otros applets cliente puedan utilizar las funcionalidades biométricas.

 Gestión de datos biométricos, de acuerdo al algoritmo (s) y a la actual API biométrica presentes en la tarjeta.

Todos los datos biométricos asociados a una huella dactilar específica se almacenan en un contenedor de plantilla, es decir, la plantilla biométrica y cualquier información biométrica (pública), así como conocer qué huella específica se ha extraído de la plantilla.

Durante la instalación del applet un número configurable de contenedores plantilla se asigna y se inicializan. El CryptoManager admite un máximo de diez contenedores plantilla.

1.4.1.2 Llaves del CryptoManager

El CryptoManager utiliza diferentes llaves para acceder a los diferentes ciclos de estado de la tarjeta. Cada ciclo de vida de un estado ofrece diferentes tipos de funcionalidades.

- ✓ **Llave de administración:** El AdminKey o llave de administración es una clave que consiste en un máximo de ocho bytes. El administrador de la tarjeta debe configurar esta llave durante el estado de inicialización del CryptoManager, es decir, el CryptoManager no puede comenzar el estado de inicialización si esta llave no ha sido configurada. La llave de administración es usada para acceder al estado de administración.
- ✓ **Llave de inicio:** El StarKey o llave de inicio es creada e inicializada por la misma tarjeta durante el estado virgen. La llave de inicio es usada para bloquear y desbloquear la llave de administración del CryptoManager cuando está bloqueada.
- ✓ **Llave privada RSA:** La llave privada RSA es creada por el host y enviada a la tarjeta durante el estado de inicialización. Como resultado, esta no puede ser cambiada a menos que el host retenga la llave de inicio mientras que la tarjeta esté siendo usada y tenga acceso nuevamente al estado de inicialización. Esta llave es provista por una tarjeta en forma CRT y es almacenada de forma segura en la tarjeta a través de todo el ciclo de vida del CryptoManager.

1.4.1.3 Estados del ciclo de vida del CryptoManager

En el orden de asegurar las diferentes operaciones que son realizadas en los estados del ciclo de vida del CryptoManager Applet, diferentes niveles de acceso son proveídos. La figura 8 ilustra los estados del ciclo de vida del CryptoManager y las posibles transiciones entre ellos.

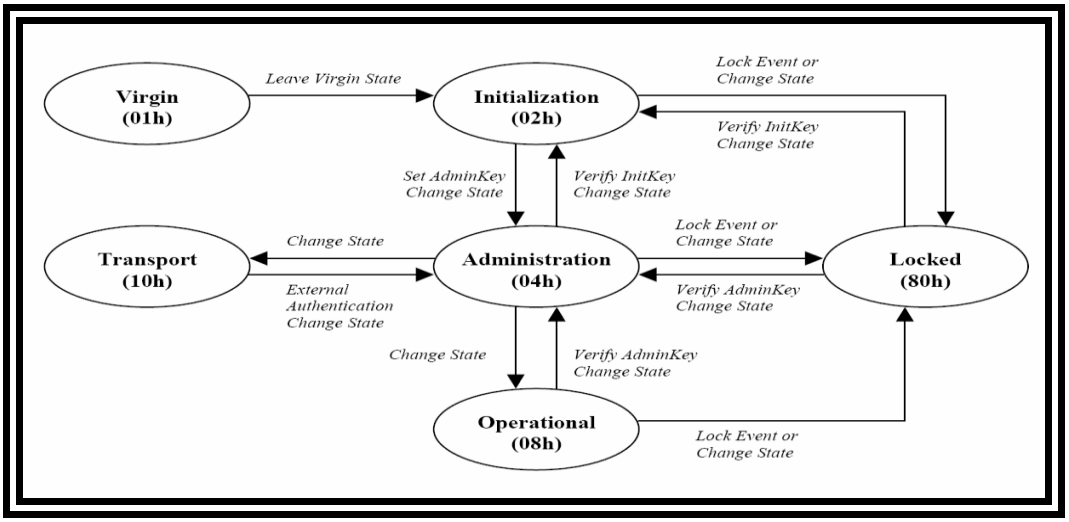

Figura 3: Ciclo de vida del Cryptomanager

1.5 Necesidad de la implementación de una aplicación Applet y Middleware

La introducción de la nueva Cédula de Identificación Electrónica en la República Bolivariana de Venezuela trae consigo que, instituciones del país se interesen en almacenar la información de los servicios que se brindan a los ciudadanos de la nación; por lo que se hace necesario incrementar el nivel de seguridad para acceder a la información que se encuentra almacenada dentro de la CIE, es por ello que la nueva cédula contará con mecanismos de autenticación biométrica, que permitan verificar la autenticidad del portador de dicha cédula. Por esta razón se hace necesario el diseño de una aplicación (Applet) que permita gestionar la información de las huellas dactilares de las personas, contenida dentro de la tarjeta electrónica, ya que la única solución existente en la actualidad (CryptoManager Applet) además de ser un software propietario, establece canal seguro, pero los datos viajan en texto claro, sin embargo, la mejor solución sería establecer canal seguro donde los datos viajen cifrados, ofreciendo de esta forma una mayor seguridad y protección a las operaciones realizadas con la información contenida en la tarjeta, evitando las modificaciones a los datos por medio de las intercepciones que se puedan realizar cuando se transmitan los mismos permitiendo que no lleguen a su destino adulterados pues solo pueden ser desencriptados por quien tenga la clave para hacerlo, además el diseño de la aplicación applet que se desarrollará brinda una solución a la República Bolivariana de Venezuela, aportándole ingresos y prestigio

al país. Conjuntamente se diseñará el componente middleware que permita la comunicación entre un ordenador y los distintos lectores de las tarjetas inteligentes.

1.6 Propuesta y selección de herramientas

La Cédula de Identidad Electrónica es una tarjeta de circuitos integrados, la cual tiene especificaciones estandarizadas a nivel internacional los cuáles deben tomarse en cuenta en el momento de realizar el diseño de la aplicación. Los estándares seleccionados para desarrollar la solución propuesta son:

- ISO/IEC 7816-4, que define cómo serán organizados los comandos que se enviarán a la tarjeta en ámbitos de seguridad, intercambio de datos, niveles de acceso a ficheros, entre otros.
- Global Platform, debido a su interoperabilidad en las tarjetas inteligentes, al permitir que sistemas heterogéneos puedan intercambiar procesos o datos.
- PC/SC por su parte, permitirá el acceso en las plataformas Windows y además permite a los desarrolladores trabajar con lectores de tarjetas de diferentes fabricantes.
- PKI: define un conjunto de aplicaciones y servicios que permite utilizar la criptografía de clave pública.

Para la gestión de la información biométrica contenida dentro de la CIE, se va a realizar la implementación de una aplicación (applet). El applet se desarrollará utilizando tecnología JavaCard, la cual permitirá implementar aplicaciones que se ejecutan dentro de las tarjetas inteligentes de modo que éstas tengan funcionalidades prácticas. Para implementar el Middleware se utilizará C#, como parte de la familia de los lenguajes de .Net, el cual brinda una gran versatilidad frente a estándares establecidos para los lectores de tarjetas inteligentes. Para interpretar las necesidades del sistema y diseñar la solución propuesta se utilizará la metodología ágil XP y se modelará además utilizando la herramienta UModel Altova, definidas las mismas por el departamento de Tarjetas Inteligentes, cumpliendo de esta forma con los requisitos que se hacen necesarios para el diseño de la actual solución para gestionar la información biométrica contenida en la Cédula de Identificación Electrónica de la República Bolivariana de Venezuela.

1.7 Conclusiones

- ✓ En este capítulo se definieron conceptualmente las aplicaciones Applet y Middleware según tecnología JavaCard mostrando la importancia que estas aplicaciones presentan para dar solución al problema planteado en el presente trabajo de diploma.
- ✓ Se describió el estado de las soluciones Applet y Middleware relacionadas con el almacenamiento de huellas dactilares en las tarjetas inteligentes, concluyendo que el CryptoMananger es la única solución encontrada que servirá de punto de partida para la realización del diseño del Applet como aplicación establecida para realizar la gestión de la información biométrica, contenida en la Cédula de Identificación Electrónica de la República Bolivariana de Venezuela, así como su componente Middleware, el cual será capaz de comunicarse con el Applet y realizarle consultas en el orden de la información que persista en la tarjeta.
- ✓ Además se desarrolló un profundo estudio, donde se identificaron metodologías, estándares y herramientas, el cual permitió elegir cuáles son las necesarias para el diseño de la solución propuesta, basado en las características propias de cada una, concluyendo que los estándares que son imprescindibles para la solución son: ISO/IEC 7816-4, Global Platform, PC/SC y PKI. La metodología que se utilizará para el desarrollo del software es XP y se modelará utilizando la herramienta UModel Altova. El Applet se desarrollará utilizando tecnología JavaCard y el Middleware empleando lenguaje C#.

Capítulo 2: Propuesta y análisis del Applet Secure Fingerprint Manager

2.1 Introducción

En este capítulo se utilizará la metodología Extreme Programming (XP) para el desarrollo ágil del software. El objetivo que se persigue con la elaboración del mismo es mostrar la evolución de la solución durante las fases iniciales de Exploración y Planificación donde se plantean a grandes rasgos las historias de usuario, 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 Modelo de Dominio

Actualmente Venezuela no cuenta con una infraestructura dedicada a los procesos de gestión de la información que se podrá contener en las Cédulas de Identificación Electrónica (CIE), por tal motivo se determina que no se pueden identificar los procesos del negocio que enmarcaran el problema a resolver. Sin embargo conceptos tecnológicos que definen correctamente cada eslabón de la solución se fueron observando, determinándose entonces la creación de un modelo de dominio, que evidencia claramente cómo funciona el entorno en el cual está enmarcado el problema. El modelo de dominio representa cosas del mundo real y para poder identificar los conceptos se hace necesario investigar el dominio del problema. (Proenza, 2005)

A continuación, se muestra el modelo de dominio que conceptualiza los elementos principales y sus relaciones entre sí.

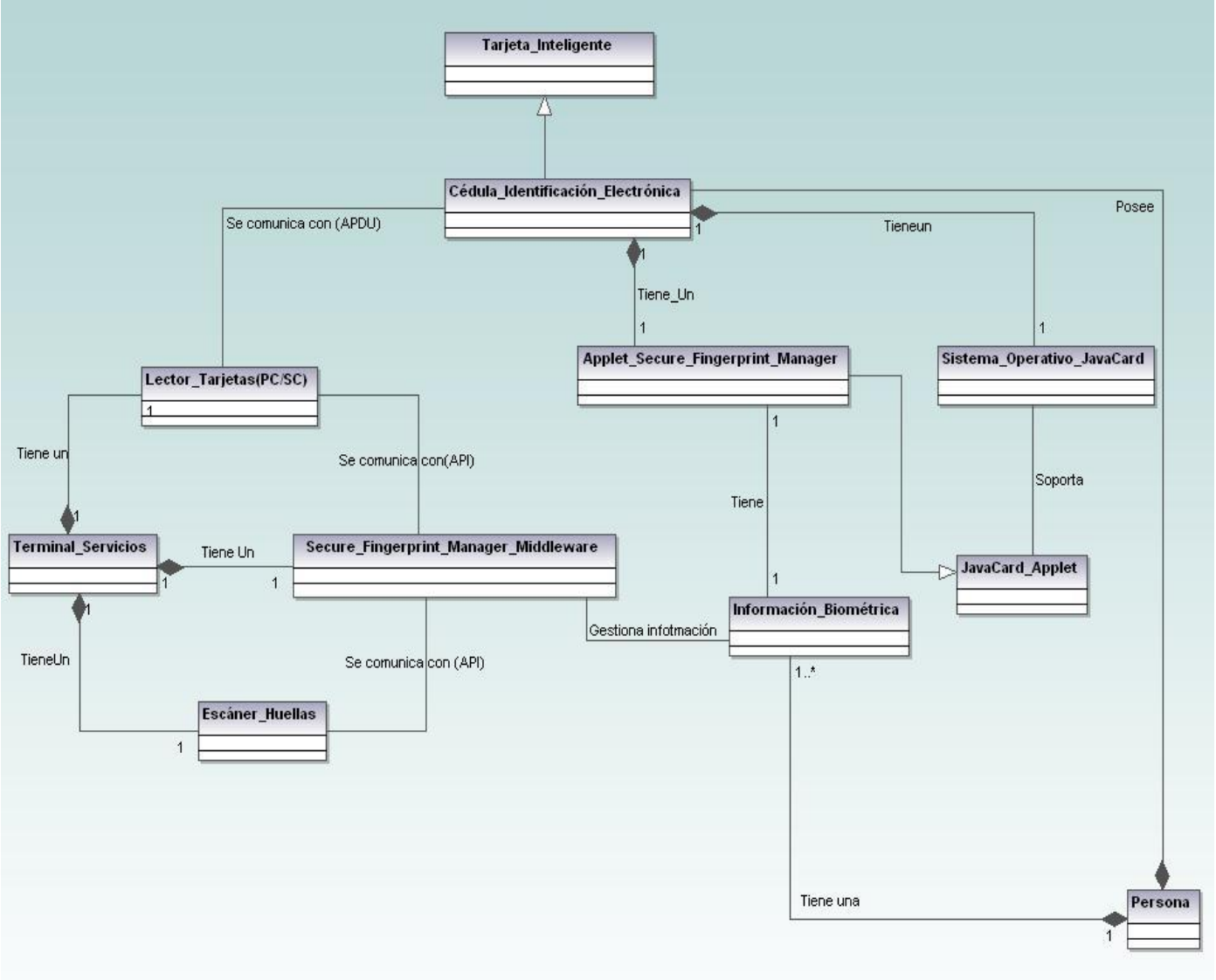

Figura 4: Diagrama de clases del modelo de dominio

2.2.1 Glosario de conceptos del modelo de dominio

2.2.1.1 Tarjeta_Inteligente

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

- Almacenar Información.
- Encriptar Información.
- Leer y escribir datos, similar a un ordenador.

