Universidad de las Ciencias Informáticas

Facultad 1

Componentes de la plataforma de extensión de capacidades de aplicaciones Java Card en tarjetas inteligentes

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

Autores: Julié Arianne Pérez Vive

Amed Alfonso Ríos

Tutores: Msc. Adonis Cesar Legón Campo

Ing. Susana María Ramírez Brey

"La Habana, junio del 2013"

"Se alcanza el éxito convirtiendo
cada paso en una meta y cada meta en un paso."
C.C. Córtez

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores del presente trabajo titulado: “Componentes de la plataforma de extensión de capacidades de aplicaciones Java Card en tarjetas inteligentes” y autorizamos a la Universidad de las Ciencias Informáticas y al Centro de Identificación y Seguridad Digital a usarlo en su beneficio.

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

_____ _____

Julié Arianne Pérez Vive Amed Alfonso Ríos

_____ ___ _____

Susana María Ramírez Brey Adonis Cesar Legón Campos

RESUMEN

Múltiples aplicaciones pueden ser desarrolladas para una tarjeta o pueden ser instaladas después de personalizadas al usuario final, sin embargo, aún existen restricciones en su uso debido a las limitaciones del procesador y de memoria, que no permite desplegar a los desarrolladores aplicaciones como la implementación de operaciones costosas para el chip de la tarjeta para este tipo de dispositivo.

Esta investigación pretende obtener una plataforma de software con los componentes necesarios para ejecutar operaciones de extensión (procesamiento y almacenamiento de la información) fuera de la tarjeta. De esta manera se resuelven las limitaciones de hardware existentes en las tarjetas y se aprovechan los recursos de hardware de las computadoras, con tarjetas de menores prestaciones, para ejecutar complejos algoritmos como, por ejemplo, la verificación biométrica del portador.

El sistema está compuesto por tres componentes que integrados funcionan como una plataforma de extensión: el componente Proxy que funciona como intermediario en la comunicación entre el componente Cliente encargado de decidir si se realiza una operación fuera de la tarjeta y el componente Servidor encargado de ejecutar las operaciones de extensión.

Palabras clave: Java Card, capacidades, procesamiento, memoria, tarjetas inteligentes.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Debido a la avanzada tecnología que se presenta en el mundo se hace necesario que constantemente se esté en evolución, y se aprovechen las ventajas que esta ofrece.

En este sentido, la búsqueda de mecanismos que permitieran el desarrollo de una tecnología para asegurar el almacenaje de datos en formato digital, simplificando las operaciones de lectura de estas por los cajeros y los terminales, fue unos de los principales objetivos de las llamadas tarjetas tradicionales de crédito y débito con banda magnética.

Junto con los beneficios de esas tarjetas y el surgimiento de la tecnología de los microprocesadores, las organizaciones se encuentran frente a la necesidad de garantizar la seguridad para prevenir fraudes y fuga de información, impulsando la evolución de las tarjetas al incrustarles un circuito integrado con capacidades de almacenamiento y procesamiento, resultando así, lo que conocemos hoy en día como tarjetas inteligentes.

Las tarjetas inteligentes nacieron a principios de los 80, y su filosofía es muy sencilla, tratan de almacenar información con una cierta autonomía y seguridad. Aunque la cantidad de información que pueden almacenar es relativamente pequeña, sus capacidades son lo suficientemente importantes como para haber producido la expansión de este tipo de tarjetas en el mercado [1, 2].

La SUSCERTE[1] define básicamente a una Tarjeta Inteligente como *"una tarjeta que contiene un pequeño microprocesador, que es capaz de hacer diferentes cálculos, guardar información y manejar programas, que están protegidos a través de mecanismos avanzados de seguridad."*

Las tarjetas más recientes contienen un co-procesador criptográfico que permite almacenar varios certificados y claves criptográficas, así como elevar la seguridad de la misma. Las tarjetas inteligentes son las soluciones de autenticación más robustas del mercado para garantizar y mejorar los accesos a las terminales de trabajo, resguardar la identidad digital de los funcionarios, preservar las claves privadas de las entidades certificantes, firmar documentos y correos, cifrar datos, proteger su software, y mucho más.

Una de las tecnologías de tarjetas programables[2] es Java Card. Una Java Card es una tarjeta inteligente capaz de incorporar y ejecutar programas escritos en un subconjunto del lenguaje de programación Java [3].

[1] Superintendencia de Servicios de Certificación Electrónica.

[2] Las tarjetas programables son aquellas capaces de realizar las tres fases de automatización: adquisición de datos, procesamiento o tratamiento de la información y control de elementos actuadores.

La tecnología Java Card permite a las tarjetas inteligentes y a otros dispositivos con memoria muy limitada ejecutar programas pequeños, llamados applets (aplicaciones que corren dentro de una Java Card), que emplean la tecnología de Java, además de proporcionar a los fabricantes de tarjetas inteligentes una plataforma de ejecución segura e interoperable que puede almacenar y actualizar varias aplicaciones en un único dispositivo. Al mismo tiempo que es compatible con estándares existentes de tarjetas inteligentes. Esta tecnología permite a los desarrolladores diseñar, construir, evaluar e implementar aplicaciones y servicios de forma rápida y segura, reduciendo costos, incrementando la productividad y generando un valor agregado al cliente.

La revolución tecnológica acontecida en los últimos años propició que los responsables de Java ofrecieran soluciones personalizadas a cada ámbito tecnológico, creando una edición distinta de Java Card según la necesidad de uso en el mundo y la cantidad de aplicaciones desplegadas para estas. Una Java Card tradicional tiene un dispositivo de 8 a 16-bit CPU (Central Processing Unit o Unidad Central de Procesamiento) corriendo a 3.7MHz, con 1Kb de RAM (Random-Access Memory o Memoria de Acceso Aleatorio) y más de 16Kb de memoria no volátil EEPROM (Erasable Programmable Read-Only Memory o ROM borrable programable). Pero en la actualidad una tarjeta de alto rendimiento posee un procesador independiente, un chip criptográfico, una memoria para cifrado, y 32 bit CPU, 24kb de RAM, más de 256 kb de ROM, y más de 128 kb EPROM [4]. Independientemente de que estas tarjetas tienen más capacidad no satisface las demandas de los desarrolladores, al mismo tiempo que no son las que predominan en el mercado por el alto costo de producción que poseen.

La tecnología Java Card tiene por objetivo permitir el desarrollo de aplicaciones en Java que puedan ser usadas en tarjetas inteligentes. Las principales restricciones que posee esta tarjeta tiene que ver con lo limitado de los recursos de hardware y el lenguaje, lo que con respecto a la tecnología Java:

- No posee Garbage Collection ni métodos destroy() explícitos.
- No posee soporte de disparadores porque no es posible implementarlos eficientemente en el hardware.
- No hay tipo de datos int (excepto en tarjetas de 32 bits) solo existen los tipos short, byte y boolean.
- No existe la clase String.
- Sólo hay soporte para arreglos unidimensionales.
- No hay soporte para tablas de Hashing.
- No hay soporte para aplicaciones Java, solo para Applets.
- Posee un API (Application Programming Interface o Interfaz de Programación de Aplicaciones) muy restringido [4, 5].

Debido a la memoria limitada y los recursos computacionales, muchos elementos y aplicaciones han sido eliminado para las Java Card; sin contar que existen mecanismos de las tarjetas tradicionales que constituyen un problema en el manejo de forma eficiente de la memoria disponible en la tarjeta, de modo tal que los desarrolladores pueden hacer uso de técnicas de ahorro de espacio que son más eficientes para el trabajo con la memoria limitada, pero esto traería anomalías en la estructura de los programas lo que es atípico para la tecnología Java y el desarrollo orientado a objetos [6].

En la Universidad de las Ciencias Informática (UCI), exactamente en el departamento Tarjetas Inteligentes del Centro de Identificación y Seguridad Digital (CISED), se trabaja con éste tipo de tecnología, con el fin de obtener productos y servicios basados en tarjetas inteligentes, pero se han detectado algunos problemas para el desarrollador, en cuanto al despliegue de las distintas aplicaciones por las limitaciones de almacenamiento y procesado de información que éstas poseen.

Derivado de la situación anteriormente expuesta se define el siguiente **problema de la investigación:** ¿Cómo extender las capacidades de procesamiento y memoria de los applets Java Card basados en tarjetas inteligentes?

Como **objeto de estudio** se tiene: Desarrollo de aplicaciones para tarjetas inteligentes.

Para resolver el **problema de investigación** trazado se precisó como **objetivo general:** Desarrollar los componentes de la plataforma que permitan extender las capacidades de procesamiento y memoria de los applets Java Card en tarjetas inteligentes.

Además como **objetivos específicos** se toman:

- Elaborar el marco teórico de la investigación.
- Realizar el diseño de los componentes de la plataforma y de una arquitectura que permita la extensión de las capacidades de procesamiento y memoria de los applets Java Card en tarjetas inteligentes.
- Implementar los componentes necesarios del lado del cliente y el servidor para la realización de las operaciones de extensión de la plataforma.
- Realizar pruebas que integren los componentes desarrollados para la plataforma con un applet Java Card que necesite de las operaciones de extensión.

Para dar respuesta a la interrogante presentada en este trabajo y con los objetivos trazados se plantea el cumplimiento de las siguientes **tareas de la investigación:**

- Análisis de las principales soluciones que existen para la extensión de las capacidades de procesamiento y memoria en tarjetas.
- Análisis del SmartCard Framework sobre el que se va a desarrollar los componentes que interactúan con la plataforma.

- Descripción de las herramientas, tecnologías y metodologías de desarrollo de software seleccionadas para el desarrollo de los componentes.
- Realización del diseño de una arquitectura flexible para dar solución a la problemática existente haciendo uso de patrones arquitectónicos.
- Realización del diseño de las clases de cada componente de la plataforma y la interacción entre estas, haciendo uso de los patrones de diseño existentes.
- Implementación de una API Java Card que abstraiga al desarrollador del envío y recepción de los comandos que se intercambian entre el middleware y un applet específico, durante la ejecución de operaciones de extensión fuera de la tarjeta.
- Implementación de un componente que funcione como proxy, que integrado al SmartCard Framework se encargue de realizar todas las operaciones de comunicación entre el cliente y el servidor.
- Implementación de un componente en el servidor que permita la ejecución de las operaciones de extensión definidas y el almacenamiento de información fuera de la tarjeta.
- Realización de pruebas de unidad a los componentes.
- Realización de pruebas de aceptación a los componentes.

La investigación se sustenta en los siguientes **métodos científicos:**

Métodos teóricos:

Analítico-sintético: En este trabajo se utiliza para sintetizar y obtener los elementos más importantes de la información contenida en todos los documentos analizados.

Análisis histórico-lógico: En este trabajo posibilita la compresión lógica del objeto de estudio haciendo un análisis riguroso de sus antecedentes y el proceso evolutivo por el cual han transitado todas las tecnologías relacionadas con las tarjetas inteligentes y las Java Card.

Modelación: En este trabajo se utiliza para la definición de la estrategia a aplicar debido a que es necesario modelar los componentes que más tarde serán utilizados para elaborar la arquitectura que dará solución al problema existente.

Métodos empíricos:

Entrevista: Se utiliza este método con el propósito de obtener información, experiencias, ideas, puntos de vistas, que contribuyan al desarrollo de la investigación y aporten conocimientos específicos del tema, a especialistas en el desarrollo de aplicaciones de tarjetas inteligentes del centro CISED.

Como **posibles resultados** se espera:

- El funcionamiento de los componentes integrados que permitan extender las capacidades de procesamiento y memoria de los applets Java Card en tarjetas inteligentes, y abstraiga al usuario desarrollador final de estas operaciones.
- Documentación de los componentes de la plataforma según la metodología de desarrollo que se seleccione.

Justificación de la Tesis:

En la actualidad una mayor cantidad de usuarios llevan consigo una tarjeta inteligente, debido a las características de portabilidad de las mismas, aceptando que las tarjetas inteligentes representan la mejor alternativa para el almacenamiento de la información del propietario. Las restricciones de procesamiento y memoria presentes no beneficiarían al progreso tecnológico, y su implementación es posible y enriquecedora para los entornos de comercio; por ello este trabajo se centra en la búsqueda de soluciones a las limitantes de las tarjetas inteligentes en cuanto a la capacidad de procesamiento y memoria, por el aumento de valor que tienen las mismas para los clientes hoy en día y por el despliegue de aplicaciones y servicios de forma rápida y segura.

Con el desarrollo de esta investigación se pretende obtener una plataforma de software con los componentes necesarios para ejecutar operaciones de extensión (procesamiento y almacenamiento de la información) fuera de la tarjeta.

De esta manera se resuelven las limitaciones de hardware existentes en las tarjetas, y se aprovechan los recursos de hardware de las computadoras, con tarjetas de menores prestaciones, para ejecutar complejos algoritmos como, por ejemplo, la verificación biométrica del portador.