2.2.1.2 Sistema_Operativo_JavaCard

La tecnología JavaCard combina parte del lenguaje de programación Java con un entorno de ejecución optimizado para tarjetas inteligentes y similares. El objetivo de la tecnología JavaCard es llevar los beneficios del desarrollo de software en Java al mundo de las tarjetas inteligentes.

2.2.1.3 Cédula_Identificación_Electrónica

Es una tarjeta inteligente que contiene tecnología JavaCard, la cual contiene datos de identificación de su portador, así como información referente a servicios que brindan distintas entidades externas a la ONIDEX en la República de Venezuela.

2.2.1.4 JavaCard_Applet

Los applets son las aplicaciones que corren embebidas en una JavaCard. Dichas aplicaciones interactúan en todo momento con el JCRE utilizando los servicios que éste brinda, e implementan la interfaz definida en la clase abstracta javacard.framework.Applet.

2.2.1.5 Applet_Secure_Fingerprint_Manager

Es una aplicación implementada en JavaCard, instalada dentro de la tarjeta inteligente la cual permite gestionar la información biométrica (huellas dactilares), que se almacene dentro de la CIE, referente a una persona.

2.2.1.6 Información_Biométrica

Son datos biométricos (huellas dactilares) referentes al portador de la CIE para su identificación en la República Bolivariana de Venezuela.

2.2.1.7 Secure_Fingerprint_Manager_ Middleware

Es un componente que funciona como capa de traducción entre el Applet Secure Fingerprint Manager y otro sistema, permitiendo una mejor comprensión de las respuestas obtenidas por la comunicación establecida con la aplicación instalada en la CIE (applet). Es el middleware encargado de gestionar todo un proceso de captación y verificación de las huellas dactilares de la persona portadora de la CIE.

2.2.1.8 Terminal_Servicios

Es un punto en una computadora donde se instalan todas las condiciones para poder interactuar con la solución para la verificación biométrica en tarjetas inteligentes de la República Bolivariana de Venezuela. Entre los elementos indispensables también podemos citar, los lectores PC/SC, escáner de huellas, entre otros.

2.2.1.9 Lectores_Tarjetas (PC/SC)

Es un lector compatible con el estándar PC/SC, el cual define la comunicación entre el Applet Secure Fingerprint Manager y el Middleware de Match on Card, sin realizarle modificaciones a ambas aplicaciones.

2.2.1.10 Escáner_Huellas

Es un dispositivo, el cual permite obtener las huellas dactilares que serán almacenadas en la CIE.

2.2.1.11 Persona

Portador de la Cédula de Identificación Electrónica en la República Bolivariana de Venezuela.

2.3 Historias de Usuario

Las historias de usuario son utilizadas en XP para especificar los requisitos del software (Ver Anexo 11: Lista de reserva del producto.) desde el punto de vista del cliente. Las escriben los propios clientes, tal y como ven ellos las necesidades del sistema, por tanto serán descripciones cortas y escritas en el lenguaje del usuario, sin terminología técnica. A continuación se presentan las historias de usuario identificadas para desarrollar la solución.

Historia de Usuario	
Número: HU_1	**Nombre de Historia de Usuario**: Inicializar comunicación.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Desarrollador	**Iteración asignada**: 1
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en desarrollo: Bajo	**Puntos reales:** 1
Descripción: Se muestran los lectores que están disponibles, se selecciona uno con el cual se va a establecer la comunicación con la Tarjeta Inteligente.	
Observaciones:	

Tabla 2. HU_1 Inicializar comunicación.

Historia de Usuario	
Número: HU_2	**Nombre de Historia de Usuario**: Establecer canal seguro del Applet Secure Fingerprint Manager.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Desarrollador	**Iteración asignada**: 1
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en desarrollo: Medio	**Puntos reales:** 1
Descripción: Se establece un canal de intercambio de información de forma segura entre el middleware y el Applet Secure Fingerprint Manager, utilizando Protocolo de Canal Seguro “01” según especificaciones de GlobalPlatform.	
Observaciones:	

Tabla 3.HU_2 Establecer canal seguro del Applet Secure Fingerprint Manager.

Historia de Usuario	
Número: HU_3	**Nombre de Historia de Usuario**: Obtener información biométrica.

Modificación de Historia de Usuario Número: Ninguna	
Usuario: Desarrollador	**Iteración asignada**: 1
Prioridad en negocio: Alta	**Puntos estimados:** 1.8
Riesgo en desarrollo: Bajo	**Puntos reales:** 1
Descripción: El Middleware realiza la petición de información biométrica al Applet Secure Fingerprint Manager, este último responde con los datos biométricos referente a la petición.	
Observaciones: El Applet Secure Fingerprint Manager debe contener datos biométricos.	

Tabla 4. HU_3 Obtener información biométrica.

	Historia de Usuario
Número: HU_4	**Nombre de Historia de Usuario**: Enrolar información biométrica.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Desarrollador	**Iteración asignada**: 1
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en desarrollo: Bajo	**Puntos reales:** 2
Descripción: El Middleware envía los datos biométricos a almacenar en el Applet Secure Fingerprint Manager, el applet almacena la información biométrica.	
Observaciones: El Applet Secure Fingerprint Manager debe estar personalizado.	

Tabla 5. HU_4 Enrolar información biométrica.

	Historia de Usuario
Número: HU_5	**Nombre de Historia de Usuario**: Realizar Match on Card.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Desarrollador	**Iteración asignada**: 1
Prioridad en negocio: Alta	**Puntos estimados:** 1.5
Riesgo en desarrollo: Medio	**Puntos reales:** 2
Descripción: El middleware envía los datos biométricos al Applet Secure Fingerprint Manager, el applet obtiene la información biométrica a verificar, efectúa la comparación y devuelve la respuesta de la misma.	
Observaciones:	

Tabla 6. HU_5 Realizar Match on Card.

	Historia de Usuario
Número: HU_6	**Nombre de Historia de Usuario**: Personalizar Applet Secure Fingerprint

	Manager.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Desarrollador	**Iteración asignada**: 1
Prioridad en negocio: Alta	**Puntos estimados:** 1.3
Riesgo en desarrollo: Medio	**Puntos reales:** 2.5
Descripción: El Middleware envía los datos para configurar y realizar la instanciación del Applet de Match on Card; el applet queda listo para almacenar la información biométrica.	
Observaciones:	

Tabla 7. HU_6 Personalizar Applet Secure Fingerprint Manager.

	Historia de Usuario
Número: HU_7	**Nombre de Historia de Usuario**: Gestionar almacenamiento de información.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Desarrollador	**Iteración asignada**: 1
Prioridad en negocio: Alta	**Puntos estimados:** 1
Riesgo en desarrollo: Medio	**Puntos reales:** 1.8
Descripción: Permitirá crear el estructura de ficheros en el Applet de Match on Card, donde se va almacenar la información biométrica y gestionar la seguridad para acceder a los datos almacenados.	
Observaciones:	

Tabla 8. HU_7 Gestionar almacenamiento de información.

	Historia de Usuario
Número: HU_8	**Nombre de Historia de Usuario**: Eliminar información biométrica.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Desarrollador	**Iteración asignada**: 2
Prioridad en negocio: Alta	**Puntos estimados:** 0.5
Riesgo en desarrollo: Bajo	**Puntos reales:** 0.9
Descripción: El middleware envía al Applet Secure Fingerprint Manager el comando que indica que se va a eliminar una de las plantillas contenedoras de información biométrica. El applet selecciona en uno de sus parámetros dicha plantilla, y la elimina.	

Observaciones: Si la ejecución del comando fue satisfactoria se envía un mensaje indicándole al usuario que todo fue bien. En caso de ocurrir algún error en la ejecución, el sistema muestra el error específico retornado por esta función.

Tabla 9. HU_8 Eliminar información biométrica.

	Historia de Usuario
Número: HU_9	**Nombre de Historia de Usuario**: Desbloquear información biométrica.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Desarrollador	**Iteración asignada**: 2
Prioridad en negocio: Alta	**Puntos estimados:** 0.4
Riesgo en desarrollo: Bajo	**Puntos reales:** 0.8
Descripción: El middleware envía al Applet Secure Fingerprint Manager el comando que indica que se va a desbloquear la plantilla. El applet selecciona en uno de sus parámetros la plantilla contenedora en la tarjeta que contiene la plantilla bloqueada y la desbloquea.	
Observaciones: Si la ejecución del comando fue satisfactoria se envía un mensaje indicándole al usuario que todo fue bien. En caso de ocurrir algún error en la ejecución, el sistema muestra el error específico retornado por esta función.	

Tabla 10. HU_9 Desbloquear información biométrica.

	Historia de Usuario
Número: HU_10	**Nombre de Historia de Usuario**: Restablecer estado de autenticación.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Desarrollador	**Iteración asignada**: 2
Prioridad en negocio: Alta	**Puntos estimados:** 0.6
Riesgo en desarrollo: Bajo	**Puntos reales:** 0.9
Descripción: El middleware envía al Applet Secure Fingerprint Manager el comando que indica que se va a restablecer el estado de autenticación. El applet restablece el indicador de estado de autenticación de la plantilla contenedora específica.	
Observaciones: Si la ejecución del comando fue satisfactoria se envía un mensaje indicándole al usuario que todo fue bien. En caso de ocurrir algún error en la ejecución, el sistema muestra el error específico retornado por esta función.	

Tabla 11. HU_10 Restablecer estado de autenticación.

	Historia de Usuario
Número: HU_11	**Nombre de Historia de Usuario**: Gestionar ID de la tarjeta.

Modificación de Historia de Usuario Número: Ninguna	
Usuario: Desarrollador	**Iteración asignada**: 2
Prioridad en negocio: Alta	**Puntos estimados:** 0.6
Riesgo en desarrollo: Bajo	**Puntos reales:** 0.9
Descripción: Esta funcionalidad presente dos estados posibles: • Modificar ID de la tarjeta • Obtener ID de la tarjeta El primero se encarga de actualizar el ID de la tarjeta. El área de datos de identificación se puede utilizar para almacenar datos o para identificar de forma única una tarjeta. El segundo estado se encarga de recuperar el ID de la tarjeta que contiene el Applet Secure Fingerprint Manager.	
Observaciones: Si la ejecución de ambos comandos fue satisfactoria se envía un mensaje indicándole al usuario que todo fue bien. En caso de ocurrir algún error en la ejecución, el sistema muestra el error específico retornado por esta función.	

Tabla 12. HU_11 Gestionar ID de la tarjeta.

	Historia de Usuario
Número: HU_12	**Nombre de Historia de Usuario**: Seleccionar applet.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Desarrollador	**Iteración asignada**: 2
Prioridad en negocio: Alta	**Puntos estimados:** 0.9
Riesgo en desarrollo: Bajo	**Puntos reales:** 1.1
Descripción: Primeramente se selecciona una instancia del Applet Secure Fingerprint Manager y una vez que la instancia del applet se ha seleccionado correctamente, los demás comandos disponibles pueden ser procesados de forma normal.	
Observaciones: Si la ejecución del comando fue satisfactoria se envía un mensaje indicándole al usuario que todo fue bien. En caso de ocurrir algún error en la ejecución, el sistema muestra el error específico retornado por esta función.	

Tabla 13.HU_12 Seleccionar A*pplet*.

	Historia de Usuario
Número: HU_13	**Nombre de Historia de Usuario**: Obtener propiedades.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Desarrollador	**Iteración asignada**: 3

Prioridad en negocio: Alta	**Puntos estimados:** 0.3
Riesgo en desarrollo: Bajo	**Puntos reales:** 0.5
Descripción: El middleware envía al *Applet* Secure Fingerprint Manager el comando que indica que se desea obtener las propiedades de dicho applet. Esta función recupera la información general acerca del applet que está corriendo en la tarjeta, junto con la información del estado de las diferentes plantillas biométricas.	
Observaciones: Si la ejecución del comando fue satisfactoria se envía un mensaje indicándole al usuario que todo fue bien. En caso de ocurrir algún error en la ejecución, el sistema muestra el error específico retornado por esta función.	

Tabla 14. HU_13 Obtener propiedades.

Historia de Usuario	
Número: HU_14	**Nombre de Historia de Usuario**: Gestionar estado.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Desarrollador	**Iteración asignada**: 3
Prioridad en negocio: Alta	**Puntos estimados:** 0.2
Riesgo en desarrollo: Bajo	**Puntos reales:** 0.4
Descripción: El middleware envía al Applet Secure Fingerprint Manager el comando que indica que se va a pasar de un estado a otro. Esta función cambia el estado actual del applet, permitiendo de esta forma que el applet pase al siguiente estado correspondiente.	
Observaciones: Si la ejecución del comando fue satisfactoria se envía un mensaje indicándole al usuario que todo fue bien. En caso de ocurrir algún error en la ejecución, el sistema muestra el error específico retornado por esta función.	

Tabla 15. HU_14 Gestionar estado.

Historia de Usuario	
Número: HU_15	**Nombre de Historia de Usuario**: Abandonar estado virgen.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Desarrollador	**Iteración asignada**: 3
Prioridad en negocio: Alta	**Puntos estimados:** 0.5
Riesgo en desarrollo: Bajo	**Puntos reales:** 1.2
Descripción: El middleware envía al Applet Secure Fingerprint Manager el comando que indica que se va a abandonar el estado virgen. Esta función pasa del estado virgen, retornando el valor de	

la llave de inicio en texto plano, de forma que esta se mantenga segura, y se procede a pasar al estado de inicialización.
Observaciones: Si la ejecución del comando fue satisfactoria se envía un mensaje indicándole al usuario que todo fue bien. En caso de ocurrir algún error en la ejecución, el sistema muestra el error específico retornado por esta función.

Tabla 16. HU_15 Abandonar estado virgen.

		Historia de Usuario
Número: HU_16	**Nombre de Historia de Usuario**: Finalizar comunicación.	
Modificación de Historia de Usuario Número: Ninguna		
Usuario: Desarrollador		**Iteración asignada**: 3
Prioridad en negocio: Alta		**Puntos estimados:** 0.3
Riesgo en desarrollo: Bajo		**Puntos reales:** 0.6
Descripción: Consiste en realizar la desconexión entre la Tarjeta Inteligente donde se encuentra el Applet Secure Fingerprint Manager, y el lector.		
Observaciones:		

Tabla 17. HU_16 Finalizar comunicación.

2.4 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. (Software, 2004)

- ✓ **Requisitos de hardware**
  - Tarjeta Inteligente (SmartCard) y escáner de huellas dactilares.
  - Lector de tarjetas incorporado a la PC que cumpla con el estándar PC/SC versión 1.0 ó superior.
  - La aplicación JavaCard debe ejecutar las APIs biométricas (BioMatch™ C o *Precise* BioMatch™ J).
- ✓ **Requisitos de software**
  - Se precisa de BioMatch™ Pro Toolkit 2.0 para generar y trabajar con las plantillas compatibles de BioMatch.
- ✓ **Usabilidad**

- El middleware debe ser de fácil utilización para lograr una mayor comodidad en su integración con aplicaciones existentes.

✓ **Rendimiento**

- El Applet debe ser capaz de realizar sus operaciones de manera eficiente, garantizando su funcionalidad en un corto intervalo de tiempo.

✓ **Soporte**

- Manual de usuarios. Sistema de ayuda. Manual de procedimientos.

✓ **Portabilidad**

- El middleware se debe desarrollar sobre una tecnología multiplataforma, que permita su utilización en distintos Sistemas Operativos.
- El middleware debe ser compatible con cualquier lector de tarjetas que cumpla con el estándar PC/SC.

✓ **Seguridad**

✓ **Confiabilidad**

- La aplicación debe recuperarse en el menor tiempo posible en caso de producirse una falla.
- La información almacenada en el Applet Secure Fingerprint Manager estará protegida de ataques externos a través de la seguridad que define el proveedor de tarjetas, su sistema operativo y la tecnología JavaCard.

✓ **Confidencialidad**

- La información biométrica almacenada dentro de la tarjeta, estará protegida de acceso no autorizado, mediante el uso de los mecanismos de seguridad y estándares definidos por GlobalPlatform.
- Los datos transmitidos y recibidos de la tarjeta, serán cifrados con criptografía simétrica, según define el estándar GlobalPlatform.

✓ **Integridad**

- La información contenida en la tarjeta, será objeto de cuidadosa protección contra la corrupción y estados inconsistentes.

✓ **Interfaz interna**

- Comunicación con lectores de tarjetas inteligentes.
- Comunicación con escáner de huella.
- Interfaz con el middleware de verificación biométrica.
- Interfaz con otras aplicaciones (API).

2.5 Metáfora

Una metáfora es una historia compartida que describe cómo debería funcionar el sistema. Es una manera sencilla de explicar el propósito del proyecto, y guiar la estructura y arquitectura del mismo. El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Martin Fowler en (Fowler, 2001) explica que la práctica de la metáfora consiste en formar un conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema. Este conjunto de nombres ayuda a la nomenclatura de clases y métodos del sistema. Debe tener suficiente contenido como para que sirva de guía a la arquitectura del proyecto. Partiendo de lo anterior surge como metáfora del sistema:

Con el diseño de la aplicación Applet Secure Fingerprint Manager se podrá desarrollar una aplicación que gestione la información biométrica contenida en la Cédula de Identificación Electrónica. Los datos biométricos asociados a una huella dactilar se almacenarán en un contenedor de plantilla (plantilla biométrica) para la gestión de los datos biométricos por medio de algoritmos y de las API biométricas presentes en la tarjeta. Se implementará además el componente middleware que será la capa de comunicación entre los lectores de tarjetas inteligentes y los ordenadores, para la transmisión de los comandos APDU hacia el applet que está corriendo en la tarjeta y recibir las respuestas de la misma mostrándolas en el ordenador en un formato entendible para el usuario.

2.6 Arquitectura

La solución propuesta presenta una arquitectura que refleja cómo va a funcionar la aplicación Applet Secure Fingerprint Manager, la cual controlará las funcionalidades biométricas presentes en la tarjeta, tales como:

- Proporcionar una API Externa, permitiendo que el portador de la tarjeta pueda utilizar la biometría presente en la misma.

- Gestionar datos biométricos, de acuerdo al algoritmo y a la actual API biométrica presentes en la tarjeta.

Además el Applet Secure Fingerprint Manager implementa una interfaz compartida que permitirá hacerle saber al mismo qué funcionalidades puede compartir con otros applets clientes, así como un API biométrico, a través del cual se establecerá la comunicación entre el algoritmo biométrico que se encuentra embebido dentro del sistema operativo de la tarjeta y el applet. Conjuntamente, todos los datos biométricos asociados a una huella dactilar específica se almacenarán en un contenedor de plantilla, dicha plantilla contendrá información pública acerca de la huella, tal como: conocer el dedo específico del cual ha sido extraída la huella, etc. Durante la instalación del applet un número configurable de contenedores de plantilla se asigna y se inicializan. El Applet Secure Fingerprint Manager admite un máximo de diez contenedores de plantilla, referidas a la información de cada dedo del portador de la CIE. La figura a continuación muestra como sería la arquitectura:

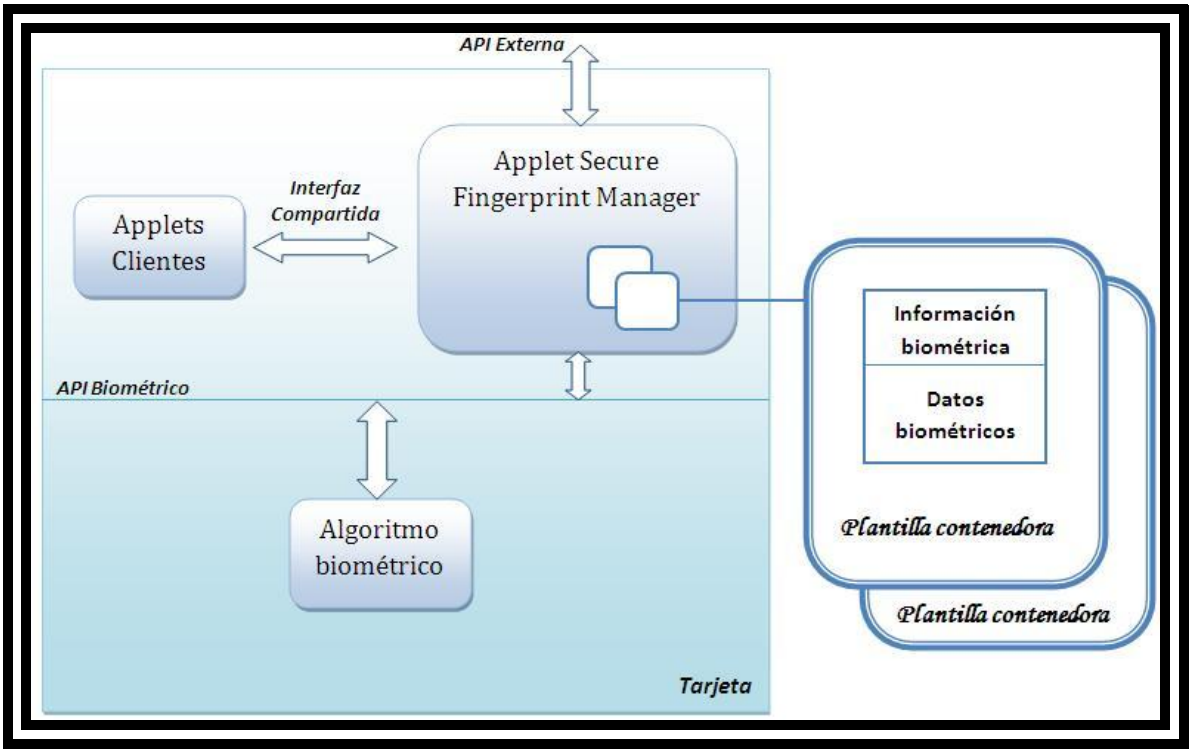

Figura 5: Arquitectura del Applet Secure Fingerprint Manager

2.7 Plan de entregas

Las historias de usuario servirán para crear el plan estimado de entrega. El plan de entregas se usará para crear los planes de iteración para cada iteración. Es en este momento cuando clientes y desarrolladores toman las decisiones comerciales y técnicas respectivamente. Con cada historia de usuario previamente evaluada en tiempo de desarrollo ideal, el cliente las agrupará en orden de importancia. De esta forma se puede trazar el plan de entregas en función de estos dos parámetros: tiempo de desarrollo ideal y grado de importancia para el cliente. Este artefacto se elabora con la intención de fijar qué período de tiempo puede tardar la implementación de cada una de las historias, definiéndose las fechas en que serán liberadas las versiones funcionales del producto. (Ver Anexo 1: Plan de entregas)

2.8 Estimación de tiempo

Los programadores estiman el tiempo necesario para desarrollar cada historia de usuario, este valor se expresa en semanas. Con el transcurso de las iteraciones, se irá acercando a la realidad. (Ver Anexo 2: Estimación de tiempo de las historias de usuario.)

2.9 Plan de iteraciones

Como parte del ciclo de vida de un proyecto usando la metodología XP se crea el plan de duración de cada una de las iteraciones que se han definido, que tiene como objetivo mostrar la duración y el orden en que serán implementadas las historias de usuario dentro de cada iteración. Para la solución se han definido 16 historias de usuario divididas en 3 iteraciones, de acuerdo a los intereses del cliente, para una duración total del proyecto de 22 semanas. (Ver Anexo 3: Plan de iteraciones.)

2.10 Estudio de factibilidad

Una de las tareas de mayor importancia en la planificación de proyectos de software es la estimación, la cual consiste en determinar, con cierto grado de certeza, los recursos de hardware y software, costo, tiempo y esfuerzo necesarios para el desarrollo de los mismos. Para lo cual se utilizó COCOMO II como modelo de estimación de costos. Este modelo permite realizar estimaciones en función del tamaño del software y de un conjunto de factores de costo y de escala. Los factores de costo describen aspectos

relacionados con la naturaleza del producto, hardware utilizado, personal involucrado y características propias del proyecto.

- **Estimación del esfuerzo**

COCOMO II está compuesto por tres modelos denominados: Composición de aplicación, Diseño temprano y Post-arquitectura, los cuales surgen en respuesta a la diversidad del mercado actual y futuro de desarrollo de *software*. El modelo *Diseño Temprano* se utiliza en las primeras etapas del desarrollo en las cuales se evalúan las alternativas de hardware y software de un proyecto, mientras que el modelo *Post-Arquitectura* se aplica en la etapa de desarrollo propiamente dicho, después que se define la arquitectura del sistema, y en la etapa de mantenimiento. Debido a las necesidades y características de la aplicación applet se ha utilizado una combinación de los dos últimos.