Lo novedoso de la investigación está reflejado en el uso de este enfoque para aumentar las capacidades de memoria y procesamiento de las tarjetas inteligentes, en contraposición con la tendencia tradicional de desarrollar el hardware y no buscar soluciones de software para las limitaciones que estas poseen. Para Cuba representa un avance en el desarrollo de soluciones para esta tecnología, que si bien es conocida y ampliamente utilizada a nivel mundial, en el país se están dando los primeros pasos para la utilización de la misma.

El trabajo se encuentra estructurado de la siguiente forma:

Capítulo 1: Fundamentación Teórica: Este capítulo contiene una base teórica para entender el problema planteado, en él se describen los aspectos fundamentales relacionados con tarjetas inteligentes y los componentes de la tecnología Java Card; además de las tecnologías relacionadas con el contexto de este trabajo, y que conforman un sólido estado de la cuestión en estas materias.

Capítulo 2: Propuesta de Solución: En este capítulo se presenta la propuesta de los componentes de la plataforma que permita extender las capacidades de procesamiento y memoria de los applets Java Card

en tarjetas inteligentes, que abstraerá al usuario de estas operaciones. Se explican todos los aspectos concernientes al diseño de los componentes de la plataforma de extensión, se detallan los elementos relacionados con su estructura, el papel que juegan las actividades y tareas, incluyendo los métodos y procedimientos que se aplican, así como los responsables de ejecutarlas.

Capítulo 3: Implementación y Prueba: En este capítulo se presentan los resultados obtenidos después de haber aplicado la estrategia en el proyecto, así como los métodos empleados para publicar y dar seguimiento a las deficiencias encontradas en aras de evitar la repetición de los errores en posteriores etapas de pruebas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

La tecnología Java Card orientada a la programación de tarjetas inteligentes fue diseñada excluyendo ciertas estructuras de Java complejas o no aplicables a la programación de tarjetas inteligentes y agregando funciones específicas como el manejo de transacciones en tarjetas inteligentes. Para la realización de este capítulo se hizo un análisis del estado del arte de la tecnología Smart Card y Java Card incluyendo los estándares asociados a la implementación de éstas, se presenta un examen crítico de sistemas o soluciones con respecto al tema y una propuesta y selección de las herramientas, tecnologías, metodologías y lenguajes para el desarrollo de los componentes de la plataforma de extensión.

1.2 Tecnología Smart Card

1.2.1 Introducción a las Smart Card

Una tarjeta inteligente (Smart Card), o tarjeta con circuito integrado (ICC por su nombre en inglés Integrated Circuit Card), es cualquier tarjeta tamaño de bolsillo o más pequeña, con circuitos integrados que permiten la ejecución de cierta lógica programada [7]. Esto implica que las tarjetas inteligentes pueden recibir información y procesarla mediante las aplicaciones incluidas en el chip integrado, y enviarla a otros dispositivos.

La tecnología de tarjeta inteligente se ajusta a las normas internacionales (ISO/IEC 7816 y ISO/ IEC 14443) y está disponible en una variedad de formatos, incluyendo tarjetas de plástico, dijes, módulos de identidad de abonado (SIM) que se utilizan en los teléfonos móviles GSM, y USB-basados en *tokens*.

Según su arquitectura, una tarjeta inteligente puede constituir casi un computador minimalista[3], ya que incluye [8]:

- **CPU:** El procesador de la tarjeta; que suele ser de 8 bits, a 5 MHz. y 5 voltios, con módulos de hardware opcionales para operaciones criptográficas.
- **ROM:** Memoria interna (normalmente entre 12 y 30 KB) donde se graba el sistema operativo de la tarjeta, las rutinas del protocolo de comunicaciones y los algoritmos de seguridad de alto nivel. Esta memoria, como su nombre indica, no se puede reescribir y se inicializa durante el proceso de fabricación.
- **EEPROM:** Memoria de almacenamiento (equivalente al disco duro en una PC) en el que está grabado el sistema de archivos, los datos usados por las aplicaciones, claves de seguridad y las

[3] El término minimalista en su ámbito más general, se refiere a cualquier cosa que haya sido reducida a lo esencial, despojada de elementos sobrantes.

propias aplicaciones que se ejecutan en la tarjeta. El acceso a esta memoria está protegido en distintos niveles por el sistema operativo de la tarjeta.

- **RAM:** Memoria volátil para trabajo del procesador.

Aunque existe un diverso rango de ICC, se reconocen dos categorías principales. **Las tarjetas de memoria**, con sólo componentes de memoria no volátil (RAM) y posiblemente alguna lógica de seguridad, y **las tarjetas microprocesadoras**, que contienen memoria y microprocesadores. Estas tarjetas no contienen baterías; la energía es suministrada por los lectores de tarjetas.

Las primeras presentan externamente un área de contacto, regularmente con varios sectores de contacto revestidos de oro u otro material de alta conducción, con una superficie total de aproximadamente un cm^2. Cuando la tarjeta es insertada en un lector, el chip se conecta con los conectores eléctricos del dispositivo (por ejemplo los del teléfono celular en el caso de las tarjetas SIM o GSM), que pueden leer y transmitir información desde y hacia la ICC [7].

Tarjetas inteligentes sin contacto (contactless smart card) son ICC que se comunican usando tecnología RFID[4]. Estas tarjetas solo requieren estar cerca de una antena para completar la transmisión de información (el estándar sugiere distancias de 10 cm para algunos usos y hasta 50 cm para otros). Aunque existen aplicaciones de este tipo, por ejemplo, cajeros automáticos o sistemas de reconocimiento en aeropuertos, acceso a edificios y otros, las más ampliamente usadas en el presente sirven para sistemas de tránsito masivo, identificación en puestos de inmigración, acceso a instalaciones, estacionamientos, etc.

Dos categorías adicionales de las tarjetas son las tarjetas de interfaz dual y tarjetas híbridas. Una tarjeta híbrida tiene dos chips, uno con una interfaz de contacto y otro con una interfaz sin contacto, estos no están interconectados.

Una tarjeta de interfaz dual tiene un solo chip con dos interfaces de contacto y sin contacto. Con tarjetas de interfaz dual, es posible acceder al mismo chip usando ya sea un contacto o interfaz sin contacto con un nivel muy alto de seguridad [8].

1.2.2 Estándares y especificaciones de Smart Card

Organización Internacional de Normalización (ISO) / Comisión Electrotécnica Internacional (IEC)

La ISO/ICE es uno de los estándares mundiales de normalización de la tecnología de tarjetas, incluidas las tarjetas de plástico. Los patrones primarios para las tarjetas inteligentes son las normas ISO/IEC 7816, ISO/IEC 14443, ISO/IEC 15693 e ISO/IEC 7501.

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

El objetivo de este estándar es lograr la interoperabilidad entre los distintos fabricantes de tarjetas inteligentes y lectores de las mismas, en lo que respecta a características físicas, comunicación de datos y seguridad. ISO/IEC 7816 es una norma internacional dividida en catorce partes. ISO/IEC 7816 partes 1, 2 y 3 describe las tarjetas inteligentes de contacto y define los distintos aspectos de la tarjeta y sus interfaces, incluyendo las dimensiones físicas de la tarjeta, la interfaz eléctrica y los protocolos de comunicación [7]. La ISO/IEC 7816 partes 4, 5, 6, 8, 9, 11, 13 y 15 son relevantes para todos los tipos de tarjetas inteligentes (de contacto así como sin contacto), definen la estructura lógica de la tarjeta (archivos y elementos de datos), varios comandos que utiliza la interfaz de programación de aplicaciones para el uso básico, gestión de aplicaciones, la verificación biométrica y servicios criptográficos. La ISO/IEC 7816 parte 10 es utilizada por las tarjetas de memoria para aplicaciones tales como tarjetas telefónicas pre-pagadas o máquinas expendedoras (maquinas que proporcionan aperitivos, bebidas, golosinas y otros productos). La ISO/IEC 7816 parte 7 define un método seguro de base de datos relacional para tarjetas inteligentes basadas en las interfaces SQL (SCQL).

ISO/IEC 14443 es una norma internacional que define las interfaces a una proximidad de las tarjetas inteligentes sin contacto, incluyendo la interfaz de radio frecuencia (RF), la interfaz eléctrica, y las comunicaciones y protocolos anticolisión. ISO/IEC 14443 define especificaciones para las tarjetas compatibles a operar a 13,56 MHz y tiene una autonomía de hasta 10 centímetros (3,94 pulgadas).

GlobalPlatform

GlobalPlatform (GP) es una organización internacional sin fines de lucro. Su misión es establecer, mantener y fomentar la adopción de normas que permitan una infraestructura abierta e interoperable para tarjetas inteligentes, dispositivos y sistemas que simplifiquen y aceleren el desarrollo, despliegue y gestión de aplicaciones en las industrias.

La figura 1(tomado de [9]) muestra la arquitectura definida por GlobalPlatform:

- **Runtime Environment:** Comprende la máquina virtual de Java Card (JCVM) junto a clases y servicios definidos en la Java Card Application Programming Interface (API).
- **Security Domain:** Es un área protegida en la tarjeta inteligente, a estos dominios de seguridad se le asignan aplicaciones que pueden utilizar los servicios criptográficos que el dominio ofrece.
- **Card Manager:** Es el componente central de una tarjeta con arquitectura GlobalPlatform. Todos los servicios son ejecutados por él y ofrece interfaces para utilizar los servicios, internamente a través de las APIs de GlobalPlatform y externamente a través de los comandos APDU.
- **GlobalPlatform API:** Permite el acceso a las funciones del CardManager y autenticar el terminal con la tarjeta, utilizando un canal seguro.

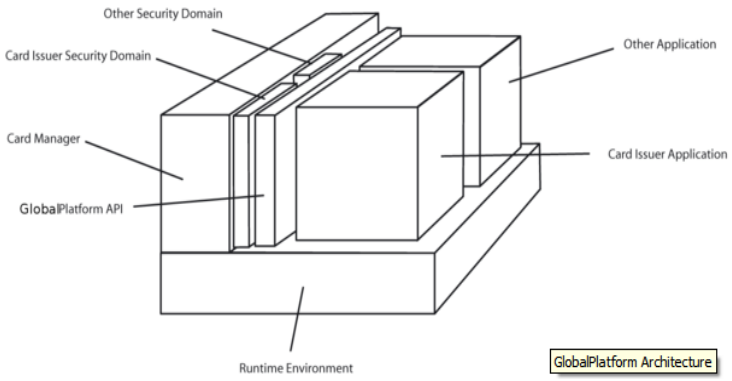

Figura 1 Arquitectura de las tarjetas según GlobalPlatform.

Ordenador Personal / Tarjeta Inteligente (PC/SC) Open Workgroup

El PC/SC Workgroup se formó en 1996 e incluyó Schlumberger Transacciones Electrónicas, Bull CP8, Hewlett-Packard, Microsoft y otros proveedores líderes. Este grupo ha desarrollado especificaciones abiertas para la integración de tarjetas inteligentes con las computadoras personales. Las especificaciones son independientes de la plataforma y basadas en estándares industriales existentes. PC/SC define una arquitectura de propósito general para el uso de tarjetas inteligentes en sistema de ordenadores, que permite a los desarrolladores crear aplicaciones basadas en las aplicaciones de red para la banca, la salud, la seguridad corporativa y el comercio electrónico. Las especificaciones incluyen funcionalidades criptográficas y almacenaje con seguridad, interfaces de programación para los lectores de tarjetas inteligentes y PCs, y una interfaz de aplicación de alto nivel para el desarrollo de aplicaciones. Los datos están basados en la norma ISO/IEC 7816, soporte EMV y estándares de aplicación GSM [7].

1.2.3 Aplicaciones y Problemas Existentes en las Smart Card

Las tarjetas inteligentes se utilizan en una amplia gama de industrias en todo el mundo para soportar aplicaciones de acceso, la identidad, de pago y otros. Día tras día se pueden encontrar nuevos tipos de aplicaciones que hacen uso de ellas. Las aplicaciones fundamentales de las tarjetas inteligentes son:

- Identificación del titular.
- Pago electrónico de bienes o servicios mediante dinero virtual.
- Almacenamiento seguro de información.

Existen en la actualidad una infinidad de productos disponibles en el mercado para todas las áreas del comercio y con las más distintas configuraciones de estándares y características de seguridad, pero debido a las limitaciones que existen respecto a la capacidad de almacenamiento y procesamiento, muchas empresas solo la utilizan para almacenar una pequeña parte de la información que se desea o simplemente están diseñadas para aplicaciones específicas, sin poder unir varias aplicaciones en una única tarjeta, ejemplo de ello podemos citar que los desarrolladores podrían incluir aplicaciones especiales

como la inserción de la tarjeta en una página web, la puesta en marcha de una página web de soporte configurado para las necesidades del usuario y el almacenamiento de toda la información personal (sito, historias clínicas, datos personales, certificados, entre otros). Estas tarjetas inteligentes usadas en las instituciones de educación superior son cada vez más importantes ya que se puede llevar un control de acceso físico, brindar diversos servicios de manera ordenada y efectiva y por ende un control a nivel de seguridad y de las actividades que se desarrollen en la institución. Cuanto mayor sea la capacidad de almacenamiento y la velocidad de procesamiento de la información de los chips, mayor será también su capacidad de soportar cualquier aplicación y funcionalidad. Esto último está directamente relacionado con su inserción en la vida de los ciudadanos, de esta forma se hace necesario buscar vías para lograr extender las capacidades de procesamiento y memoria de estas tarjetas.