El modelo de Diseño Temprano ajusta el esfuerzo nominal usando siete factores de costo (Ver Anexo 4: Multiplicadores de esfuerzo). La fórmula para el cálculo del esfuerzo es la siguiente:

$$\text{PM}_{\text{estimado}} = \text{PM}_{\text{nominal}} \text{ X } \prod_{i=1}^{7} EMi$$

Donde:

$$\text{PM}_{\text{nominal}} = \text{A X } (KSLOC)^{B}$$

Donde:

PMestimado: es el esfuerzo nominal ajustado por 7 factores, que reflejan otros aspectos propios del proyecto que afectan al esfuerzo necesario para la ejecución del mismo, expresado en meses/personas.

KSLOC: es el tamaño del software a desarrollar expresado en miles de líneas de código fuente.

A: es una constante que captura los efectos lineales sobre el esfuerzo de acuerdo a la variación del tamaño, (**A= 2.94**).

B: es el factor exponencial de escala, toma en cuenta las características relacionadas con las economías y deseconomías de escala producidas cuando un proyecto de software incrementa su tamaño.

EMi: corresponde a los factores de costo que tienen un efecto multiplicativo sobre el esfuerzo, llamados Multiplicadores de esfuerzo.

Los modelos de estimación de costos frecuentemente tienen un factor exponencial para considerar las economías y deseconomías de escala. En particular, COCOMO II captura esos efectos en el exponente B y es calculado con la siguiente ecuación:

$$B = 1.01 + 0.01 \times \sum_{j=1}^{5} Wj$$

Si B < 1.0, el proyecto exhibe economía de escala.

Si el B = 1.0, las economías y deseconomías de escala están en equilibrio.

Si el B > 1.0, el proyecto muestra deseconomía de escala.

Un factor de escala de un proyecto, ***Wj***, se calcula sumando todos los factores y se usa para determinar el exponente de escala, ***B***.

Sustituyendo valores en la fórmula anterior queda que ***B* = 1.1575.**

- **Métricas:**

El modelo COCOMO II usa Puntos función y/o Líneas de código fuente (SLOC por sus siglas en inglés) como base para medir tamaño en el modelo de estimación de Diseño temprano. Los Puntos función procuran cuantificar la funcionalidad de un sistema de software. La meta es obtener un número que caracterice completamente al sistema. Son útiles estimadores ya que están basados en información que está disponible en las etapas tempranas del ciclo de vida del desarrollo de software.

- **Puntos función:**

La fórmula de Albretch (Albretch, 1979) para calcular los puntos función, es la siguiente:

FP = UFP X TCF

Donde:

FP: Puntos función

UFP: Puntos función no ajustados

TCF: Factor de complejidad técnica

Para calcular los UFP, se deben identificar los siguientes ítems: Entradas externas, Salidas externas, Archivo lógicos internos, Archivos externos de interfaces, Solicitudes externas.

Luego se clasifican de acuerdo al grado de complejidad en: bajo, promedio o alto. Se asigna un peso a cada uno según el tipo y el grado de complejidad correspondiente. (Ver Anexo 8: Tablas de estimación de los Puntos Función sin ajustar.) Finalmente los UFP son calculados mediante la sumatoria de los pesos de todos los ítems identificados.

$$UFP = \sum_{i=1}^{15} (Cantidad\_Items\_Tipo_i) \; X \; (Peso)$$

La siguiente tabla muestra como se determinan los niveles de complejidad de cada tipo de ítem en función del número y tipo de elementos de datos y archivos involucrados.

Para archivos lógicos internos y archivos externos de interfase				Para salidas y consultas externas				Para entradas externas			
Elementos de Registro	Elementos de datos			Tipos de archivos	Elementos de datos			Tipos de archivos	Elementos de datos		
	1-19	20-50	51+		1-5	6-19	20+		1-4	5-15	16+
1	Bajo	Bajo	Prom.	0 ó 1	Bajo	Bajo	Prom.	0 ó 1	Bajo	Bajo	Prom.
2-5	Bajo	Prom.	Alto	2-3	Bajo	Prom.	Alto	2-3	Bajo	Prom.	Alto
6+	Prom.	Alto	Alto	4+	Prom.	Alto	Alto	3+	Prom.	Alto	Alto

Figura 6: Tabla de Puntos Función. Determinación del peso. (Boehm, 1995/2)

Tipo de función	Peso del Factor de Complejidad		
	Bajo	Promedio	Alto
Entradas Externas (Inputs)	3	4	6
Salidas Externas (Outputs)	4	5	7
Archivo Lógicos Internos (Archivos)	7	10	15
Archivos Externos de Interfase (Interfases)	5	7	10
Consultas Externas (Queries)	3	4	6

Figura 7: Tabla de peso del factor de complejidad. (Boehm, 1995/2)

Despejando de la fórmula original, anteriormente planteada, y teniendo en cuenta la determinación del peso de factor de complejidad y la tabla de Puntos Función se tiene un total de 21 de Puntos función sin ajustar (UFP).

Para el cálculo del Factor de complejidad técnica, TCF, se considera la siguiente fórmula:

$$TCF = 0.65 + 0.01 \times \sum_{i=1}^{14} Fi$$

Donde los ***Fi*** corresponden a los pesos asignados a diferentes factores. (Ver Anexo 9: Pesos asignados a factores.)

- **Líneas de código**

El tamaño de una aplicación se mide en unidades de líneas de código fuente (KSLOC), este valor en COCOMO II puede estimarse a partir de Puntos función sin ajustar convirtiendo a SLOC y luego dividiendo por 1000. (Ver Anexo 10: Cálculo de líneas de código.)

Una vez calculados los elementos necesarios, se procede al cálculo del Esfuerzo nominal del proyecto con la fórmula anteriormente planteada, quedando que:

1.1575

PMnominal = 2.94 x 1.550

PMnominal = 4.88 meses/persona

Derivándose del resultado anterior se encuentra el esfuerzo estimado del proyecto, quedando:

PMestimado = 4.88 x 0.77

PMestimado = 3,76 meses/persona

- **Estimación del cronograma**

$$TDEV = \left[3.0 \times PM_{*}^{(0.33+0.2\times(B-1.01))}\right] \times \frac{SCED\%}{100}$$

Donde:

TDEV es el tiempo calendario en meses que transcurre desde la determinación de los requerimientos a la culminación de una actividad que certifique que el producto cumple con las especificaciones.

PM* es el esfuerzo expresado en meses personas, calculado sin tener en cuenta el multiplicador de esfuerzo SCED.

B es el Factor de Escala

SCED% es el porcentaje de compresión/expansión del cronograma.

Aplicando la fórmula anterior queda que:

(0.33+0.2x (1.1575-1.01))

$$\left[TDEV = 3.0 \times 3.76 \right] \times \frac{100\%}{100}$$

***TDEV* = 4.83 meses**

El tiempo y esfuerzo de desarrollo se consideran mucho más próximos a la realidad al utilizar el modelo COCOMO II, se destaca el hecho de que para proyectos pequeños resulta efectivo el lograr un prototipo rápido como técnica de cumplimiento a los requisitos, que si bien al inicio demanda de un esfuerzo y tiempo adicional, esto se ve claramente compensado al momento de desarrollar el producto final.

- **Costos y beneficios**

El desarrollo de un producto siempre trae aparejado un costo de producción, el cual debe ser justificado de acuerdo con los beneficios que el mismo reporta. La solución propuesta no incurre en grandes gastos, y basándonos en los resultados de los cálculos anteriormente planteados, se concluye que su desarrollo es factible.

La aplicación Applet Secure Fingerprint Manager no se ha concebido con fines comerciales inmediatos, pero es un software independiente, que puede comercializarse en el futuro, reportando beneficios monetarios al Centro de Identificación y Seguridad Digital y en general a la Universidad de Ciencias Informáticas.

2.11 Conclusiones

- ✓ En el capítulo concluyente se determinaron las bases del por qué la elección de un modelo de dominio, concluyendo que el mismo brinda 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 obtenidos fueron las historias de usuario y la lista de reserva del producto, lográndose una visión clara y objetiva de los requerimientos del cliente.
- ✓ Se realizó un estudio de factibilidad para estimar el esfuerzo y el tiempo que tomaría la realización de la solución, además de analizarse los gastos en que se incurrirá y los beneficios que reportará la implementación de la misma, llegando a la conclusión de que es factible su desarrollo.

Capítulo 3: Diseño del *Applet Secure Fingerprint Manager*

3.1 Introducción

En el presente capítulo se desarrollan los artefactos relacionados con la fase de *Iteraciones* donde todo el trabajo de la iteración es expresado en tareas de ingeniería. Se realiza además el diseño de la solución y los diagramas de diseño correspondientes a esta fase.

3.2 Tareas de Ingeniería

Todo el trabajo de la iteración es expresado en tareas de ingeniería, las cuales se realizan para especificar las acciones llevadas a cabo por los programadores en cada historia de usuario, pues estas 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. A continuación se muestran las tareas de ingeniería derivadas de cada historia de usuario.

Iteración	Historia de Usuario	Tarea
1	Inicializar comunicación	✓ Mostrar lectores disponibles. ✓ Seleccionar lector. ✓ Establecer comunicación con la tarjeta inteligente.
	Establecer canal seguro del *Applet Secure Fingerprint Manager*	✓ Establecer canal seguro.
	Obtener información biométrica	✓ Seleccionar plantilla biométrica específica. ✓ Devolver información descriptiva sobre los datos biométricos.
	Enrolar información biométrica	✓ Comprobar que se encuentre en estado de administración. ✓ Seleccionar la plantilla biométrica. ✓ Almacenar los datos biométricos en la plantilla seleccionada.

	Realizar Match on Card	✓ Seleccionar la plantilla biométrica. ✓ Realizar verificación.
	Personalizar el Applet Secure Fingerprint Manager	✓ Enviar datos de configuración
	Gestionar almacenamiento de información	✓ Seleccionar plantilla biométrica. ✓ Eliminar plantilla biométrica.
2	Eliminar información biométrica	✓ Seleccionar plantilla ✓ Eliminar información biométrica
	Desbloquear información biométrica	✓ Seleccionar plantilla ✓ Desbloquear información biométrica
	Restablecer estado de autenticación	✓ Seleccionar plantilla ✓ Restablecer estado de autenticación
	Gestionar ID de la tarjeta	✓ Modificar ID de la tarjeta ✓ Obtener ID de la tarjeta
	Seleccionar Applet	✓ Seleccionar instancia del Applet Secure Fingerprint Manager
3	Obtener propiedades	✓ Obtener información de Applet ✓ Obtener información del estado de las plantillas biométricas
	Gestionar estado	✓ Cambiar estado
	Abandonar estado virgen	✓ Retornar llave de inicio ✓ Pasar a estado de inicialización
	Finalizar comunicación	✓ Desconectar la tarjeta del lector

Tabla 18. Distribución de las tareas de ingeniería por iteraciones

A continuación las especificaciones de cada tarea de ingeniería:

	Tareas de ingeniería
Número de tarea: HU1_T1	**Historia de usuario:** Inicializar comunicación.
Nombre: Mostrar lectores disponibles.	
Tipo: Desarrollo	**Puntos Estimados:**
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: Accediendo a la Winscard.dll se obtiene la lista de lectores conectados al ordenador del usuario, mostrándole una lista desplegable con los nombre de los lectores disponibles para que escoja uno.	

Tabla 19. HU1_T1 Mostrar lectores disponibles

	Tareas de ingeniería
Número de tarea: HU1_T3	**Historia de usuario:** Inicializar comunicación.
Nombre: Establecer comunicación con la tarjeta inteligente.	
Tipo: Desarrollo	**Puntos Estimados:**
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: Inicialmente cuando se carga la página, se listan los lectores conectados y el sistema intenta establecer conexión con la tarjeta si está insertada, sino luego que el usuario la introduzca debe hacerlo manualmente pulsando un botón para iniciar la conexión.	

Tabla 20. HU1_T2 Establecer comunicación con la tarjeta inteligente.

	Tareas de ingeniería
Número de tarea: HU2_T1	**Historia de usuario:** Establecer canal seguro del Applet Secure Fingerprint Manager
Nombre: Establecer canal seguro.	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: Utilizando Protocolo de Canal Seguro “01” según especificaciones de GlobalPlatform.	

Tabla 21. HU2_T1 Establecer canal seguro

Tareas de ingeniería

Número de tarea: HU3_T1	**Historia de usuario:** Obtener información biométrica
Nombre: Seleccionar plantilla biométrica específica.	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: En la clase SFMApplet se encuentra la lista de plantillas biométricas la cual se recorrerá hasta encontrar el número de plantilla que se específica en el comando APDU enviado.	

Tabla 22. HU3_T1 Seleccionar plantilla biométrica

	Tareas de ingeniería
Número de tarea: HU3_T2	**Historia de usuario:** Obtener información biométrica
Nombre: Devolver información descriptiva sobre los datos biométricos.	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: Se devolverá los datos biométricos asociados a la plantilla seleccionada. Si el número de plantilla especificado no existe se debe lanzar un error.	

Tabla 23. HU3_T2 Devolver información descriptiva sobre los datos biométricos

	Tareas de ingeniería
Número de tarea: HU4_T1	**Historia de usuario:** Enrolar información biométrica.
Nombre: Comprobar que se encuentre en estado de administración.	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: En la clase SFMApplet al consultar el atributo _SFMState se obtendría el estado del SFMApplet y se comprobaría si está en estado de administración.	