1.3 Tecnología Java Card

Java Card es una tecnología que permite ejecutar de forma segura pequeñas aplicaciones Java (applets) en tarjetas inteligentes y similares dispositivos incrustados. Java Card da 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 por ejemplo (identificación, pago, etc.). Se usa ampliamente en las tarjetas SIM (utilizadas en teléfonos móviles GSM) y en monedero electrónico. A nivel de lenguaje, Java Card es un subconjunto de Java donde todas las construcciones del lenguaje Java Card existen en Java y se comportan de la misma manera [10]. La arquitectura básica de una Java Card consiste de Applets, Java Card API, Java Card Virtual Machine (JCVM) / Java Card Runtime Environment (JCRE), y el sistema operativo nativo de la tarjeta (Figura 2 tomado de [11]).

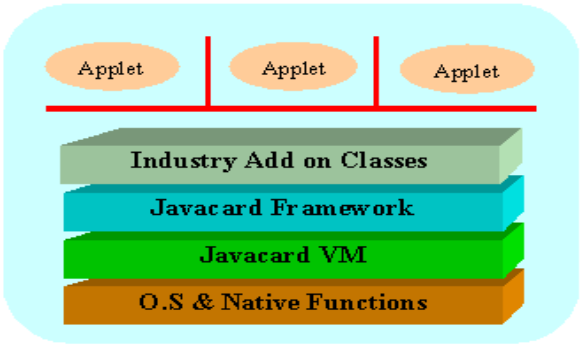

Figura 2 Estructura del software de la tarjeta

Como se ilustra en la Figura 2, la JCVM se construye sobre un circuito integrado específico (IC) y una implementación de sistema operativo nativo. La capa Java Card VM[5] oculta la tecnología propietaria con un lenguaje común y un sistema de interfaces. La capa Java Card Framework define un conjunto de

[5] Esta capa es lo que se conoce como JCVM (Java Card Virtual Machine).

clases de interfaces de programación de aplicaciones conocidas como APIs para el desarrollo de aplicaciones Java Card y para suministrar servicios de sistema a esas aplicaciones [11].

La corporativa Oracle publica la especificación de la plataforma Java Card, para proporcionar las bases para múltiples plataformas y la interoperabilidad entre proveedores de applets. La versión 2.2.2 de la especificación incluye tres documentos:

- **La especificación de tarjetas de Java Virtual Machine** define las características, servicios y comportamientos que una implementación de la tecnología Java Card debe soportar. Comprende el conjunto de instrucciones de una máquina virtual Java Card (VM), el subconjunto compatible del lenguaje Java, y los formatos de archivo utilizados para instalar los applets y las bibliotecas en tarjetas inteligentes y otros dispositivos que albergan la tecnología Java Card.
- **La especificación de tarjeta de Java Runtime Environment** define el comportamiento preciso del entorno de ejecución (RE) en cualquier aplicación de la tecnología Java Card, la cual debe incluir implementaciones de la máquina virtual de Java Card, las clases del API Java Card, y servicios de apoyo, tales como tiempo de ejecución de la selección y deselección de applets.
- **Las especificaciones del API para la plataforma Java Card** complementa la especificación de tarjeta Java Runtime Environment y describe la interfaz de programación de aplicaciones de la tecnología Java Card. El API es compatible con las normas internacionales oficiales y las normas específicas de la industria. Contiene las definiciones de clase necesarias para soportar la tarjeta Java VM y el RE Java Card.

La característica más significativa del entorno de ejecución de Java Card, JCRE, es que proporciona una separación clara entre el sistema de la tarjeta y las aplicaciones. El entorno de ejecución agrupa la complejidad y los detalles del sistema que se encuentran por debajo. Las aplicaciones solicitan servicios del sistema y recursos a través de una interfaz de programación de alto nivel bien definida. El JCRE es el responsable de gestionar los recursos de la tarjeta, de las comunicaciones, de la ejecución de los applets y de la seguridad de los procesos [10].

1.3.1 Componentes de la tecnología Java Card

Máquina Virtual de Java

La máquina virtual está dividida en dos partes una que se ejecuta fuera de la tarjeta y otra que se ejecuta dentro. Muchas de las tareas de procesado son realizadas por la máquina virtual que se ejecuta en el host, donde los recursos no son problemas [12, 13].

La especificación de la Máquina Virtual Java Card 2.1 define un formato de archivo llamado CAP, que es el estándar para la compatibilidad binaria para la plataforma Java Card.

Hay varios componentes que forman parte de un sistema Java Card, incluyendo la JCVM, el Java Card Convert, una terminal con herramienta de instalación, y un programa de instalación que corre sobre el dispositivo, tal como se muestra en la Figura 3 y 4 (tomado de [12]) .

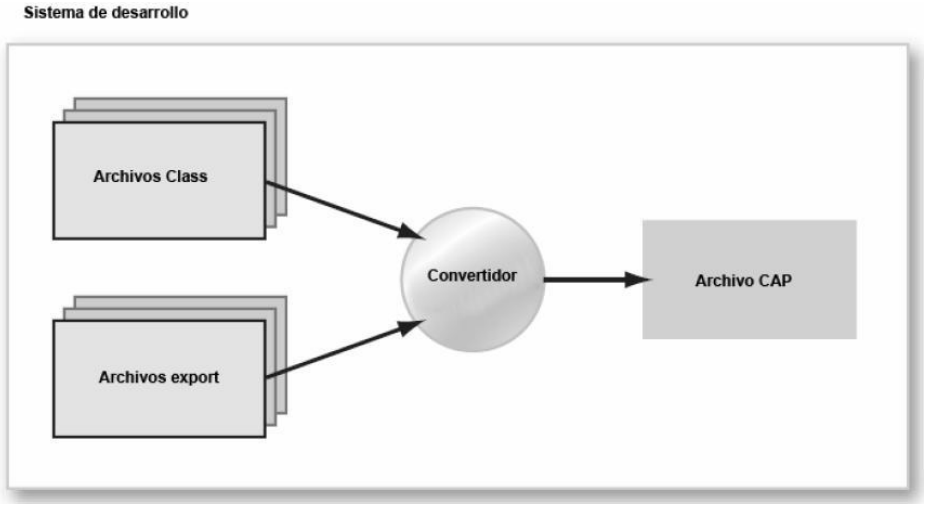

Figura 3 Máquina Virtual parte externa- Conversión de un applet Java Card.

El desarrollo de un applet Java Card comienza de la misma forma que cualquier otro programa Java donde un desarrollador escribe una o más clases Java luego compila el código fuente con un compilador Java, produciendo uno o más archivos punto *class.* El applet se corre, prueba y depura sobre una estación de trabajo usando herramientas que simulan el ambiente del dispositivo. Luego, cuando un applet está listo para ser cargado en un dispositivo, los archivos *class* que contienen el applet son convertidos a un archivo CAP usando un Java Card Converter. El Java Card Converter toma como entrada no solo los archivos punto *class* a ser convertidos, sino que también uno o más archivos *export.* Un archivo *export* contiene información sobre nombres y enlaces para el contenido de otros paquetes que son importados por las clases que son convertidas [12, 13].

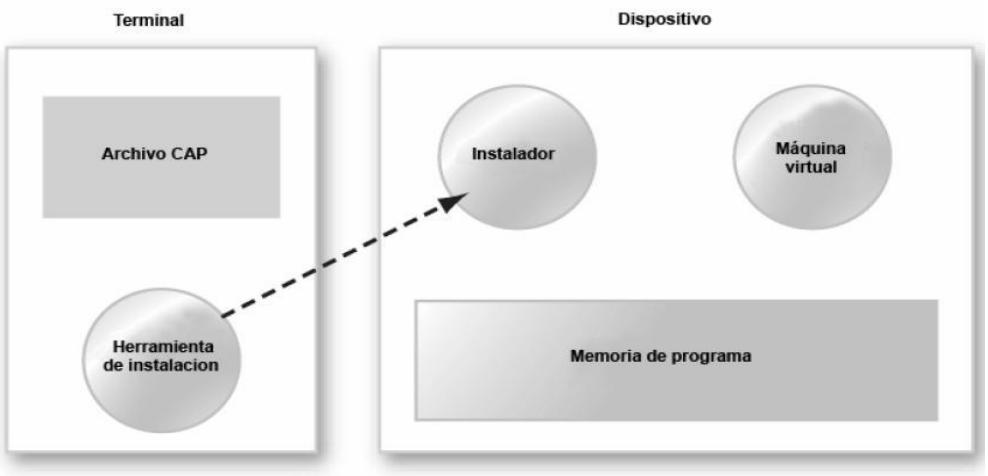

Figura 4 Máquina Virtual parte externa – Instalación de un applet Java Card.

Finalizada la conversión, el archivo CAP es copiado a una terminal Card. Luego una herramienta de instalación sobre la terminal carga el archivo CAP y transmite este al dispositivo que contiene la tecnología Java Card. Un programa de instalación sobre el dispositivo recibe el contenido del archivo CAP y prepara el applet para ser corrido por la JCVM. La máquina virtual en sí misma no necesita cargar o manipular archivos CAP, esta solo necesita ejecutar el código del applet que se encuentra en el archivo CAP que fue cargado sobre el dispositivo por el programa de instalación.

JavaCard Runtime Environment (JCRE)

El JCRE comprende la máquina virtual de Java Card junto a las clases y servicios definidos en el 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. Es responsabilidad del JCRE garantizar este comportamiento. Cuando se retira la tarjeta del terminal, se asume que se está ejecutando en un ciclo de reloj de período infinito. Otras diferencias entre ellas son las limitaciones en los tipos de datos manejados y los requerimientos de hardware para la ejecución [10, 13, 14].

Para poder comprender cómo funciona una Java Card, hay que tener en cuenta que al realizar la especificación de la plataforma, Sun[6] se apegó al estándar ISO 7816, el cual establece, entre otras cosas, la forma de comunicación entre una tarjeta inteligente y un terminal. De acuerdo al ISO 7816, el intercambio de información y comandos entre la tarjeta y el terminal se realiza a través de APDUs, los cuales son paquetes de información con un formato específico. De acuerdo al estándar, las tarjetas inteligentes nunca inician la comunicación con el terminal, sino que sólo responden a los comandos que éste les envía.[10]

La característica fundamental impuesta por el JCRE es el aislamiento de los applets usando lo que es llamado un applet cortafuego. El applet cortafuego impide que los objetos que son creados por un applet puedan ser usados por otros. Esto impide el acceso no autorizado tanto a los campos como métodos de una instancia de clase, así como también a la longitud y contenido de los arreglos. El aislamiento de los applets es una característica de seguridad importante, que requiere un mecanismo para permitir a los applets compartir objetos en situaciones donde hay una necesidad de interoperabilidad. El JCRE permite compartir datos usando el concepto de objetos de interfaces compartibles. Estos objetos proporcionan la única forma en que un applet puede hacer sus objetos disponibles para ser usados por otros applets [12].

[6] **Sun Microsystems** fue una empresa informática que se dedicaba a vender estaciones de trabajo, servidores, componentes informáticos, software (sistemas operativos) y servicios informáticos. Fue adquirida en el año 2009 por Oracle Corporation, anteriormente parte de Silicon Valley, fabricante de semiconductores y software.

Applets Java Card

Los applets son las aplicaciones que corren dentro de una tarjeta Java Card. 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.

Según el contexto del trabajo, son aplicaciones implementadas utilizando la tecnología Java Card y son accionadas dentro de las tarjetas inteligentes, encargada de realizar las operaciones que maneja, mediante los comandos APDU que le son enviados, retornando los resultados mediante APDU de respuestas [10, 13].

Una vez registrada en el JCRE (con el método register ()), entonces el applet está listo para ser ejecutado. La clase base applet es la superclase para todos los applets residentes en una tarjeta Java Card, en ella se definen las variables y métodos de un applet. El entorno de ejecución de Java Card soporta un entorno multiaplicación, es decir que pueden coexistir múltiples applets en una sola tarjeta inteligente, y un applet pude tener múltiples instancias [11-13].

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

El JCRE antes de crear la instancia del applet en la tarjeta, invoca a este método, cuya implementación habitual es llamar al constructor de la clase (generalmente este constructor es privado), crear todos los objetos necesarios para ejecutar el applet y por último registrar el applet con el método register ().

- Método select ()

Como resultado de la recepción a un SELECT APDU es solicitado este método por el JCRE. El APDU contiene el identificador de aplicación (AID por sus siglas en inglés) del applet a seleccionar.

El AID es una secuencia de entre 5 y 16 bytes, donde la ISO asigna los 5 primeros bytes (RID[7]) y los proveedores de aplicaciones definen los siguientes 11 bytes (PIX[8]) esto hace que la aplicación sea identificada de forma única, de acuerdo a la ISO/IEC 7816, y es la propia ISO quien le 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()

[7] RID: Registered application provider identifier en sus siglas en inglés o identificador de proveedor de la aplicación registrada.

[8] PIX: Proprietary application identifier extension en sus siglas en ingles o extensión de identificador de aplicación propietaria.

Para avisar al applet que se encuentra en ese momento seleccionado que va a dejar de estarlo, es que el JCRE invoca a este método.

APIs Java Card

Las APIs Java Card consisten en un juego de clases personalizadas para programar aplicaciones Smart Cards acordes al modelo ISO 7816. Estas contienen tres paquetes centrales: java.lang, javacard.framework y javacard.security. El paquete de extensión es javacard.crypto.

Las clases en las APIs de Java Card son compactas y cortas, incluyen clases adoptadas de la plataforma Java para proveer soporte al lenguaje Java y los servicios de criptografía, también contienen clases creadas específicamente para soportar el estándar ISO 7816 de Smart Cards [12, 13].

- Paquete java.lang

Es un riguroso subconjunto equivalente al paquete java.lang de la plataforma Java. Las clases soportadas son Object y Throwable y algunas clases de excepciones relacionadas con la máquina virtual. El paquete java.lang, provee el soporte fundamental para el lenguaje Java, la clase Object define una raíz para la jerarquía de clases de Java Card y la clase Throwable posee un ascendiente común para todas las excepciones.

- Paquete javacard.framework

Provee clases, soporte e interfaces para la funcionalidad principal de un applet Java Card, lo más importante es que define una clase base applet, que provee un soporte para la ejecución del applet y la interacción con el JCRE durante el tiempo de vida del applet. Su papel respecto JCRE es similar al de la clase applet en el explorador de un ordenador. Un desarrollador que quiera implementar un applet deberá extender la clase base applet y suscribir los métodos de la clase applet para implementar la funcionalidad.

- Paquete javacard.security

Provee un soporte para las funciones de criptografía soportadas en la plataforma Java Card, su diseño está basado en el paquete java.security. Define una clase de clave llamada KeyBuilder (clase que viene de fábrica) y varias interfaces que representan claves criptográficas usadas en algoritmos simétricos (como DES) o asimétricos (como DSA o RSA), además soporta las clases abstractas RandomData, Signature y MessageDigest, que se usan para generar datos aleatorios y calcular las asimilaciones y las firmas de los mensajes.

- Paquete javacards.crypto

Es un paquete de extensión que contiene clases de criptografía e interfaces que están sujetas a las exigencias expresadas en las regulaciones de exportación de los EE.UU, tolera la clase abstracta Cipher para soportar funciones de codificación y descodificación.

Define, además, junto con el paquete javacard.security las interfaces a los que los applets llaman para pedir los servicios de criptografía, sin embargo no provee ninguna implementación. [13]

1.3.2 Beneficios de la tecnología Java Card

Los proveedores de tarjetas inteligentes se benefician de las varias características únicas de la tecnología Java Card:

- **Interoperable:** Los applets desarrollados con tecnología Java Card se pueden ejecutar en cualquier tarjeta con tecnología Java habilitado para tarjetas inteligentes, independientemente del proveedor de la tarjeta y el hardware subyacente.
- **Seguridad:** La tecnología Java Card se basa en la seguridad inherente del lenguaje de programación Java para proporcionar un entorno de ejecución seguro. Diseñado por medio de un proceso abierto, las implementaciones probadas de la plataforma de la industria y las evaluaciones de seguridad garantizan que los emisores de tarjetas se beneficiaran de la tecnología más segura disponible hoy en día.
- **Capacidad extensible para múltiples aplicaciones:** La tecnología Java Card permite que varias aplicaciones coexistan de forma segura en una única tarjeta inteligente.
- **Dinámico:** Las nuevas aplicaciones se pueden instalar con seguridad después de que una tarjeta ha sido emitida, lo que los emisores de tarjetas pueden responder a las necesidades cambiantes de sus clientes de forma dinámica.
- **Compatible con los estándares existentes:** La API Java Card es compatible con las normas internacionales para las tarjetas inteligentes, como ISO 7816, o EMV. Principales normas específicas de la industria, tales como la Plataforma Global y ETSI se refirieron a ella [15].

1.3.3 Comparativa de las versiones de la tecnología Java Card. Restricciones de uso

Java Card es una tecnología que se crea con el fin de dar solución a la interoperabilidad de las tarjetas inteligentes, adaptado al lenguaje de programación Java para funcionar en tarjetas inteligentes y otros dispositivos con recursos limitados.

La antigua empresa Sun Microsystems se dio cuenta del potencial de las tarjetas inteligentes y dispositivos similares con recursos limitados y definió un conjunto de especificaciones para un subconjunto de la tecnología Java. Un dispositivo que soporta estas especificaciones se conoce como una plataforma Java Card, en la plataforma Java Card múltiples aplicaciones de diferentes proveedores pueden coexistir con seguridad. Con el paso de los años esta tecnología fue avanzando y con ellos se fueron resolviendo muchas restricciones que presentaba, en la siguiente tabla se muestran estas limitaciones (tomado de [4]).

Característica del lenguaje	No están soportadas características de carga de clase dinámica, Security Manager (java.lang.SecurityManager), hilos, clonación de objetos, y algunos aspectos de control del paquete de acceso.
Palabras claves	No son compatibles palabras como native, transient, volatile, synchronized, strictfp.
Tipos	No hay soporte para char, double, float, y long, o para arreglos multidimensionales.
Clases e interfaces	Las principales clases de la API de Java e interfaces (java.io, java.lang, java.util) no son compatibles con excepción de Object y Throwable, y la mayoría de los métodos de Object y Throwable no están disponibles.
Excepciones	Algunas excepciones y subclases de error se omiten porque las excepciones y los errores que encierran la plataforma Java Card no pueden darse.

Tabla 1 Resumen de las limitaciones de la tecnología Java Card.

También existen limitaciones en algunos modelos de programación. Por ejemplo, una biblioteca de clases cargada ya no puede ser ampliada en la tarjeta, sino que está implícitamente al final. Para hacer frente a las oportunidades en ambos extremos del espectro del hardware de las tarjeta inteligente, Sun introduce tecnología Java Platform Card Classic Edition, la primera nueva arquitectura de la tecnología Java Card en 10 años lanzada en el 2008, que sirve a los mercados existentes y a las aplicaciones desarrolladas, lo que permite liberar a las organizaciones de las limitaciones generalmente asociada con el despliegue de los servicios de seguridad.

Esta nueva edición cuenta con un entorno de ejecución significativamente mejorado y una nueva máquina virtual, se incluye una nueva red orientada a características como el soporte para aplicaciones Web, para applets extendidos y capacidades avanzadas, y se dirige a la gama alta de tarjetas inteligentes que viene de los últimos avances en tecnología de silicio. Esta edición es compatible con las aplicaciones escritas para versiones anteriores y comparte las características de seguridad derivada de los 10 años de la implementación de la seguridad en productos de la tecnología de tarjetas inteligentes [4].

Al igual que con versiones anteriores de la plataforma Java Card, se basa en una división de la tecnología de máquina virtual que permite el pre-procesamiento fuera de la tarjeta de las aplicaciones que se cargan en la tarjeta. Esta tecnología de máquina virtual que asegura la plataforma Java Card se puede implementar en las tarjetas con un mínimo de memoria y requisitos de procesamiento [16, 17].

En comparación con la tecnología Java Card 2.1.1, tecnología Java Platform Card Classic Edition, introduce varios cambios incrementales diseñados para asegurar que la tecnología siga siendo actual, y ocupándose de las necesidades de la mayoría de las implementaciones de tarjetas inteligentes. Esta

contiene soporte para los últimos algoritmos de criptografía, incluyendo 4096-bit RSA[9] y el Suite B[10] de la NSA, y la alineación con los últimos estándares de tarjetas inteligentes sin contacto.

La plataforma Java Card Connected Edition está diseñada para funcionar con una amplia variedad de tarjetas inteligentes y dispositivos seguros con recursos limitados. Típicamente tiene un procesador más rápido y una RAM más volátil y un EEPROM persistente.

Una característica clave de esta tecnología de gama alta es el apoyo de un full-duplex, de interfaz de alta velocidad con su dispositivo de alojamiento (tal como un teléfono) y conectividad a algunos tipo de red, por lo general a través de su dispositivo de alojamiento [4]. La siguiente tabla (tabla 2 tomado de [4]) compara el hardware de tarjeta inteligente dirigida por la Plataforma Java Card, tecnología Connected Edition con el hardware tradicional dirigida por las anteriores versiones de la tecnología Java Card y la tecnología Classic Edition.

Hardware tradicional	**Hardware de la tecnología de gama alta**
8/16 bit CPU	32 bit CPU
2kb RAM	24 kb RAM
48 a 64 kb ROM	>265 kb ROM
8 a 32 kb EPROM	>128 EPROM
Serial I/O interface	High-Seep interfaces
9.6 kb/sec – 30 kb/sec	1.5 Mb/sec – 12 Mb/sec
Full duplex[11]	Half duplex[12]

Tabla 2 Evolución de la tecnología Java Card dirigida a las configuraciones de dispositivos.

La plataforma Java Card Connected Edition incluye las características de la tecnología Java que no se encuentra en la edición clásica, como pueden ser, que hay soporte para char, double, float, y long, o para arreglos multidimensionales, además manipula las clases de string como StringTokenizer, StringBuffer, StringBuilder, y usa un recolector de basura automático, entre otras cosas [4].

Conjuntamente de las mejoras tecnológicas, la próxima generación de tecnología Java Card se dirige a los requisitos ya mencionados, ofreciendo las siguientes características:

[9] El sistema criptográfico con clave pública RSA es un algoritmo asimétrico cifrador de bloques, que utiliza una clave pública, la cual se distribuye (en forma autenticada preferentemente), y otra privada, la cual es guardada en secreto por su propietario.

[10] Estándar de algoritmos de cifrados interoperable.

[11] Full Duplex: transmite y recibe en ambas direcciones al mismo tiempo.

[12] Half dúplex: transmite y recibe en ambas direcciones, pero sólo ocurre una transmisión a la vez, es decir, no hay comunicación bidireccional simultáneamente, se debe esperar a que se termine de transmitir para poder recibir.

- Smart Card Web Server.
- La ejecución en paralelo de distintos servicios.
- Conectividad mejorada: Sin necesidad de software específico en el lector de tarjetas inteligentes.
- Modelo mejorado, seguridad escalable [16, 17].

1.4 Análisis Crítico de Soluciones Similares

Extending the Data Storage Capabilities of Java – based Smart Card es un artículo publicado por los profesores de informática Clemens H. Cap, Nico Maibaum y Lars Heyden de la Universidad de Rostock, Alemania, donde exponen una solución similar a lo que se quiere lograr. En este artículo la idea básica de la extensión de la tarjeta inteligente es para almacenar datos de aplicación no en la propia tarjeta inteligente, pero si en un servidor externo (memoria virtual). Esto requiere una conexión de red a un servidor de memoria y dos componentes de software: el *Servidor de Comunicación* para la comunicación con el servidor y el *Administrador de Memoria* que decide qué acceso a dato es mapeado dentro de la tarjeta y cual quiere delegado al servidor. Enfocan su solución solo al almacenamiento de información y no a la ejecución de operaciones de mucho costo computacional para el chip de las tarjetas.

En el diseño que plantean para la extensión de la tarjeta inteligente, el CardService se modifica, tomando la decisión de asignar un acceso a datos en el servidor de memoria externa desde el terminal o en la tarjeta, además de una extensión de la CardService, esta perspectiva requiere una pequeña modificación de las funciones de acceso a las aplicaciones de la tarjeta inteligente y provee un id de acceso que permite dar plena confianza en el terminal lo que no siempre es seguro. La estructura de la arquitectura de extensión se muestra en la figura 5 (tomado de [6]).

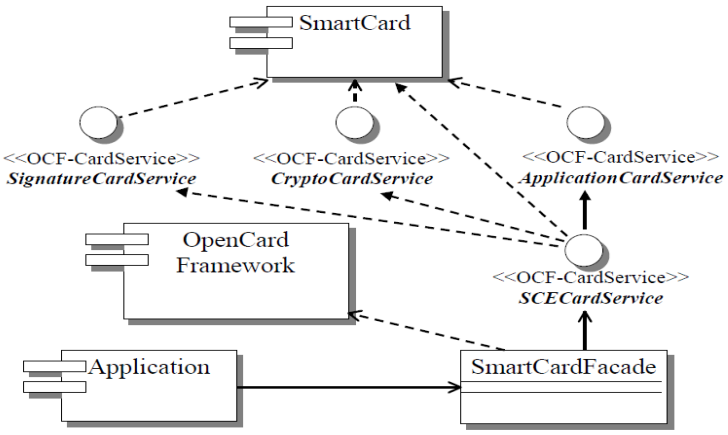

Figura 5 Extensión de las tarjetas inteligentes.