Tabla 24. HU4_T1 Comprobar que se encuentre en estado de administración

	Tareas de ingeniería
Número de tarea: HU4_T2	**Historia de usuario:** Enrolar información biométrica.
Nombre: Seleccionar plantilla biométrica.	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: En la clase SFMApplet se encuentra la lista de plantillas biométricas la cual se recorrerá hasta encontrar el número de plantilla que se específica en el comando APDU enviado.	

Tabla 25. HU4_T2 Seleccionar plantilla biométrica

	Tareas de ingeniería
Número de tarea: HU4_T3	**Historia de usuario:** Enrolar información biométrica.
Nombre: Almacenar datos biométricos en la plantilla biométrica.	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: Almacenar en la clase ContainerBiometricData la información biométrica enviada en el comando APDU. Si el número de plantilla especificado no existe se debe lanzar un error.	

Tabla 26. HU4_T3 Almacenar datos biométricos en la plantilla biométrica

	Tareas de ingeniería
Número de tarea: HU5_T1	**Historia de usuario:** Realizar MoC
Nombre: Seleccionar la plantilla biométrica.	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: En la clase SFMApplet se encuentra la lista de plantillas biométricas la cual se recorrerá hasta encontrar el número de plantilla que se específica en el comando APDU enviado.	

Tabla 27. HU5_T1 Seleccionar plantilla biométrica

	Tareas de ingeniería
Número de tarea: HU5_T2	**Historia de usuario:** Realizar MoC
Nombre: Realizar verificación.	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: En la clase SFMApplet se comprueba que la información biométrica almacenada en la tarjeta específicamente en la clase ContainerBiometricData corresponda con la enviada en el comando APDU.	

Tabla 28. HU5_T2 Realizar verificación

	Tareas de ingeniería
Número de tarea: HU6_T1	**Historia de usuario:** Personalizar el Applet Secure Fingerprint Manager
Nombre: Enviar datos de configuración	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: Se envía un comando APDU con los datos de la configuración	

Tabla 29. HU6_T1 Enviar datos de configuración

	Tareas de ingeniería
Número de tarea: HU7_T1	**Historia de usuario:** Gestionar almacenamiento de información
Nombre: Seleccionar plantilla biométrica.	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: En la clase SFMApplet se encuentra la lista de plantillas biométricas la cual se recorrerá hasta encontrar el número de plantilla que se específica en el comando APDU enviado.	

Tabla 30. HU7_T1 Seleccionar plantilla biométrica

	Tareas de ingeniería
Número de tarea: HU7_T2	**Historia de usuario:** Gestionar

	almacenamiento de información
Nombre: Eliminar plantilla biométrica.	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: Se selecciona la plantilla biométrica en la lista de plantillas de la clase SFMApplet por el número de plantilla enviada en el comando APDU y se procede a eliminarla de la lista.	

Tabla 31. HU7_T2 Eliminar plantilla biométrica

Tareas de ingeniería	
Número de tarea: HU8_T1	**Historia de usuario:** Eliminar información biométrica
Nombre: Seleccionar plantilla biométrica.	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: En la clase SFMApplet se encuentra la lista de plantillas biométricas la cual se recorrerá hasta encontrar el número de plantilla que se específica en el comando APDU enviado.	

Tabla 32. HU7_T2 Seleccionar plantilla biométrica

Tareas de ingeniería	
Número de tarea: HU8_T2	**Historia de usuario:** Eliminar información biométrica
Nombre: Eliminar información biométrica.	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: Se selecciona la plantilla biométrica en la lista de plantillas de la clase SFMApplet por el número de plantilla enviada en el comando APDU y se procede a eliminarla la información biométrica que ésta contiene.	

Tabla 33. HU8_T2 Eliminar plantilla biométrica

Tareas de ingeniería

Número de tarea: HU9_T1	**Historia de usuario:** Desbloquear información biométrica.
Nombre: Seleccionar plantilla biométrica.	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: En la clase SFMApplet se encuentra la lista de plantillas biométricas la cual se recorrerá hasta encontrar el número de plantilla que se específica en el comando APDU enviado.	

Tabla 34. HU9_T1 Seleccionar plantilla biométrica

	Tareas de ingeniería
Número de tarea: HU9_T2	**Historia de usuario:** Desbloquear información biométrica.
Nombre: Desbloquear información biométrica.	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: Se debe validar la AdminKey y proceder a la administración del Estado donde las plantillas biométricas pueden ser desbloqueadas, y luego volver al estado operativo donde la plantilla puede volver a utilizarse. Si el AdminKey está bloqueado, entonces, la única opción es validar la StartKey y proceder a la inicialización del estado.	

Tabla 35. HU9_T2 Desbloquear información biométrica

	Tareas de ingeniería
Número de tarea: HU10_T1	**Historia de usuario:** Restablecer estado de autenticación
Nombre: Seleccionar plantilla biométrica	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: En la clase SFMApplet se encuentra la lista de plantillas biométricas la cual se recorrerá hasta encontrar el número de plantilla que se específica en el comando APDU enviado.	

Tabla 36. HU10_T2 Seleccionar plantilla biométrica

	Tareas de ingeniería
Número de tarea: HU10_T2	**Historia de usuario:** Restablecer estado de autenticación
Nombre: Restablecer estado de autenticación	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: Se debe validar la AdminKey y proceder a la administración del estado donde las plantillas biométricas pueden ser desbloqueadas, y luego volver al estado operativo donde la plantilla puede volver a utilizarse. Si el AdminKey está bloqueado, entonces, la única opción es validar la StartKey y proceder a la inicialización del estado.	

Tabla 37. HU10_T2 Restablecer estado de autenticación

	Tareas de ingeniería
Número de tarea: HU11_T1	**Historia de usuario:** Gestionar ID tarjeta
Nombre: Modificar ID de la tarjeta	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: Almacenar los datos abiertos que se puede leer en todos los estados, se debe comprobar que la longitud del ID sea válido y que la variable de estado SFMState esté en Inicialization.	

Tabla 38. HU11_T1 Modificar ID de la tarjeta

	Tareas de ingeniería
Número de tarea: HU11_T2	**Historia de usuario:** Gestionar ID tarjeta
Nombre: Obtener ID de la tarjeta	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: Recuperar la tarjeta de identificación del Applet Secure FingerPrint Manager Teniendo en cuenta que solo un máximo de 20 bytes está disponible. Se debe comprobar que la longitud del ID sea válido y que la variable de estado SFMState no sea en estado Virgin ni Transport.	

Tabla 39. HU11_T2 Obtener ID de la tarjeta

	Tareas de ingeniería
Número de tarea: HU12_T1	**Historia de usuario:** Seleccionar Applet
Nombre: Seleccionar instancia del Applet Secure Fingerprint Manager	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: Crear un objeto de SFMApplet.	

Tabla 40. HU12_T1 Seleccionar instancia del Applet Secure Fingerprint Manager

	Tareas de ingeniería
Número de tarea: HU13_T1	**Historia de usuario:** Obtener propiedades
Nombre: Obtener información del estado de las plantillas biométricas	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: Cantidad de plantillas bloqueadas, cantidad de intentos antes de que se bloquee accediendo a los atributos de la clase ContainerBiometricData, _TryCounter y _MaxTryCounter.	

Tabla 41. HU13_T1 Obtener información del estado de las plantillas biométricas

	Tareas de ingeniería
Número de tarea: HU13_T2	**Historia de usuario:** Obtener propiedades
Nombre: Obtener información de *Applet*	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: Accediendo a las variables _SFMState y _SFMKeyStore se pueden obtener las propiedades del Applet.	

Tabla 42. HU13_T2 Obtener información del Applet

	Tareas de ingeniería
Número de tarea: HU14_T1	**Historia de usuario:** Gestionar estado
Nombre: Cambiar estado	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**

Responsable: Yarisel Aguilera Nieblas
Descripción: Accediendo a la clase SFMApplet, se procede a cambiar el atributo _SFMState.

Tabla 43. HU14_T1 Cambiar estado

	Tareas de ingeniería
Número de tarea: HU15_T1	**Historia de usuario:** Abandonar estado virgen
Nombre: Retornar llave de inicio	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: La StartKey debe ser creada utilizando una funcionalidad de la clase RandomData y enviado a un host para un guardado seguro.	

Tabla 44. HU15_T1 Retornar llave de inicio

	Tareas de ingeniería
Número de tarea: HU15_T2	**Historia de usuario:** Abandonar estado virgen
Nombre: Pasar a estado de inicialización	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: Se debe acceder a la llave de inicio o StartKey para entrar en el estado de inicialización donde se procede a limpiar todas las variables que se usen en el SFMApplet.	

Tabla 45. HU15_T2 Pasar a estado de inicialización

	Tareas de ingeniería
Número de tarea: HU16_T1	**Historia de usuario:** Finalizar comunicación
Nombre: Desconectar la tarjeta del lector	
Tipo: Desarrollo	**Puntos Estimados:**2
Fecha inicio:	**Fecha fin:**
Responsable: Yarisel Aguilera Nieblas	
Descripción: La conexión con la tarjeta puede ser cerrada en cualquiera de estos escenarios: • Si el usuario saca la tarjeta del lector. • Si el usuario desconecta el lector de la computadora.	

Tabla 46. HU16_T1 Desconectar la tarjeta del lector

3.3 Diseño de la solución

Después de desglosar las historias de usuario en tareas de ingeniería, se hace necesario crear sesiones con el equipo de trabajo donde se presentarán las tarjetas CRC (clase, responsabilidad, colaboración), que darán la idea de la cantidad de clases a implementar y las responsabilidades que estas tendrán. XP propone la puesta en práctica de ciertos principios a la hora de realizar estas sesiones de trabajo, para garantizar la agilidad en el proceso de desarrollo:

Según Wells, J. Donovan (Wells, 1999) estos principios son:

- **Simplicidad:** Hay que conseguir diseños simples y sencillos. Procurar hacer todo lo menos complicado posible para entender e implementar el diseño, esto a la larga costará menos tiempo y esfuerzo.
- **Uso de tarjetas CRC:** (clase, responsabilidad, colaboración), estas tarjetas son manejadas por el equipo de desarrollo durante la codificación de la solución; generalmente cada tarjeta representa una clase diferente en la codificación y tienen como ventaja que todo el equipo contribuye a la elaboración del diseño de la solución.
- **No adicionar funcionalidades tempranamente:** No añadir funcionalidad extra al programa aunque se piense que en un futuro será utilizada. Sólo el 10% de la misma es utilizada lo que demuestra que el desarrollo de funcionalidad extra es un desperdicio de tiempo y recursos.

3.3.1 Definiciones de las tarjetas CRC

Las tarjetas CRC determinan el comportamiento de cada actividad. La aplicación Applet Secure Fingerprint Manager está formada por las siguientes clases:

1. SFMApplet
2. ContainerBiometricData
3. SFMState
4. List
5. SFMAPDUInterpreter
6. SFMKeyStore
7. SFMGlobalPlatformService
8. Item
9. ProxyBiometricAPI

Clase: SFMState **Responsabilidades**	**Clases Relacionadas**
Definir estados de SFMApplet (Virgen, Inicialización, Transporte, Bloqueado, Operacional).	SFMApplet

Tabla 47: Tarjeta CRC SFMState

Clase: ContainerBiometricData **Responsabilidades**	**Clases Relacionadas**
Almacenar las plantillas biométricas	SFMApplet

Tabla 48: Tarjetas CRC CaontainerBiometricData

Clase: SFMKeyStore **Responsabilidades**	**Clases Relacionadas**
Almacenar las llaves de inicio y de administración	SFMApplet

Tabla 49: Tarjetas CRC SFMKeyStore

Clase: List **Responsabilidades**	**Clases Relacionadas**
Almacenar la información biométrica de las huellas dactilares	SFMApplet

Tabla 50: Tarjetas CRC List

Clase: Item **Responsabilidades**	**Clases Relacionadas**
Almacenar la información del elemento cabecera de la lista	List

Tabla 51: Tarjetas CRC Item

Clase: ProxyBiometricAPI **Responsabilidades**	**Clases Relacionadas**
Realizar el MoC de la información biométrica	SFMAPDUInterpreter

Tabla 52: Tarjeta CRC ProxyBiometricAPI

Clase: SFMAPDUInterpreter **Responsabilidades**	**Clases Relacionadas**
Interpretar los comandos APDU enviados al SFMApplet	SFMApplet

Tabla 53: Tarjetas CRC SFMAPDUInterpreter

Clase: SFMGlobalPlatformService **Responsabilidades**	**Clases Relacionadas**
Establecer canal seguro	SFMApplet

Tabla 54:Tarjetas CRC SFMGlobalPlatformService

Clase: SFMApplet **Responsabilidades**	**Clases Relacionadas**
Almacenar información biométrica	ContainerBiometricData
Verificar información biométrica	ContainerBiometricData
Devolver información general del SFMApplet	SFMApplet,SFMKeyStore
Cambiar el estado del SFMApplet	SFMState
Desbloquear la información biométrica	ContainerBiometricData
Abandonar estado virgen	SFMState

Tabla 55: Tarjetas CRC SFMState

Además para una mayor especificación de las clases, sus relaciones y responsabilidades se ha desarrollado un diagrama de clases Ver (Anexo 12: Diagrama de Clases.)