En la actualidad esta arquitectura se utiliza en el proyecto FASME (Facilitating Administrative Services for Mobile Europeans) de la Unión Europea para almacenar un mayor número de documentos en formato

XML de base administrativa en una Java Card para los propósitos de gobierno electrónico, y en el ámbito social se utiliza para mapear los procesos administrativos requeridos para el ciudadano europeo en la infraestructura electrónica móvil (ejemplo: registrar un coche o un nuevo lugar para vivir).

A modo de conclusión se considera que el resultado de la investigación de los informáticos alemanes además de ser un producto propietario al que no se tiene acceso al código fuente, lo que exponen sobre el acceso a los datos desde el terminal hace que la arquitectura que ellos definen corra el riesgo de ser manipulada por un personal no autorizado, al mismo tiempo que solo enfocan su solución al almacenamiento de información. Lo que en comparación a la solución que se propone se hace todo desde la aplicación Java Card o sea del chip, para garantizar un modelo de implementación orientado al desarrollador Java Card que es, en un final, a quien le interesa ejecutar una operación de extensión, o no.

1.5 Herramientas y tecnologías de desarrollo

1.5.1 Metodología XP

Kent Beck, el padre de XP, la describe como una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo [18]. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes y simplicidad en las soluciones implementadas. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico [19].

Razones para utilizar XP

La mayoría de los proyectos de software fracasan porque exceden sus plazos, superan su presupuesto, no se ajustan a las auténticas necesidades del cliente, presentan una calidad deficiente o en muchos casos, son abortados.

El presente proyecto de desarrollo trata de controlar esta situación mediante la utilización de la metodología ágil XP, cuya característica principal consiste en contemplar y dar respuesta a las necesidades dinámicas del cliente, mediante la identificación y reducción de riesgo por medio de un desarrollo iterativo, con capacidad de respuesta ante cambios; permitiendo la adaptación a nuevos requisitos de la organización, considerando una mínima inversión inicial y mostrando resultados tangibles logrando de esta forma efectividad y reduciendo (sin eliminar) la necesidad de documentación escrita.

1.5.2 Altova UModel

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

software mediante la potencia del estándar UML 2.1. Permite modelar los diseño de aplicación y generar código Java o C# a partir de sus planos, o realizar ingeniería inversa de programas existentes a diagramas UML claros y precisos. Corrige el código generado o los modelos y produce automáticamente nuevos diagramas o regenera el código. [20, 21].

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

1.5.3 Gemalto Developer Suite

Developer Suite es un IDE (Entorno de Desarrollo Integrado referenciado a JavaCard) que proporciona un conjunto de herramientas para crear y depurar applets Java Card. Está diseñado para crear rápidamente aplicaciones prácticas en los estándares de una tarjeta inteligente real. 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 [22].

Developer Suite a diferencia del NETBeans permite desarrollar aplicaciones para la versión de Java Card que poseen las tarjetas utilizadas en el departamento de tarjetas inteligentes del CISED, además tiene integrado el JCardManager para la simulación de las funcionalidades de los applet.

1.5.4 JCardManager

JCardManager es una solución de Gemalto que permite a cualquier desarrollador de Java Card administrar y probar nuevas aplicaciones con Java Card y GlobalPlatform.

Es una herramienta para la gestión del uso de una Java Card o un simulador móvil dentro de un entorno JDK. Sus funciones principales son:

- Realización de tarjeta / autenticación del terminal y luego, cargar e instalar applets en la tarjeta.
- Gestión de las características de cifrado de la tarjeta.
- Envío de comandos de alto nivel APDU o comandos definidos por el usuario a una tarjeta de destino y la observación de las respuestas.
- Mostrar y registrar un seguimiento de toda la actividad entre el programa y la tarjeta.
- Mostrar el contenido de una tarjeta.
- Permitir grabar y reproducir secuencias de comandos para automatizar la emisión de secuencias de comandos.

Se usará JCardManager como una aplicación cliente para probar y depurar un applet específico.

1.5.5 Lenguaje Programación C#

C# es un lenguaje de programación diseñado para crear un amplio número de aplicaciones empresariales que se ejecutan en .NET Framework. El código creado mediante C# se compila como código administrado, lo cual significa que se beneficia de los servicios de Common Language Runtime. Estos servicios incluyen interoperabilidad entre lenguajes, recolección de elementos no utilizados, mejora de la seguridad y mayor compatibilidad entre versiones [23-25].

Se utiliza este lenguaje por la sencillez que presenta eliminando muchos elementos que otros lenguajes utilizan, que en .Net son innecesarios, por ser un lenguaje orientado a objetos simple, elegante y con seguridad en el tratamiento de tipos, proporciona la capacidad de generar componentes de sistema duraderos en virtud de las siguientes características:

- Gran robustez, gracias a la recolección de elementos no utilizados (liberación de memoria) y a la seguridad en el tratamiento de tipos.
- Seguridad implementada por medio de mecanismos de confianza intrínsecos del código.
- Plena compatibilidad con conceptos de metadatos extensibles.

1.5.6 SmartCard Framework

El SmartCard Framework (Figura 6 tomado de [26]) es un marco de trabajo para el desarrollo de middlewares y aplicaciones clientes sobre la tecnología Microsoft .Net Framework o Mono .Net Framework, que permitan la interacción con tarjetas inteligentes, sus aplicaciones y los lectores para establecer la comunicación con estas. Actualmente está diseñando para los lectores compatibles con PC/SC, pero tiene soporte para conectar fácilmente API propietario para la comunicación.

Figura 6 Estructura del SmartCard Framework.

Este marco se está desarrollando actualmente para el framework. NET 4.0. Las consideraciones de diseño se están tomando para que las personas puedan usar y reemplazar componentes de la estructura que necesitan y no se preocupen de hacer funciones para PC/SC, el MCP[13], y otras variantes de API. Por

[13]MCP: es un acrónimo de Multimedia Personal Computer y pertenece a la categoría Hardw are.

ejemplo, si quieren añadir soporte para nuevas capacidades de la tarjeta SIM, se puede hacer eso con el SDK [23, 26].

Se utiliza este framework por ser un modelo de seguridad sobre el cual se sustentan todos los protocolos de canal seguro y mecanismos de autenticación antes mencionados, con posibilidad de ser extendidos hacia otros mecanismos definidos para escenarios y aplicaciones específicas. Además del ser el framework implementado en el departamento.

1.5.7 .NET Framework

.NET es un framework de Microsoft que hace énfasis en la transparencia de redes, con independencia de plataforma de hardware, y permite un rápido desarrollo de aplicaciones. 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. Su propuesta es ofrecer una manera rápida y económica, a la vez que sea segura y robusta para el desarrollo de aplicaciones o 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 [27].

Se utiliza esta herramienta por la sencillez que tiene a la hora de programar, porque simplifica la interoperabilidad entre lenguajes y las máquinas de Microsoft Windows en especial, y por último, es un modelo de programación único para todo tipo de aplicaciones y dispositivos de hardware.

1.5.8 Visual Studio .NET

Es un entorno de desarrollo integrado para sistemas operativos Windows. Soporta varios lenguajes de programación tales como C++, C#, 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, así como servicios web en cualquier entorno que soporte la plataforma .NET [28].

Otra de las funcionalidades del entorno de desarrollo es que permite la ejecución de los programas paso a paso e incluir puntos de interrupción diversos, además de poder analizar el contenido de las variables, a medida que se está ejecutando la aplicación. Por otro lado permitirá en este proyecto un fácil manejo del SmartCardFramework que está implenentado sobre este IDE de desarrollo y en el lenguaje de programación C#.

1.5.9 VisualStudio.QualityTools.UnitTestFramework

El Visual Studio Unit Testing Framework describe suite de herramientas de pruebas unitarias de Microsoft, tal como consta en algunas versiones de Visual Studio 2005 y versiones posteriores. El marco de pruebas unitarias se define en Microsoft.VisualStudio.QualityTools.UnitTestFramework.dll. Las pruebas unitarias

creadas en el marco de pruebas unitarias se pueden ejecutar en Visual Studio o utilizando MSTest.exe, desde una línea de comandos [29].

En este proyecto nos facilita la creación de los proyectos de pruebas y nos genera la estructura básica que tiene que tener la prueba.

1.5.10 Lenguaje Java

Java es un lenguaje de programación y la primera plataforma informática creada por Sun Microsystems en 1995. Es un lenguaje moderno, de alto nivel, que recoge los elementos de programación que típicamente se encuentran en todos los lenguajes de programación, permitiendo la realización de programas profesionales [30].

En este trabajo se utiliza por la rapidez, seguridad y fiabilidad que presenta este lenguaje. Aparte de que contiene importantes mejoras para el rendimiento, estabilidad y seguridad de las aplicaciones.

1.5.11 NetBeans

NetBeans es una plataforma para el desarrollo de aplicaciones de escritorio usando el lenguaje Java y un entorno de desarrollo integrado (IDE) para desarrollar bajo esta plataforma, pero también admite otros lenguajes de programación como C y C++ mediante los cuales se pueden crear aplicaciones gráficas, por ejemplo usando librerías como wxWidgets [31]. Es sumamente completa, fácil de usar, cómoda y de excelente calidad; y es completamente gratis. Es muy famosa entre los programadores de java hoy en día, por lo que hay mucha información al respecto.

En este proyecto se utiliza por ser de código abierto y por presentar una ventaja adicional sobre los demás IDE de desarrollo de la tecnología Java, al ser desarrollado por la compañía Sun, la misma que creó Java, lo que puede descargarse e instalarse en un mismo paquete con el kit de desarrollo JDK, lo cual simplificará su instalación a los programadores.

1.5.12 JUnit

JUnit es un “framework” para automatizar las pruebas unitarias de aplicaciones Java. Se utiliza en la fase de desarrollo, es una herramienta de software libre, por lo cual se puede extender. Su utilización por parte de los desarrolladores permite la creación de software de mayor calidad. Permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera [32, 33].

Se utiliza esta herramienta en este proyecto para llevar a cabo las pruebas de unidad en los métodos que se desarrollaron de manera automática, encargándose esta herramienta de crear los casos y clases de pruebas.

1.5.13 Servidor Web - TomCat

Un servidor web es aquel que almacena principalmente documentos HTML (son documentos a modo de archivos con un formato especial para la visualización de páginas web en los navegadores de los clientes), imágenes, videos, texto, presentaciones, y en general todo tipo de información. Además se encarga de enviar estas informaciones a los clientes.

El presente proyecto ha escogido Apache, también conocido como simplemente Tomcat o Jakarta Tomcat, que es un servidor web multiplataforma que funciona como contenedor de servlets, que implementa las especificaciones de los servlets y de JavaServer Pages o JSP de Sun Microsystem [34].

Se utiliza, dado que dicho producto, fue desarrollado en Java, éste puede ejecutarse sobre cualquier sistema operativo, puede funcionar como servidor web por sí mismo.

1.5.14 Axis2

Apache Axis2 es un contenedor para servicios web. Es un rediseño total y una reimplementación completa de la ampliamente difundida pila SOAP "Apache Axis". Esta nueva arquitectura, en la que se basa Axis2, es más flexible, eficiente y configurable en comparación con la de Axis1.x.

Axis originalmente se convirtió en uno de los contenedores más extendidos para implantar soluciones basadas en Web Services o que proveían acceso a sistemas preexistentes mediante Web Services SOAP. Con la aparición de Axis2 y sobre todo con el aumento de la documentación y mejora de estabilidad de las versiones iniciales, Axis2 se ha convertido en la solución de referencia para construir Web Services y desplegarlos bajo los requisitos más exigentes [34, 35].

Este framework provee la capacidad de agregar servicios web a las aplicaciones web, funcionar como servidor autónomo, además de incluir su propia implementación de referencia.

1.5.15 Servidor Base Datos - PostgreSQL

Un servidor de base de datos es aquel que da servicios de almacenamiento y gestión de bases de datos a sus clientes. Por ejemplo, todos los datos de los clientes de un banco y sus movimientos en las cuentas.

El presente proyecto ha escogido PostgreSQL como lenguaje de base de datos, ya que constituye un lenguaje normalizado con un código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando [36].

1.5.16 pgAdmin

PgAdmin 3 es una herramienta de código abierto para la administración de bases de datos PostgreSQL y derivados. Permite a los usuarios escribir desde simple consultas SQL hasta desarrollar bases de datos complejas. Su interface gráfica soporta todas las características de PostgreSQL y hace simple la administración. Está disponible en más de una docena de lenguajes y para varios sistemas operativos, incluyendo Microsoft Windows, Linux, FreeBSD, Mac OSX y Solaris [36, 37].

Se ha escogido esta herramienta por la edición rápida de consultas que posee, tiene soporte para todos los tipos de objetos de PostgreSQL, por lo que se hace útil como herramienta para hacer las consultas de la base de datos.

1.6 Propuesta y Selección de las Herramientas

Para el desarrollo de los componentes de la plataforma que permita extender la capacidad de procesamiento y memoria de los applets JavaCard en tarjetas inteligentes, se ha seleccionado la Plataforma .Net como un conjunto de tecnologías dispersa, que permite simplificar el desarrollo de la aplicación, proporcionar un entorno de ejecución robusto, seguro, y que brinde la posibilidad de interoperabilidad del código existente.

Se determina que el API JavaCard se realiza con el IDE de desarrollo Devolper Suite el cual proporciona un conjunto de herramientas para crear y depurar applets Java Card, además de poseer integrado el simulador JCardManager, una herramienta con una interfaz intuitiva que ayudará a acelerar el trabajo permitiendo probar las funcionalidades del applet antes de cargarlo en una tarjeta. Para demostrar las necesidades del sistema y diseñar la solución propuesta se utilizará la Metodología ágil XP ya que es utilizada en proyectos de cortos plazo, por pequeños equipos de desarrollo y cuyo plazo de entrega es relativamente corto, además de que posee capacidad de respuesta ante los cambios. Para implementar uno de los componentes se utilizará el lenguaje de programación 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 PC/SC y reduce el costo y el tiempo de desarrollo de los servicios .Net.

Se modelará la aplicación utilizando la herramienta Altova UModel, ya que por su fácil manejo hace que sea práctico para todos los programadores y gestores de proyecto, además que es la definida por el departamento de tarjetas inteligentes. Para el servicio web se utiliza Apache-Tomcat-7.0.23 porque es uno de los servidores web más usados en el mundo, es multiplataforma, además de que esta desarrollado sobre el lenguaje Java.

Para el almacenamiento y gestión de bases de datos de la información de la tarjeta se utiliza el servidor de base de datos PostgresSQL, ya que posee un código fuente libre y de alta calidad, es multiplataforma, además de ser fiable y de estabilidad tradicional. Finalmente para las administración de la base de datos

se hará uso de la herramienta pgAdmin que tiene soporte sobre PostgresSQL y permite escribir simples consultas SQL; cumpliendo de esta manera con los requisitos necesarios para la implementación de la solución.

1.7 Conclusiones del Capítulo

Con el análisis de los principales conceptos y tecnologías relacionadas con tarjetas inteligentes, además de llevar a cabo una investigación de las soluciones existentes que resolvían el problema de memoria y capacidad de las tarjetas Java, se logró elaborar el marco teórico de la investigación.

El estudio de las herramientas, lenguajes, tecnologías, metodología y estándares, permitió obtener una valoración crítica la cual asevera que las identificadas por los desarrolladores son las más adecuada para la propuesta de la solución. Al mismo tiempo que garantizó al usuario llevar a cabo una o varias tareas específicas que debía cumplir el sistema.

CAPÍTULO 2: PROPUESTA DE LA SOLUCIÓN

2.1 Introducción

La solución a desarrollar debe ser fruto de un correcto análisis y una amplia comprensión de todos los elementos que se relacionan en correspondencia al tema de los componentes que interactúan en el proceso, así como de un profundo estudio de las características que posibiliten desarrollarlos.

En este capítulo se hace un estudio de los principales aspectos en que se enmarca el problema, concluyendo que se debe realizar una modelación del dominio, identificando para esto las entidades principales que se tendrán, y las relaciones entre ellas. Para guiar el proceso de desarrollo de software se utilizará la metodología de desarrollo XP y se interpretarán las necesidades del sistema mediante la especificación de los requerimientos funcionales y los no funcionales.

Asimismo, se expone el avance de la solución durante las fases iniciales de Planificación y Diseño, donde se plantean de manera precisa 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, 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

Se realiza un modelo de dominio como punto de partida para el diseño del sistema. Se utiliza para mostrar de manera visual los principales conceptos que se manejan, ayudando a los usuarios, desarrolladores e interesados a emplear un vocabulario común para poder entender el contexto en que se desarrolla el sistema. También contribuirá a identificar personas, eventos, transacciones y objetos involucrados en el sistema. En la figura 7 se muestra el diagrama de clases del modelo de dominio donde se desarrollara el sistema.

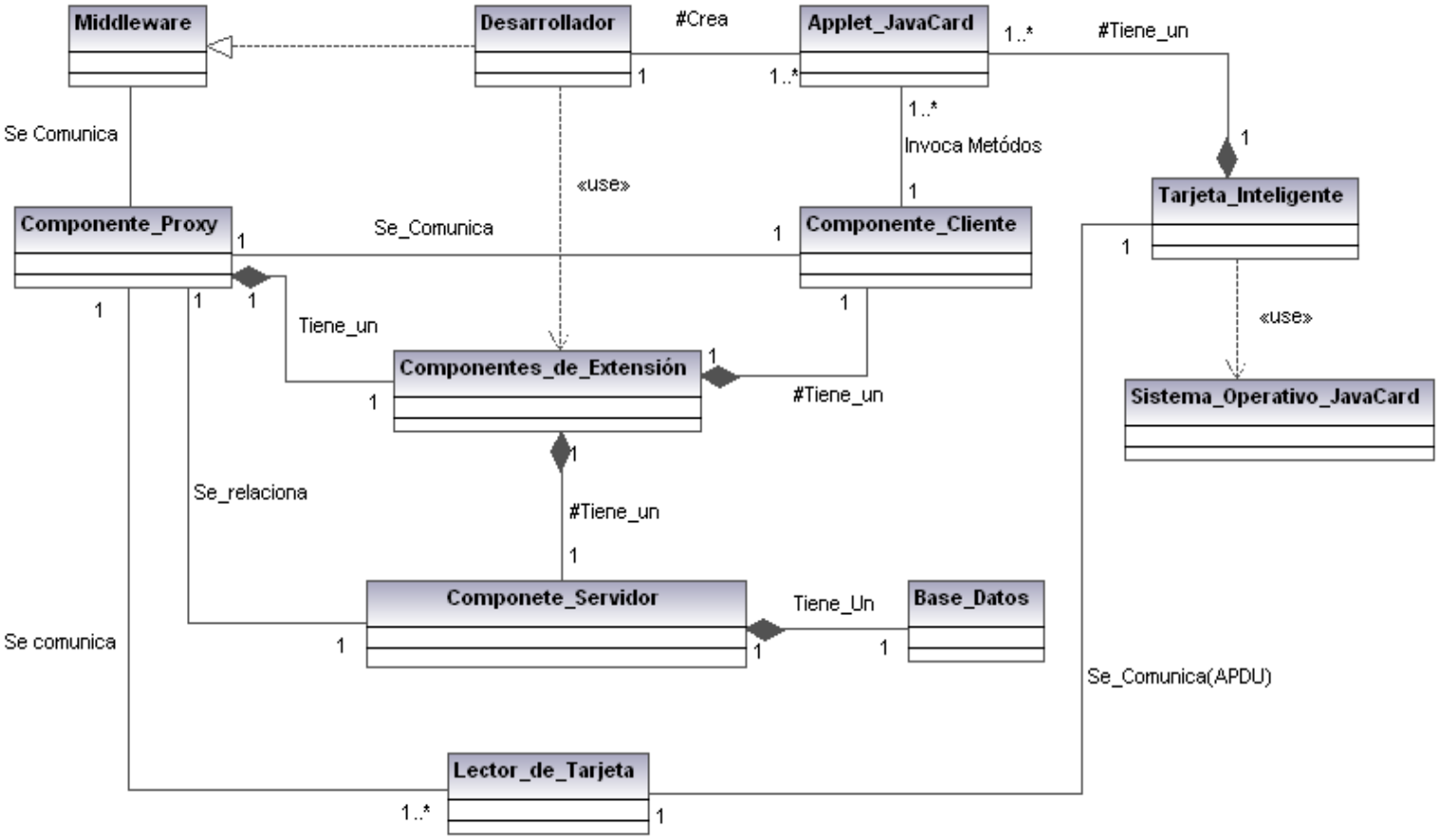

Figura 7 Diagrama de clases del modelo de dominio.

2.2.1 Glosario de Conceptos del Modelo de Dominio

API Java Card: conjunto de todas las clases en que se llevará a cabo las operaciones de extensión.

Applet Java Card: aplicaciones implementadas utilizando la tecnología Java Card, ejecutada dentro de las tarjetas inteligentes. Encargadas de ejecutar las operaciones que maneja mediante los comandos APDUs que le son enviados, retornando los resultados mediante APDU respuestas.

Base Datos: medio donde va a persistir la información de las operaciones de extensión implementadas en el servidor (dígase identificador, parámetros que devuelve).

Componente Cliente: encargado de abstraer al desarrollador del envío y recepción de los comandos que se intercambian entre el componente Proxy y el applet en cuestión, durante la ejecución de operaciones de extensión.

Componente Proxy: encargado de realizar todas las operaciones de comunicación entre el componente JCAEC y el JCAES, abstrayendo al middleware a realizar solo operaciones necesarias para su comunicación con el applet o los applets que necesita.

Componente Servidor: es el encargado de ejecutar las operaciones de extensión y guardar el resultado del mismo.

Desarrollador: encargado de programar el middleware y el applet, según la necesidad del mismo.

Lectores de Tarjetas PC/SC: es un lector compatible con el estándar PC/SC, el cual define la comunicación entre el applet y el proxy, sin realizarle modificaciones a ambas aplicaciones.

Middleware: encargado de enviar un comando APDU con la instrucción que se desea realizar.

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.

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

2.3 Descripción del Sistema

La presente solución ha sido diseñada con tres componentes:

El componente JavaCardAppletExtensionClient(JACEC), recibe los comandos APDU enviados del JACEP y se encarga de procesarlos para conformar la respuesta de la petición realizada con los datos de la respuesta final o la información necesaria para conformar la operación de extensión.

El componente JavaCardAppletExtensionServer(JACAES), servidor de extensiones encargado de recibir los pedidos para ejecutar operaciones fuera de la tarjeta, utilizando la *extensión* específica y de devolver el resultado al JCAEP.

El componente JavaCardAppletExtensionProxy (JCAEP) encargado de comunicarse con el applet mediante el lector y con el servidor de extensiones (JCAES). Sus funciones son invocar la solicitud sobre una operación a realizar y servir de intermediario para dirigir las transmisiones de APDU.

En la siguiente figura se muestra un diagrama de comunicación que refleja de manera concreta la relación entre estos componentes y sus funciones.

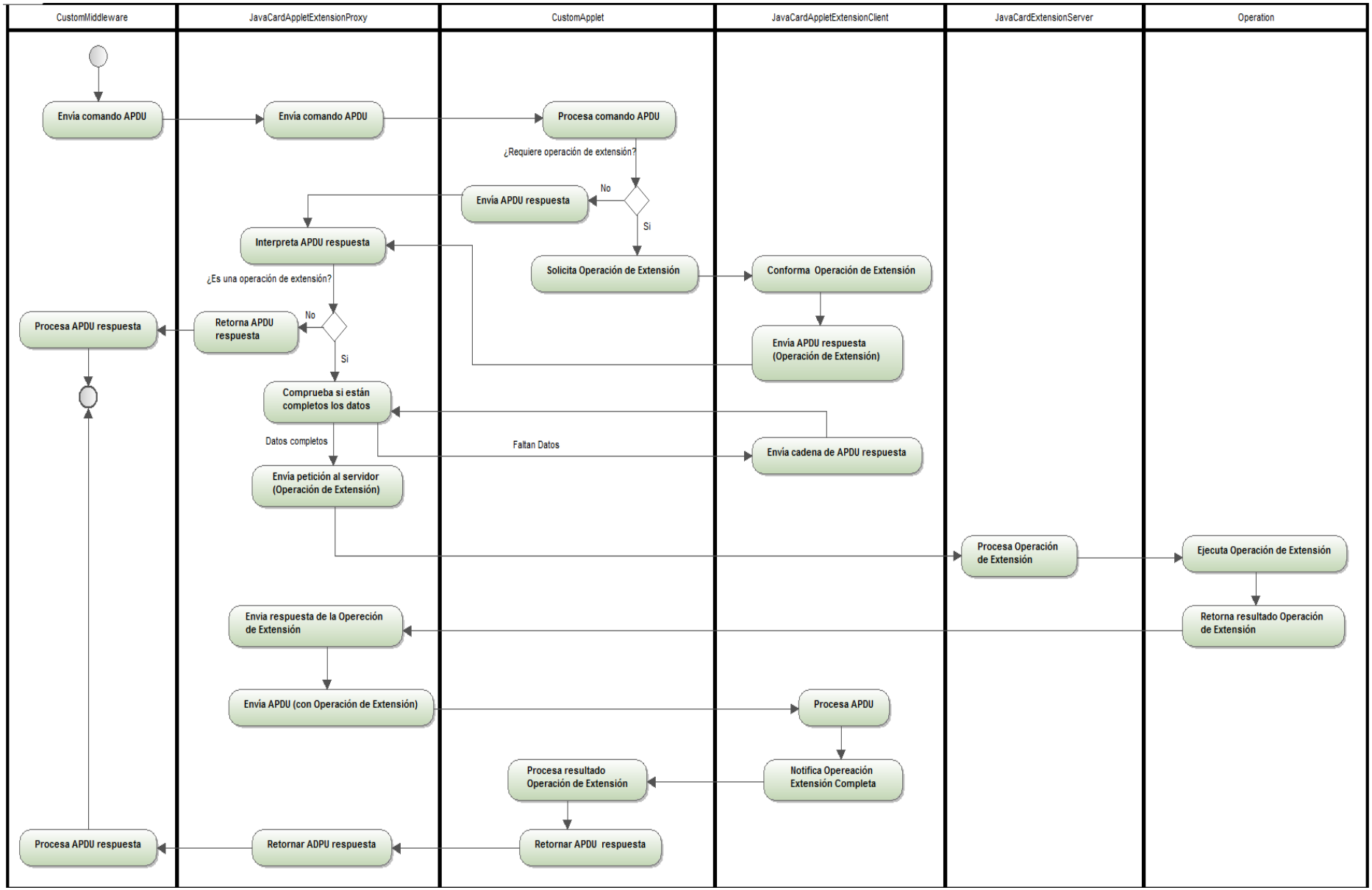

Figura 8 Flujo de comunicación de los componentes de la plataforma de extensión.