3.3.2 Descripción del diseño

El modelo de diseño describe la realización física de las historias de usuario, centrándose en los requisitos funcionales y no funcionales; junto con otras restricciones relacionadas con el entorno de implementación que tienen impacto en el sistema a considerar. El modelo de diseño sirve de abstracción a la implementación. El mismo puede contener: los diagramas, relaciones, colaboraciones, atributos, las realizaciones de las tareas de ingeniería, entre otros diagramas que se puedan considerar para el sistema en desarrollo.

A continuación se hace una breve descripción de los principales componentes del middleware para entender el funcionamiento de la aplicación:

SmartCard .Net Framework: es un paquete contenedor de los wrapper WinPCSC y Mono PCSCLite.

WinPCSC Wrapper: es la solución para la ejecución de aplicaciones de SmartCard que cumplan las especificaciones PC/SC para el sistema Operativo Windows.

Microsoft.Net Framework: contiene todas las librerías bases que provee la tecnología .Net.

Winscard.dl: permite la comunicación con cualquier equipo que cumpla con las especificaciones PC/SC.

MonoPCSCLite: es la solución para la ejecución de aplicaciones de SmartCard que cumplan las especificaciones PC/SC para software libre, este wrapper incluye todas las características para ser ejecutado bajo ambiente Linux o Windows.

Mono.Net Framework: utiliza la biblioteca <<winscard.dll>>para implementar PC/SC para Windows y de la librería <<libpcsclite.so>> librería para implementar PC/SC para entorno Linux.

SmartCardFramework: framework compuesto por un conjunto de DLLs que permiten a través de su uso la comunicación con la tarjeta inteligente. Define cómo se deben implementar los middlewares para que puedan ser integrados a esta plataforma y que además ya provee algunos middlewares estándares para la comunicación segura con tarjetas.

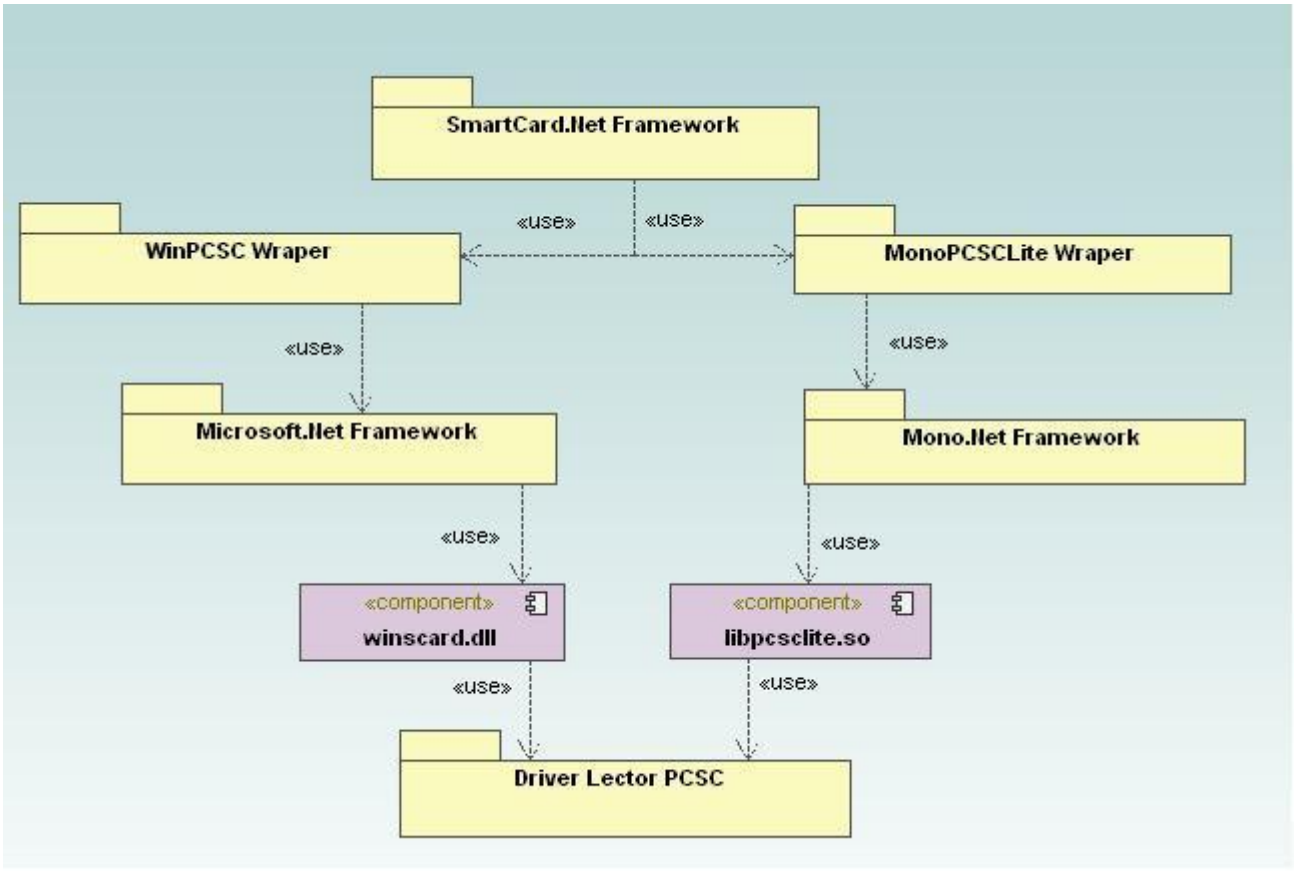

Figura 8: Diagrama de Componentes del Middleware

3.3.3 Descripción del flujo de proceso: "Verificar autenticidad del portador de la CIE"

El proceso más importante que debe ejecutar la aplicación es el de verificar que la información biométrica contenida en la aplicación Applet Secure FingerPrint Manager corresponda con la introducida por el portador de Cédula de Identificación Electrónica. A continuación una breve descripción:

- La Terminal de Servicio de la entidad externa a través del Secure FingerPrint Manager Middleware solicita la captura de la huella dactilar del portador de la CIE.
- El Secure FingerPrint Manager Middleware obtiene las minucias de la huella a través del Middleware de Verificación Biométrica.
- El Secure FingerPrint Manager Middleware envía las minucias al Applet Secure Fingerprint Manager en la CIE, para realizar la comparación.

  - Si la comparación de las minucias es correcta se notifica que la autenticación ha sido comprobada.

  - Si la comparación de las minucias es incorrecta se notifica que la comparación ha sido fallida por lo tanto la autenticación es incorrecta.

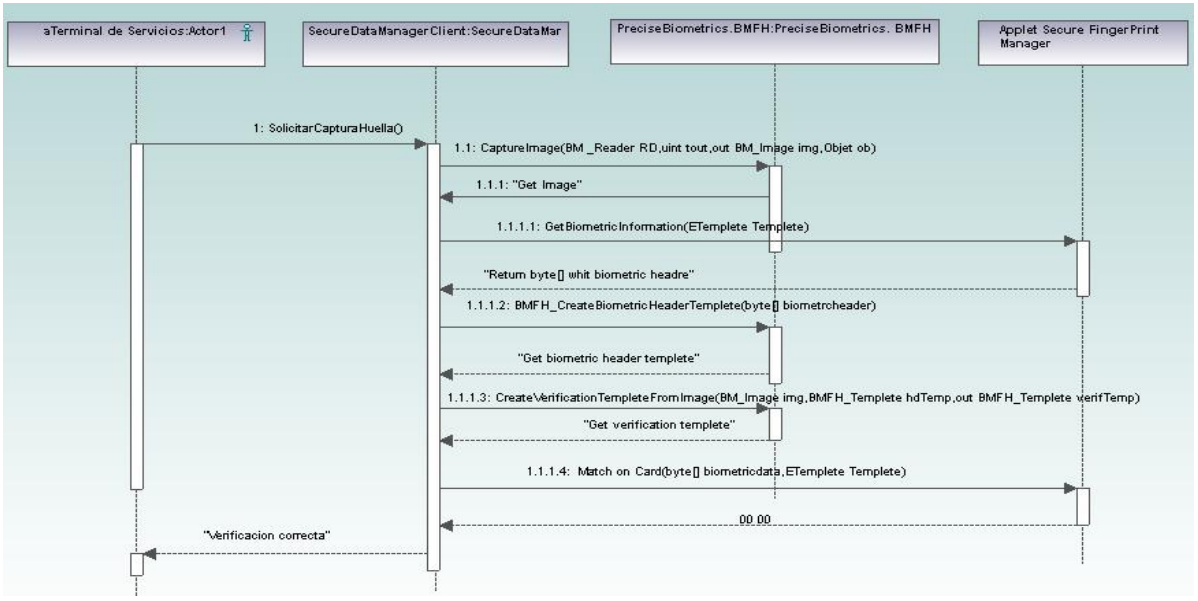

Figura 9: Diagrama de Secuencia "Verificar autenticidad del portador de la CIE"

3.4 Conclusiones

- ✓ El desglose de las historias de usuario en tareas de la ingeniería es una buena práctica que mostrará a los programadores las funcionalidades específicas que se deben implementar.
- ✓ La realización de los diferentes diagramas de ingeniería específicamente el diagrama de componentes brinda una solución multiplataforma en la cual, tanto en sistemas operativos propietarios como libres sea capaz de ejecutarse el Middleware y utilizar los componentes y API necesarios para la comunicación con los lectores de las tarjetas inteligentes.
- ✓ El diagrama de secuencia describe el funcionamiento del proceso Validar Autenticación brindando instrucciones para su futura implementación.

CONCLUSIONES GENERALES

Luego del estudio realizado se logró el análisis y el diseño del sistema que servirá como base para futuras labores de implementación, además, con este trabajo, se brinda una guía para familiarizar a implementadores y usuarios finales con los conceptos y actividades relacionadas con el Diseño de un Applet y Middleware para gestionar la información biométrica contenida en la Cédula de Identificación Electrónica de la República Bolivariana de Venezuela.

- ✓ Con esta investigación se resuelve uno de los problemas que enfrenta el nuevo proceso de cedulación en Venezuela en cuanto a la seguridad de los servicios que se brindarán a las personas con la Cédula de Identificación Electrónica.
- ✓ Para la realización de este trabajo se investigaron los procesos que se llevan a cabo para la gestión de la información biométrica en la Cédula de Identificación Electrónica, además de realizarse un análisis de las diferentes tecnologías de desarrollo utilizadas.
- ✓ La modelación del sistema se desarrolló utilizando la metodología XP pues permite el desarrollo a corto plazo del software, se adapta a los cambios y reduce la documentación. Se definieron los requerimientos del sistema, tanto funcionales como no funcionales, estructurándose además el modelo de dominio y los diagramas del diseño.
- ✓ Con el desarrollo de este trabajo se logró dar cumplimiento satisfactoriamente a las tareas de investigación trazadas para el análisis y diseño de una aplicación applet y su componente middleware para gestionar la información biométrica contenida en la Cédula de Identificación Electrónica de la República Bolivariana de Venezuela además se han incluido una serie de recomendaciones que deben tenerse en cuenta para el trabajo futuro.

RECOMENDACIONES

A modo de recomendaciones para próximas iteraciones se propone:

- ✓ Seguir profundizando en el estudio de los procesos de la gestión de información biométrica en tarjetas inteligentes.
- ✓ Realizar la modelación e implementación de la autenticación asimétrica entre la Cédula de Identificación Electrónica (CIE) y la Terminal de Servicios, para garantizar que estas se reconozcan como válidos y poder intercambiar datos de forma segura.
- ✓ Continuar con la implementación del estándar ISO/IEC 7816 para soportar diversos mecanismos de almacenamiento y elementos de seguridad.
- ✓ Organizar una infraestructura económica lo suficientemente robusta como para aplicar esta solución en nuestro país.
- ✓ Ampliar la visión y alcance de la aplicación e incursionar en la web.

BIBLIOGRAFÍA

1. **ORTEGA, M.** Implementación de Tarjeta Inteligente Java Card para el Control de Acceso a Instalaciones. [En línea] 2011. Disponible en: http://www.eatis.org/eatis2010/portal/paper/memoria/html/files/sistemas/Maria%20Ortega.pdf.

2. **LEYVA., M. R. P. C. L. F.** Sistema de Administración de Tarjetas Inteligentes y Aplicaciones para la Cédula de Identidad Electrónica de la República Bolivariana de Venezuela. Trabajo. UCI. 2008.

3. **ANDRES.** Tarjetas Inteligentes. [En línea] Disponible en:http://aprendamossobretarjetasdelpc.blogspot.com/2010_04_01_archive.html.