El flujo de comunicación inicia con el envió de un comando APDU del *CostumMiddleware*[14] al componente JCAEP que reenvía este comando a la tarjeta. El *CostumApplet* de la tarjeta se encarga de procesar el comando APDU recibido para ver si es una operación de extensión, en caso de no serlo elabora una respuesta final y se la envía al componente JCAEP que interpreta si el APDU respuesta recibido de la tarjeta es una operación de extensión y en caso de no serlo le envía la respuesta obtenida de la tarjeta al middleware.

En caso de ser una operación de extensión el componente JCAEC se encarga de conformar esta respuesta con los datos necesarios para la creación de una operación de extensión. Esta respuesta es enviada al componente JCAEP que se encarga de interpretar esa respuesta para ver si es una operación de extensión, en caso de serlo se encarga de comprobar si están todos los datos para llevar a cabo la operación de extensión. En caso de faltar datos, el componente JCAEC envía cadenas de APDUs respuesta con los datos.

Luego de comprobado los datos si no falta ninguno se le envía la petición de la operación de extensión al componente JCAES, este se encarga de procesar esta petición y el componente *Operation* se encarga de ejecutarla, las operaciones en el servidor pueden ser de almacenamiento o de ejecución de operaciones de mucho costo computacional que por el tiempo de ejecución y respuesta es mucho más rápido y factible que en la tarjeta, por lo que es necesario realizarlas fuera de estas. Al instante de realizadas las operaciones, el componente JCAES se encarga de enviar la respuesta al componente JCAEP y este de reenviarla al componente JCAEC.

Una vez recibida la respuesta de la operación de extensión el componente JCAEC se encarga de procesarla para ver a quien pertenece la operación y notificar que la operación de extensión ha sido completada. Este se la reenvía al componente *CostumApplet* que se encarga de procesar la respuesta y enviarla al componente JCAEP y este de reenviarla el middleware concluyendo el flujo de comunicación.

2.4 Especificaciones de los Requerimientos del Software

2.4.1 Requerimientos Funcionales

Los requerimientos funcionales especifican acciones que el sistema debe ser capaz de realizar, sin tomar en consideración ningún tipo de restricción física, es decir, especifican el comportamiento de entrada y salida del sistema, y surgen de la razón fundamental de la existencia del producto [38].

Con el objetivo de comprender que se espera que haga el sistema por parte del usuario se definieron los siguientes requerimientos para los componentes de la plataforma:

[14] Estos componentes en cursiva no son parte de la solución pero si en algún momento dado lo crea el desarrollador que utilizará esta plataforma para llevar a cabo las operaciones de extensión. En este proyecto se crearon con el objetivo de probar el funcionamiento de los componentes.

Componente JCAEP

- RF1 Realizar conexión.
- RF2 Obtener lectores disponibles.
- RF3 Recibir comando APDU.
- RF4 Enviar comando APDU a la tarjeta.
- RF5 Interpretar comando APDU.
- RF6 Enviar petición de extensión al servidor
- RF7 Enviar APDU respuesta de operación de extensión a la tarjeta.
- RF8 Enviar APDU respuesta al middleware.
- RF9 Realizar desconexión.

Componente JCAEC

- RF10 Recibir comando APDU.
- RF11 Conformar respuesta de la petición realizada.
- RF12 Enviar APDU respuesta.
- RF13 Enviar cadena de APDU respuesta.
- RF14 Recibir comando APDU de extensión.
- RF15 Procesar resultado de la operación de extensión.

Componente JCAES

- RF16 Ejecutar operación de extensión.
- RF17 Almacenar información de la tarjeta.
- RF18 Enviar respuesta de la petición de extensión.

2.4.2 Requerimientos No Funcionales

Los requerimientos no funcionales son las propiedades o cualidades que el producto debe tener para que este sea atractivo, usable, rápido y confiable [38].

Requerimientos de software

- Para la prueba funcional a los componentes se requiere que estén instalados en la PC cliente los drivers del lector de tarjetas y el marco de trabajo SmartCard Framework.
- En la PC cliente se podrá utilizar Microsoft Windows en cualquiera de sus versiones y en las distintas distribuciones de Linux donde pueda ejecutarse MonoDevelop.

Requerimientos de hardware recomendados para el desarrollador

- Lector de tarjetas incorporado a la PC que cumpla con el estándar PC/SC versión 1.0 o superior.
- Requerimientos recomendados para el Sistema Operativo Windows XP Professional SP3

Requisitos	**Máquina Virtual**	**Visual Studio**	**NetBeans**	**Servidor Tomcat con Axis**	**Necesidad**
Procesador	Intel Pentium IV o equivalente	Intel Pentium IV o equivalente	Intel Pentium IV o equivalente	Intel Pentium	Intel Pentium IV o equivalente
Velocidad del Procesador	1 GHz	1 GHz	2.6 GHz	700MHz	3 GHz
Espacio en disco	2 GB	6 GB	1 GB	1GB	50 GB
Memoria RAM	256 MB	256 MB	1 GB	300 MB	1GB

Tabla 3 Requisitos recomendados para el Sistema Operativo Windows XP Professional SP3

Requerimientos de diseño e implementación

- Framework de desarrollo que se utiliza es: .NET Framework, SmartCardFramework.
- IDE: Visual Studio 2005 SDK 1.1.

Rendimiento

- La aplicación está concebida para ser utilizada por los desarrolladores para facilitar su trabajo, por lo que los tiempos de respuestas deben ser rápidos, así como la velocidad de procesamiento de información.

Portabilidad

- El sistema debe ser compatible con cualquier lector de tarjetas que cumpla con el estándar PC/SC.

2.5 Planificación

El propósito de la fase de planificación es establecer un acuerdo entre los clientes y desarrolladores sobre el menor tiempo en que la mayor cantidad de historias de usuarios puedan ser realizadas. Por ello se realizó la planificación con el objetivo de maximizar el valor del software producido a partir de la puesta en producción de las características más importantes lo antes posible, se realiza una estimación del esfuerzo requerido para la implementación de las HU y se define el tiempo de entrega y cada iteración [18, 39].

2.5.1 Historia de Usuario

Son la técnica utilizada en XP para especificar los requisitos del software. De una manera simple describen una tarea breve que aporta valor al usuario o al negocio ya sean requisitos funcionales como no funcionales. Se usaron con el objetivo de potenciar la participación del equipo en la toma de decisiones, persiguiendo el principio básico de las metodologías agiles.

Cada historia de usuario debe ser comprensible y estar delimitada, de forma que los programadores puedan realizarla en unas semanas [39]. A continuación se muestran las historias de usuarios más importante de cada componente el resto está en el Epígrafe Anexo. (Ver Anexo 1: Historia de Usuario)

Componente JCAEP

Historia de Usuario	
Número: HU_4	**Usuario:** Desarrollador
Nombre historia: Enviar un comando APDU a la tarjeta	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Amed Alfonso Ríos	
Descripción: El componente JCAEP envía los correspondientes comandos APDU con una solicitud de operación a un applet específico, utilizando las funcionalidades que brinda el SmartCardFramework para la interacción con tarjetas inteligentes.	
Observaciones: Si la tarjeta no está en estado conectada en el lector, el sistema lanza una excepción. Si el desarrollador no está autenticado el sistema lanza una excepción. Si la clase o la instrucción del comando APDU enviado son incorrectas el sistema lanza una excepción.	

Tabla 4 HU_4 Enviar un comando APDU a la tarjeta

Historia de Usuario	
Número: HU_5	**Usuario:** Desarrollador
Nombre historia: Interpretar comando APDU.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Amed Alfonso Ríos	
Descripción: Interpreta si el APDU respuesta, obtenido del componente JCAEC solicita una	

operación de extensión o no, mediante la comparación del primer byte de la respuesta con la clase AE definida para las operaciones de extensión[15]. En este caso se comprueba que en el APDU respuesta se encuentren todos los datos.
Observaciones:

Tabla 5 HU_5 Interpretar comando APDU.

Componente JCAEC

Historia de Usuario	
Número: HU_10	Usuario: Desarrollador
Nombre historia: Recibir un comando APDU.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Julié Arianne Pérez Vive	
Descripción: Este recibe un comando APDU con toda la información enviada por el componente JCAEP y realiza una comparación entre la clase del APDU y la clase 00 definida para la operación de extensión e instrucción AE, para ver si el comando enviado es una operación para el componente JCAEC o para el applet en cuestión.	
Observaciones: Si la clase o la instrucción no son correctas el sistema envía un error de clase o instrucción no soportada.	

Tabla 6 HU_10 Recibir un comando APDU.

Historia de Usuario	
Número: HU_11	**Usuario:** Desarrollador
Nombre historia: Conformar respuesta de la petición realizada.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Julié Arianne Pérez Vive	
Descripción: Se conforma el TLV[16] para dar respuesta a la petición realizada con los datos de la respuesta final o la información necesaria para conformar la operación de extensión.	

[15] Véase Capitulo 3: Implementación y Prueba. Definición de las instrucciones y estructuras

[16] Véase Capitulo 3: Implementación y Prueba. Definición de las instrucciones y estructuras

TLV: se refiere al formato Tipo-Longitud-Valor o Type-Length-Value, usado para representar información, de forma que haya información que pueda tener presencia opcional y longitud variable.

Observaciones:

Tabla 7 HU_11 Conformar respuesta de la operación a realizar.

Historia de Usuario	
Número: HU_15	**Usuario:** Desarrollador
Nombre historia: Procesar resultado de la operación de extensión.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Amed Alfonso Ríos	
Descripción: Al recibir el APDU respuesta de la operación de extensión este busca por el identificador de la misma a que clase del applet corresponde la operación de extensión recibida.	
Observaciones:	

Tabla 8 HU_15 Procesar resultado de la operación de extensión.

Componente JCAES

Historia de Usuario	
Número: HU_16	**Usuario:** Desarrollador
Nombre historia: Ejecutar operación de extensión.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Amed Alfonso Ríos	
Descripción: Ejecuta la operación de extensión con los datos recibido del componente JCAEP mediante un arreglo de byte, que contiene el identificador y los parámetros necesarios para la ejecución de la operación. Analiza el fichero de configuración, donde se encuentra la dirección de los archivos .jar y el nombre de las clases; para que sea posible ejecutar dinámicamente las cargas de las clases.	
Observaciones:	

Tabla 9 HU_16 Ejecutar operación de extensión.

Historia de Usuario	
Número: HU_17	Usuario: Desarrollador
Nombre historia: Almacenar información de la tarjeta.	

Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Amed Alfonso Ríos	
Descripción: En caso de ser necesario para el desarrollador se almacena en la Base de Datos del componente JCAES cualquier información asociada a la tarjeta definida por el desarrollador (dígase identificador, atributo y valor).	
Observaciones:	

Tabla 10 HU_17 Almacenar información de las operaciones de extensión.

Historia de Usuario	
Número: HU_18	**Usuario:** Desarrollador
Nombre historia: Enviar respuesta de la petición de extensión.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Julié Arianne Pérez Vive	
Descripción: Envía la respuesta al componente JCAEP con la información de la operación, que puede ser el resultado de un algoritmo procesado o una información almacenada, en un arreglo de byte el cual contiene en su primera posición el identificador de la operación de extensión asociada, seguido de los datos de la respuesta obtenida.	
Observaciones:	

Tabla 11 HU_18 Enviar respuesta de la petición de extensión.