4. **DAYRON ALMEIDA SOTOLONGO, J. S. V.** Solución para el control de acceso a la información de las entidades externas, en la cédula de identificación electrónica de la República Bolivariana de Venezuela. 2008.

5. **GLOBALPLATFORM.** Card Specification. 2003.

6. **KARL.** Middleware. [En línea] 2011. Disponible en: http://www.buenastareas.com/ensayos/Middleware/2031177.html.

7. **EXPÓSITO, E. D.** Metodologías de desarrollo de software. ¿Cuál es el camino? .

8. **GARCERANT.** Modelo de Dominio. [En línea] 2010. Disponible en: http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.

9. **LODOÑO.** Una Introducción a Scrum. [En línea] 2005. Disponible en: http://www.scribd.com/doc/27034150/002-introduccion-a-SCRUM..

10. **Agora SIC_39.** Identificación biométrica y su unión con trajetas inteligentes. España : s.n., 2000.

11. **(ISO/IEC, 2005), 2000), 2002), 2008. MOC_FAQs.**

12. **(ISO/IEC, 2005), 2000), 2002), 2008).** Biometría en las Tarjetas Inteligentes.

13. **(ISO/IEC, 2005), 2000), 2002). 2008.** Gemalto. IAS Classic Applet. 2008.

14. **(ISO/IEC, 2005), 2000), 2002), 2008).** RMI_Client_API_2_2_2.pdf.

15. **(ISO/IEC, 2005), 2000).** RSA, Laboratorios. pkcs-1v2-1.pdf. 2002.

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

17. **Gemalto.** Criptomanager Applet,Reference Manual. 2008.

18. **GLOBALPLATFORM**. Card Specification. 2003.

19. **Kniberg, Henrik.** Scrum y XP desde las trincheras. 2007. 978-1-4303-2264-1.

20. **Fowler, M.** “Is Design Dead?”. [En línea] 2001.http://www.martinfowler.com/articles/designDead.html.

21. **ERLICH, J.** Especificación Formal de la Máquina Virtual Java Card. [En línea] Disponible en:http://www.fing.edu.uy/.../informacion/.../documentacion_especificacionjavacard.doc.

REFERENCIAS BIBLIOGRÁFICAS

Agora SIC. 2000. Agora SIC. 2000.

Álvarez Reyes, Ing. Julio C. Tecnologías Biométricas. [En línea] http://www.slideshare.net/juliozet/tecnologiass-biometricas.

Chan, Y.L., H.Y. What is a Java Smart Card? s.l. : SURPRISE 98.

Dallas Semiconductor. Dallas Semiconductor - iButton. iButton. [En línea] http://www.ibutton.com.

Di Giorgio, Chen. 1998. Understanding Java Card 2.0, Sun Microsystems. [En línea] 1998. http://www.javaworld.com/javaworld/jw-03-1998/jw-03-javadev.html.

Gemalto. Developer Suite. [En línea] http://www.gemalto.com.

Gemplus. Gemplus. [En línea] http://www.gemplus.com.

Giorgio, Di. 1997. Smart Cards: a primer, Sun Microsystems. [En línea] 1997. http://www.javaworld.com/jw-12-1997/jw-12-javadev.html.

ISO 7816. ISO 7816. ISO 7816/2.

ISO 7816-1. 7816-1.

ISO/IEC 7816. 2006. 2006.

Kniberg, Henrik. 2007. Scrum y XP desde las trincheras. 2007. 978-1-4303-2264-1.

Mendoza, Eduardo. 2008. Mendoza. 2008.

Precise Biometrics. Precise BioMatch™ J 3.0 Manual.

—. 2005. Precise Biometrics. 2005.

SAIME. 2011. SAIME. [En línea] 2011. http://www.saime.com.

Schlumberger. Schlumberger. [En línea] http://www.slb.com.

Sun Microsystem. 1999. Java Card 2.1 API Specifications. [En línea] 1999. http://java.sun.com/products/javacard/javacard21.html.

Sun Microsystems. 1999.
Java Card 2.1 Runtime Environment (JCRE) Specification. [En línea] 1999. http://java.sun.com/products/javacard/javacard21.html.

—. 1999. Java Card Virtual Machine Specification v2.1. [En línea] 1999. http://java.sun.com/products/javacard/javacard21.html.

Technology, Information. 2002. Information technology– Abstract Syntax Notation One (ASN.1). 2002.

thefreedictionary. 2010. thefreedictionary Applet. [En línea] 25 de Noviembre de 2010. http://www.thefreedictionary.com/applet.

—. 2010. thefreedictionary Middleware. [En línea] 25 de Noviembre de 2010. http://encyclopedia2.thefreedictionary.com/middleware.

Proenza. 2005. 2005.

Penadés, Patricio Letelier y Mª Carmen. 2004. Métodologías Ágiles para el desarrollo de software: eXtreme Programming (XP). [En línea] 2004. http://www.willydev.net/descargas/masyxp.pdf.

Escribano, Gerardo Fernández. eXtreme Programing/Programación Extrema.

Albretch. 1979. 1979.

Boehm. 1995/2. 1995/2.

COCOMO II.

Fowler, M,. 2001. 2001.

Wells, J. Donovan. 1999. [En línea] 1999. http://www.extremeprogramming.org/.

GLOSARIO DE TÉRMINOS

Autenticación: Acto de establecimiento o confirmación de algo (o alguien) como auténtico, es decir probar que es verdadero. La autenticación de un objeto puede significar la confirmación de su procedencia, mientras que la autenticación de una persona consiste en verificar su identidad.

Identificación Biométrica: Sistema de reconocimiento estadístico de patrones que establece la autenticidad de una característica fisiológica o de comportamiento que posee un usuario.

Cédula de Identificación Electrónica (CIE): La CIE es un documento que acredita la identidad de una persona. Es de carácter personal e intransferible, y constituye el documento principal de identificación para los actos civiles, mercantiles, administrativos y judiciales, y para todos aquellos casos en los cuales su presentación sea exigida por la ley, además está constituida por un chip que tiene alojado los applets que permiten la gestión de la información que se almacena.

Applet: Es un componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo un navegador web. Los applets, según el contexto del trabajo, son aplicaciones implementadas utilizando la tecnología JavaCard y son ejecutadas dentro de las tarjetas inteligentes.

Middleware: Software que funciona como una conversión o capa de traducción. Los middlewares son soluciones personalizadas que se han desarrollado durante décadas para permitir a una aplicación comunicarse con otra, ya sea que se ejecuta en una plataforma diferente o viene de un proveedor distinto.

CAD: Tarjeta de dispositivo de aceptación.

Match on Card (MoC por sus siglas en inglés): es una tecnología que realiza comparaciones de huellas dactilares en tarjetas.

API: Una interfaz de programación de aplicaciones o API (en inglés Application Programming Interface) es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción.

HSM: Por sus siglas en inglés es Hardware Security Module (Módulo de Seguridad Hardware). HSM es un dispositivo criptográfico basado en hardware que genera, almacena y protege claves criptográficas y suele aportar aceleración hardware para operaciones criptográficas.

Subset de Java: Es un subconjunto del lenguaje Java, ya que introduce algunos elementos nuevos en la JVM, como mecanismos de seguridad adicionales.

APDU: El Application Protocol Data Unit (APDU) es la unidad de comunicación entre un lector de tarjetas inteligentes y una tarjeta inteligente. La estructura de un APDU está definida en los estándares ISO/IEC 7816.

ANEXOS

Anexo 1: Plan de entregas

Entregable	Fin Iteración 1	Fin Iteración 2	Fin Iteración 3
Applet y Middleware para gestionar la información biométrica contenida en la Cédula de Identificación Electrónica.	Febrero 2011	Marzo 2011	Abril 2011

Tabla 56. Plan de entregas.

Anexo 2: Estimación de tiempo de las historias de usuario.

Historia de Usuario	Estimación
Inicializar comunicación	**0.5**
Establecer canal seguro del Applet Secure Fingerprint Manager	**1**
Obtener información biométrica	**1**
Enrolar información biométrica	**1**
Realizar Match on Card	**1.5**
Personalizar el Applet Secure Fingerprint Manager	**1**
Gestionar almacenamiento de información	**2**
Eliminar información biométrica	**0.9**
Desbloquear información biométrica	**0.6**
Restablecer estado de autenticación	**0.4**
Gestionar ID de la tarjeta	**1**
Seleccionar Applet	**0.8**
Obtener propiedades	**0.6**
Gestionar estado	**1**
Abandonar estado virgen	**0.3**
Finalizar comunicación	**0.5**

Tabla 57. Estimación de tiempo de las historias de usuario.

Anexo 3: Plan de iteraciones.

Iteración	No.HU	Historia de Usuario	Duración estimada (semanas)
1	**HU_1**	Inicializar comunicación	12
	HU_2	Establecer canal seguro del Applet Secure Fingerprint Manager	
	HU_3	Obtener información biométrica	
	HU_4	Enrolar información biométrica	
	HU_5	Realizar Match on Card	
	HU_6	Personalizar el Applet Secure Fingerprint Manager	
	HU_7	Gestionar almacenamiento de información	
2	**HU_8**	Eliminar información biométrica	6
	HU_9	Desbloquear información biométrica	
	HU_10	Restablecer estado de autenticación	
	HU_11	Gestionar ID de la tarjeta	
	HU_12	Seleccionar Applet	
3	**HU_13**	Obtener propiedades	4
	HU_14	Gestionar estado	
	HU_15	Abandonar estado virgen	
	HU_16	Finalizar comunicación	

Tabla 58. Plan de iteraciones.

Anexo 4: Multiplicadores de esfuerzo

Nombre	Abreviatura	Valor	Justificación
Confiabilidad y complejidad del producto.	RCPX	1.74	La aplicación presenta un nivel de complejidad alto.
Reusabilidad requerida.	RUSE	1.15	El nivel de reusabilidad es alto.
Dificultad de la plataforma.	PDIF	0.87	Uso de la memoria y almacenamiento bajo, plataforma estable.
Aptitud del personal	PERS	0.83	La capacidad del personal es alta.
Experiencia del personal	PREX	0.87	El nivel de experiencia en el uso del lenguaje y la plataforma de trabajo es

			alto.
Facilidades	FCIL	0.62	Se utilizan herramientas de modelación que facilitan el trabajo y entornos de desarrollo integrados.
Cronograma de desarrollo requerido.	SCED	1.00	Se utilizó el tiempo planificado para el desarrollo de la aplicación Applet y su componente Middleware.
Total (*EMi*)		**0.77**	

Tabla 59. Multiplicadores de esfuerzo (factores de costo).

Anexo 5: Clasificación de los factores de esfuerzo.

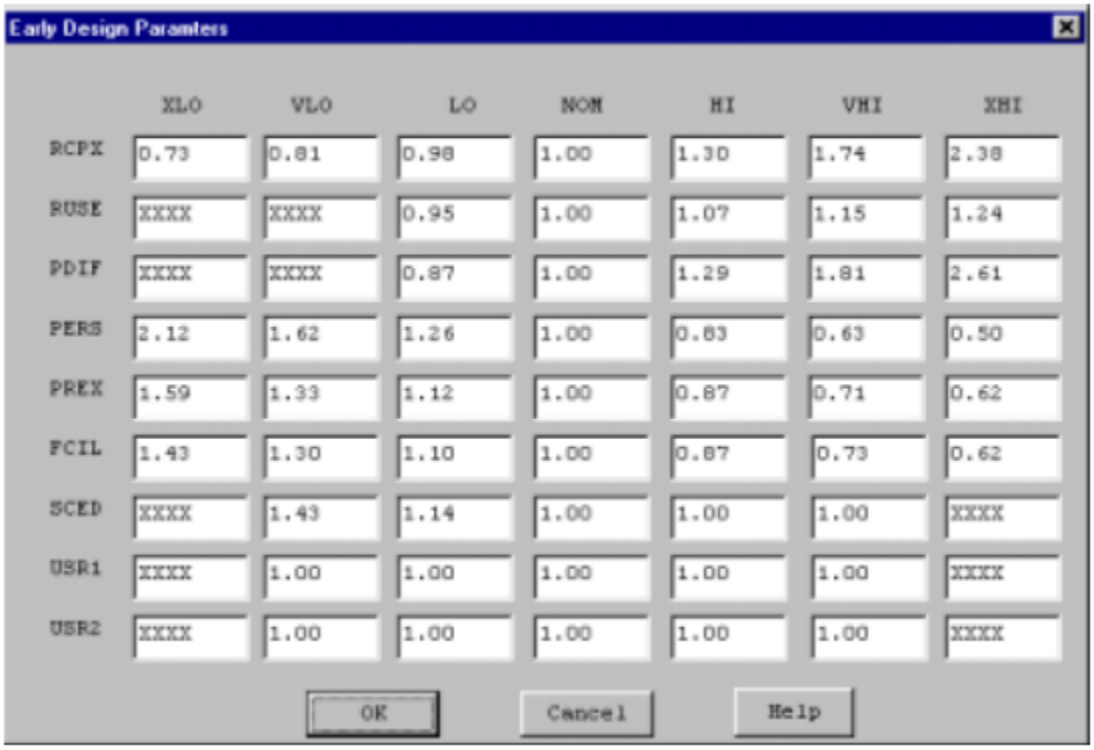

Figura 10: Multiplicadores de esfuerzo del modelo Diseño Temprano.

(COCOMO II)

Anexo 6: Factores de escala

Nombre	Abreviatura	Valor	Nivel

Precedencia	PREC	3.72	Normal
Flexibilidad en el desarrollo	FLEX	1.01	Muy Alto
Arquitectura/Resolución de riesgo	RESL	4.24	Normal
Cohesión de equipo	TEAM	1.10	Muy Alto
Madurez del proceso	PMAT	4.68	Normal
Total (*Wj*)		14.75	

Tabla 60. Factores de escala

Anexo 7: Clasificación de los factores de escala.

Figura 11: Factores de escala. (COCOMO II)

Anexo 8: Tablas de estimación de los Puntos Función sin ajustar.

Nombre de la entrada externa	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación (Bajo/Promedio/Alto)
Aplicaciones contenidas dentro de la tarjeta.	1	1	Bajo
Middlewares externos.	1	1	Bajo
Total		**2**	

Tabla 61.Entradas externas.

Nombre de la salida externa	Cantidad de ficheros	Cantidad de elementos de	Clasificación (Bajo/Promedio/Alto)

		datos	
Comandos APDU enviados a la tarjeta.	1	1	Bajo
Respuestas APDU enviadas al Middleware externo.	1	1	Bajo
Total		**2**	

Tabla 62. Salidas externas.

Nombre del fichero interno	Cantidad de elementos de registro	Cantidad de elementos de datos	Clasificación (Bajo/Promedio/Alto)
Plantilla contenedora con datos biométricos de referencia.	1	1	Bajo
Total		**1**	

Tabla 63. Archivos lógicos internos.

Nombre del fichero externo	Cantidad de elementos de registro	Cantidad de elementos de datos	Clasificación (Bajo/Promedio/Alto)
Total			

Tabla 64. Archivos externos de interfaces.

Nombre de la solicitud externa	Cantidad de elementos de registro	Cantidad de elementos de datos	Clasificación (Bajo/Promedio/Alto)
Total			

Tabla 65. Solicitudes externas.

Elementos	Bajo		Promedio		Alto		Subtotal
	No.	Peso	No.	Peso	No.	Peso	
Entradas externas	2	3	0	4	0	6	6
Salidas externas	2	4	0	5	0	7	8
Archivos lógicos internos	1	7	0	10	0	15	7

Archivos externos de interfaces	0	5	0	7	0	10	0
Solicitudes externas	0	3	0	4	0	6	0
Total							**21**

Tabla 66. Puntos Función sin ajustar.

Anexo 9: Pesos asignados a factores.

Fi	Nombre del factor	Peso (0-5)
F1	Mecanismos de recuperación y backups confiables.	3
F2	Comunicación de datos.	5
F3	Funciones de procesamiento distribuido.	2
F4	Performance.	2
F5	Configuración usada rigurosamente.	2
F6	Entrada de datos online.	1
F7	Factibilidad operativa.	1
F8	Actualización de archivos online.	1
F9	Interfaces complejas.	0
F10	Procesamiento interno complejo.	4
F11	Reusabilidad.	5
F12	Fácil instalación.	2
F13	Soporte de múltiples instalaciones.	1
F14	Facilidad de cambios y amigabilidad.	3
Total		**32**

Tabla 67. Pesos asignados a Fi.

Anexo 10: Cálculo de líneas de código.

Características	Valor
Puntos de Función sin ajustar	21
Líneas de código fuente	1550
Miles de líneas de código fuente	**1.550 (KSLOC)**

Tabla 68. Cálculo de líneas de código.

Anexo 11: Lista de reserva del producto.

Código	Descripción de la Historia de Usuario	Prioridad
RF1	Se muestran los lectores que están disponibles, se selecciona	Alta

	uno con el cual se va a establecer la comunicación con la Tarjeta Inteligente.	
RF2	Se establece un canal de intercambio de información de forma segura entre el Middleware y el Applet de Match on Card, utilizando Protocolo de Canal Seguro “01” según especificaciones de GlobalPlatform.	Alta
RF3	El Middleware realiza la petición de información biométrica al Applet Secure Fingerprint Manager, este último responde con los datos biométricos referente a la petición.	Alta
RF4	El Middleware envía los datos biométricos a almacenar en el Applet Secure Fingerprint Manager, el Applet almacena la información biométrica.	Alta
RF5	El Middleware envía los datos biométricos al Applet Secure Fingerprint Manager, el Applet obtiene la información biométrica a verificar, efectúa la comparación y devuelve la respuesta de la misma.	Alta
RF6	El Middleware envía los datos para configurar y realizar la instanciación del Applet de Match on Card; el Applet queda listo para almacenar la información biométrica.	Alta
RF7	Permitirá crear el estructura de ficheros en el Applet de Match on Card, donde se va almacenar la información biométrica y gestionar la seguridad para acceder a los datos almacenados.	Alta
RF8	El middleware envía al applet Secure Fingerprint Manager el comando que indica que se va a eliminar una de las plantillas contenedoras de información biométrica. El applet selecciona en uno de sus parámetros dicha plantilla, y la elimina.	Alta
RF9	El middleware envía al applet Secure Fingerprint Manager el comando que indica que se va a desbloquear la plantilla. El applet selecciona en uno de sus parámetros la plantilla contenedora en la tarjeta que contiene la plantilla bloqueada y la desbloquea.	Alta
RF10	El middleware envía al applet Secure Fingerprint Manager el	Alta

	comando que indica que se va a restablecer el estado de autenticación. El applet restablece el indicador de estado de autenticación de la plantilla contenedora específica.	
RF11	Esta funcionalidad presente dos estados posibles: • Modificar ID de la tarjeta • Obtener ID de la tarjeta El primero se encarga de actualizar el ID de la tarjeta. El área de datos de identificación se puede utilizar para almacenar datos o para identificar de forma única una tarjeta. El segundo estado se encarga de recuperar el ID de la tarjeta que contiene el applet Secure Fingerprint Manager.	Alta
RF12	Primeramente se selecciona una instancia del applet Secure Fingerprint Manager y una vez que la instancia del applet se ha seleccionado correctamente, los demás comandos disponibles pueden ser procesados de forma normal.	Alta
RF13	El middleware envía al applet Secure Fingerprint Manager el comando que indica que se desea obtener las propiedades de dicho applet. Esta función recupera la información general acerca del applet que está corriendo en la tarjeta, junto con la información del estado de las diferentes plantillas biométricas.	Alta
RF14	El middleware envía al applet Secure Fingerprint Manager el comando que indica que se va a pasar de un estado a otro. Esta función cambia el estado actual del applet, permitiendo de esta forma que el applet pase al siguiente estado correspondiente.	Alta
RF15	El middleware envía al applet Secure Fingerprint Manager el comando que indica que se va a abandonar el estado virgen. Esta función pasa del estado virgen, retornando el valor de la llave de inicio en texto plano, de forma que esta se mantenga segura, y se procede a pasar al estado de inicialización.	Alta
RF16	Consiste en realizar la desconexión entre la Tarjeta Inteligente donde se encuentra el Applet Secure Fingerprint Manager, y el lector.	Alta

RNF (Requisitos no funcionales)	
	Usabilidad
RNF1	El middleware debe ser de fácil utilización para lograr una mayor comodidad en su integración con aplicaciones existentes.
	Rendimiento
RNF2	El Applet debe ser capaz de realizar sus operaciones de manera eficiente, garantizando su funcionalidad en un corto intervalo de tiempo.
	Soporte
RNF3	Manual de usuarios. Sistema de ayuda. Manual de procedimientos.
	Interfaz Interna
RNF4	✓ Comunicación con lectores de tarjetas inteligentes. ✓ Comunicación con escáner de huella. ✓ Interfaz con el middleware de verificación biométrica. ✓ Interfaz con otras aplicaciones (API).
	Interfaces Hardware
RNF5	✓ Tarjeta Inteligente (SmartCard) y escáner de huellas dactilares. ✓ Lector de tarjetas incorporado a la PC que cumpla con el estándar PC/SC versión 1.0 ó superior. ✓ La aplicación JavaCard debe ejecutar las APIs biométricas (BioMatch™ C o *Precise* BioMatch™ J).
	Interfaces Software
RNF6	Se precisa de *BioMatch*™ Pro Toolkit 2.0 para generar y trabajar con las plantillas compatibles de *BioMatch*.
	Portabilidad
RNF7	✓ El middleware se debe desarrollar sobre una tecnología multiplataforma, que permita su utilización en distintos Sistemas Operativos. ✓ El middleware debe ser compatible con cualquier lector de tarjetas que cumpla con el estándar PC/SC.
Seguridad	
	Confiabilidad
RNF8	✓ La aplicación debe recuperarse en el menor tiempo posible en caso de

	producirse una falla. ✓ La información almacenada en el applet Secure Fingerprint Manager estará protegida de ataques externos a través de la seguridad que define el proveedor de tarjetas, su sistema operativo y la tecnología JavaCard.
	Confidencialidad
RNF9	✓ La información biométrica almacenada dentro de la tarjeta, estará protegida de acceso no autorizado, mediante el uso de los mecanismos de seguridad y estándares definidos por GlobalPlatform. ✓ Los datos transmitidos y recibidos de la tarjeta, serán cifrados con criptografía simétrica, según define el estándar GlobalPlatform.
	Integridad
RNF10	La información contenida en la tarjeta, será objeto de cuidadosa protección contra la corrupción y estados inconsistentes.

Tabla 69. Lista de reserva del producto.

Anexo 12: Diagrama de Clases.

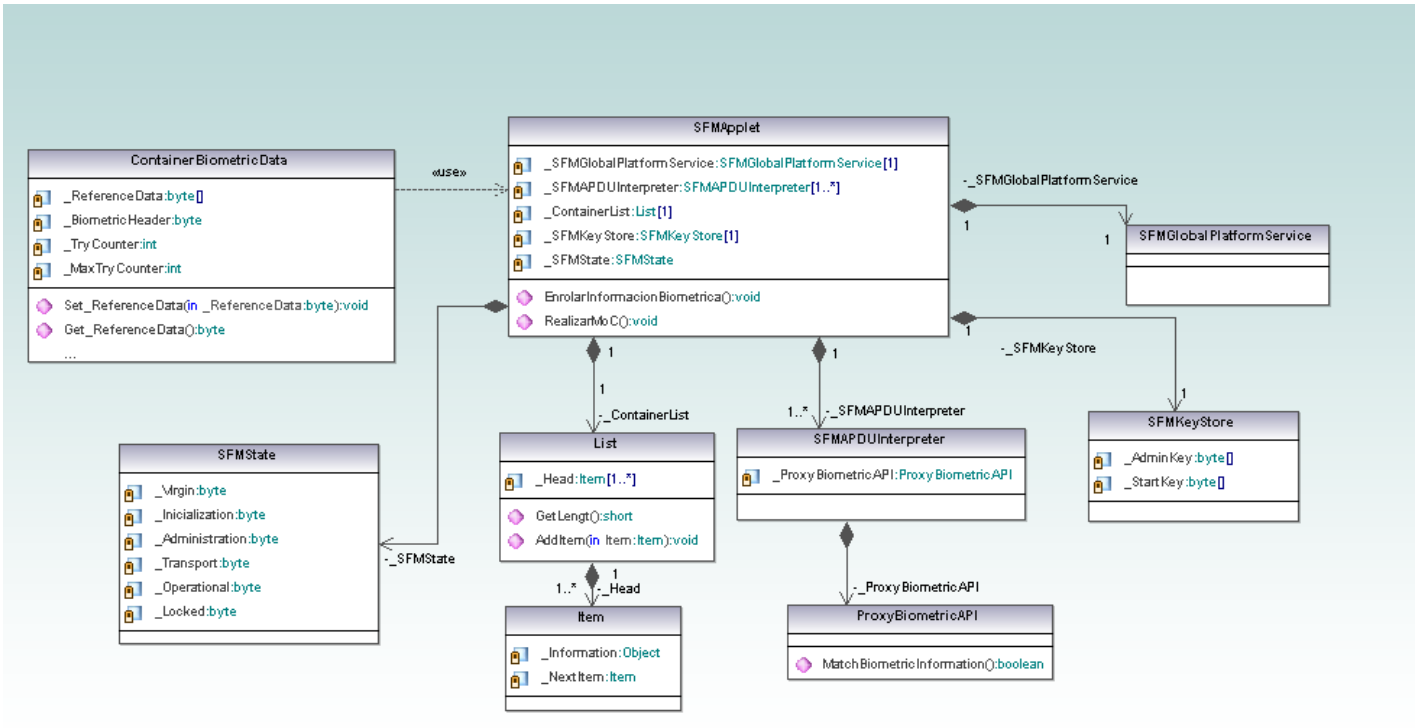

Figura 12: Diagrama de Clases