2.5.2 Estimación de Esfuerzo

Para el desarrollo de la aplicación se realizó una estimación de esfuerzo por cada una de las historias de usuario identificadas por el cliente. Las estimaciones de esfuerzo asociado a la implementación se determinan utilizando como medida la cantidad de puntos. Se le dan valores de 1 a 3 puntos donde un punto equivale a una semana de programación [18, 39].

Especificando un tiempo estimado para la elaboración de cada una en base a una semana de 5 días y un día de 8 horas. (Ver Anexo 2: Estimación de Esfuerzo).

2.5.3 Plan de Entregas

El plan de entregas especifica exactamente que historias de usuario serán implementadas en cada entrega del sistema y sus prioridades, de modo que también permita conocer con exactitud qué historias de usuario serán implementadas en la próxima liberación. Para la elaboración del plan de entrega de este

proyecto y aplicando los parámetros de desarrollo bajo la metodología XP, se establece el tiempo calendario de acuerdo a un mes de 4 semanas y una semana de 5 días y un día de 8 horas. (Ver Anexo 3: Plan de Entrega).

2.5.4 Plan de Iteraciones

La planificación en iteraciones y el diseño iterativo permite mantener discusiones continuas acerca de los problemas y avances realizados en las tareas, fomentando así la comunicación entre las partes involucradas y ayudando a la optimización de tiempo y esfuerzo empleado por el correcto desarrollo del proyecto [18, 40]. Para la elaboración del plan de iteración de este proyecto, se leyeron y detallaron las historias de usuarios, a las que luego se le asignaron tareas en cada una de las iteraciones que se van a llevar a cabo, dando como resultado de cada iteración la entrega de un módulo, habiendo superado éste las pruebas de aceptación establecidas por el cliente de forma que verifique los requisitos (Ver Anexo 4: Plan de Iteraciones).

2.5.5 Tareas de Ingeniería

Todo el trabajo de la iteración es expresado en tareas de programación, las cuales se realizan para especificar las acciones llevadas a cabo por los programadores en cada historia de usuario, ya que no ofrecen el nivel de detalle requerido para saber qué implementar. Según el Plan de Iteraciones las historias de usuario se agruparon en tres iteraciones. A continuación se muestran de manera general las tareas de ingeniería derivadas de cada historia de usuario (Ver Anexo 5: Tareas de Ingeniería).

2.6 Diseño

La metodología XP hace especial énfasis en los diseños simples y claros. Sólo se diseñan aquellas historias de usuario que el cliente ha seleccionado para la iteración actual. Kent Beck dice que en cualquier momento el diseño adecuado para el software es aquel que: supera con éxito todas las pruebas, no tiene lógica duplicada, refleja claramente la intención de implementación de los programadores y tiene el menor número posible de clases y métodos[18, 39].

2.6.1 Metáfora

El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema (conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema, ayudando a la nomenclatura de clases y métodos del sistema) [41].

Con el diseño de los componentes se logrará desarrollar una aplicación que permita extender las capacidades de procesamiento y memoria de las aplicaciones Java Card en tarjetas Inteligentes. En

tiempo de codificación la elaboración de un API Java Card (componente JACEC) abstraería al desarrollador del envío y recepción de los comandos APDU que se intercambian entre el middleware y el applet en cuestión durante la ejecución de operaciones de extensión, y en tiempo de ejecución la elaboración de un componente que sea un WrapperPCSC integrado al SmartCardFramework (JCAEP), que se encargue de realizar todas las operaciones de comunicación entre el componente JCAEC y el JCAES, abstraería al middleware a solo realizar operaciones necesarias para su comunicación con el applet o los applets que necesita. En sentido general este sistema podrá ser utilizado por todos los desarrolladores que necesiten realizar cualquier operación en la que la capacidad de la tarjeta no le sea suficiente para ejecutar operaciones dentro de la misma.

2.6.2 Definiciones de las tarjetas CRC

Las tarjetas CRC (Clase o cargo, Responsabilidad y Colaboración), brindan la posibilidad de trabajar con una metodología basada en objetos. Contribuyen en la tarea de diseño y representan escenarios derivados de las historias de usuarios, los cuales especifican las posibles clases de la aplicación e identificando las responsabilidades y colaboraciones para finalmente ser traducidas en métodos y relaciones en sus respectivas iteraciones.

La siguiente tabla muestra la tarjeta CRC del componente JCAEP las otras pueden consultarlas en el epígrafe de Anexo. (Ver Anexo 6: Tarjetas CRC)

Nombre de la clase: JACEProxy.cs	
Tipo de clase: Controladora	
Atributo:	**Tipo:**
_ClienteGlobalPlatform	GlobalPlatformClient
_GenericCardReader	GenericPCSCCardReader
Para cada responsabilidad:	
Nombre:	JCAEProxy()
Descripción:	Constructor de la clase que se encarga de crea el objeto de la clase JCAEProxy().
Nombre:	Send_APDU(APDUCommand Apdu)
Descripción:	Método público encargado de enviar comandos APDUs a la tarjeta, interpreta la respuesta obtenida de la misma para ver si es una operación de extensión o no. En caso de ser una operación de extensión, después de ser procesada el resultado se lo envía a la tarjeta y si no le envía la respuesta al middleware.
Nombre:	TotalConvert_TLV(APDUResponse _Response)

Descripción:	Método privado que construye un arreglo de byte con parámetros enviados de la tarjeta y comprueba si faltan datos por venir de la misma, en ese caso los solicita para completar los datos de los parámetros.
Nombre:	Convert_TLV(APDUResponse _Response)
Descripción:	Método privado que se encarga de convertir un APDU respuesta cualquiera en una arreglo de byte, este es usado por el método TotalConvert_TLV ().
Nombre:	SendRequestServer(byte[]Date)
Descripción:	Método privado encargado de enviar la petición al servidor y obtener la respuesta de la operación de extensión.

Tabla 12 Descripción de la clase JCAEProxy.cs

2.6.3 Diagrama de Clases

Los diagramas de clases sirven para representar la estructura estática de un sistema incluyendo una colección de elementos de modelación estáticos, tales como clases y relaciones [42]. Representan las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. Permite visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y dependencia. Un diagrama de clases está compuesto por los siguientes elementos: Clase: atributos, métodos y visibilidad. Relaciones: Herencia, Composición, Agregación, Asociación y Uso [43].

Diagrama de clase del componente JCAEP

La clase JACEProxy.cs es la clase controladora encargada de comunicarse con el applet mediante el lector y con el servidor de extensiones (JCAES), sus funciones son invocar una llamada sobre una operación a realizar y servir de intermediario para dirigir las transmisiones de APDUs. Se relaciona con la clase ExcepcionJCAEProxy.cs que hereda de la DLL SmartCardExpection que aunque no forme parte de la solución hace uso de la misma para crear las excepciones que lanzaría el sistema en caso de que exista algún error. En la figura 9 se muestra el diagrama de clase del componente JCAEP descrita anteriormente.

Figura 9 Diagrama de clase Componente JCAEP

Diagrama de clase del componente JACEC

La clase CustomAppExtManager.class es la clase controladora encargada de procesar los comandos APDU enviados del componente JACEP, sus funciones son identificar si el comando APDU es de extensión, conformar la respuesta de la petición realizada con los datos de la respuesta final o la información necesaria para conformar la operación de extensión y procesar el resultado de la operación de extensión. Se relaciona con la clase CustomApplet que aunque no forma parte de la solución es creada por el desarrollador para integrar y probar las funcionalidades del sistema. También está relacionada a la clase List.class que pude ser de tipo NodoSE.class o NodoIE.class para poder en ellas guardar las operaciones de extensión y los parámetros de la misma.

Figura 10 Diagrama de clase Componente JCAEC

Diagrama de clase del componente JACES

La clase ServerOnline.java es la clase controladora que se encarga de controlar la ejecución de las operaciones a realizar por el servidor, conformar la respuesta solicitada por el componente JACEP. Además conforma una lista de parámetros con un arreglo de byte recibido desde el componente JCAEP y crea una instancia de la clase Reflaction para ejecutar la operación correspondiente al identificador enviado en el arreglo de byte.

La clase Reflection.java creada con el objetivo de monitorear todas las operaciones se encarga de cargar en tiempo de ejecución las operaciones de extensión que se encuentran en un .jar y ejecutar la operación de extensión correspondiente a la operación solicitada.

La clase MemoryManager.java es la encargada de la comunicación con la Base de Datos.

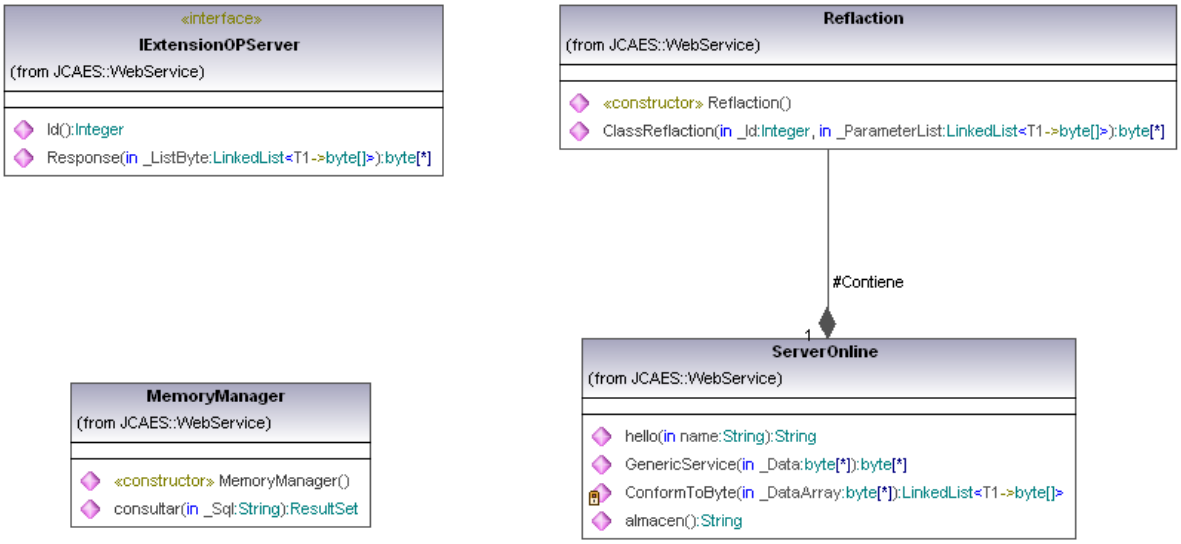

Figura 11 Diagrama de clase Componente JCAEP

2.6.4 Arquitectura del Sistema

El estándar ANSI/IEEE 1471 [44], define la arquitectura del software como la organización fundamental de un sistema descrita en sus componentes, sus relaciones entre ellos y el ambiente y los principios que dictan su diseño y evolución, involucra un conjunto de decisiones significativas acerca de la organización del sistema, posee una selección de sus elementos estructurales y sus interfaces, comportamiento especificado en función de la colaboración de los elementos y una composición de un sub-sistemas más grande a partir de elementos estructurales y elementos con comportamientos [45].

La IMB[17] define al modelo Cliente-servidor**:** como la tecnología que facilita al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o, a través de la organización, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o clientes, resultan en un trabajo realizado por otros computadores llamados servidores [46].

2.6.5 Descripción de la arquitectura

La propuesta de la elaboración de los componentes de la plataforma para la extensión de las capacidades de procesamiento y memoria de los applets Java Card en tarjetas inteligentes basa su implementación en la tradicional arquitectura cliente/servidor, dado que este es un modelo que permite distribuir físicamente los procesos y los datos en forma más eficiente lo que en computación afecta directamente el tráfico de la red, reduciéndolo grandemente, y como arquitectura distribuida permite a los usuarios finales obtener

[17] **International Business Machines** o **IBM** (conocida coloquialmente como **el Gigante Azul**): Es una empresa transnacional que fabrica y comercializa herramientas, programas y servicios relacionados con la informática. IBM tiene su sede en Armonk (Nueva York, Estados Unidos) y está constituida como tal desde el 15 de junio de 1911, pero lleva operando desde 1888.

acceso a la información en forma transparente aún en entornos multiplataformas [46]. Para su funcionamiento se tiene una máquina cliente, que requiere un servicio de una máquina servidor y de la tarjeta, y estas realizan la función para la que están programados. En el caso de esta arquitectura el componente JCAEP se comporta como cliente, manejando todas las funciones relacionadas con la manipulación y despliegue de datos, durante el envío y recepción de información recibida tanto del componente JCAES como del componente JCAEC que se comportan como servidores encargándose de atender las peticiones solicitadas por el cliente; estos envían uno o varios mensajes con la respuesta. La distribución de la arquitectura queda representada de la forma que se muestra en la Figura 12.

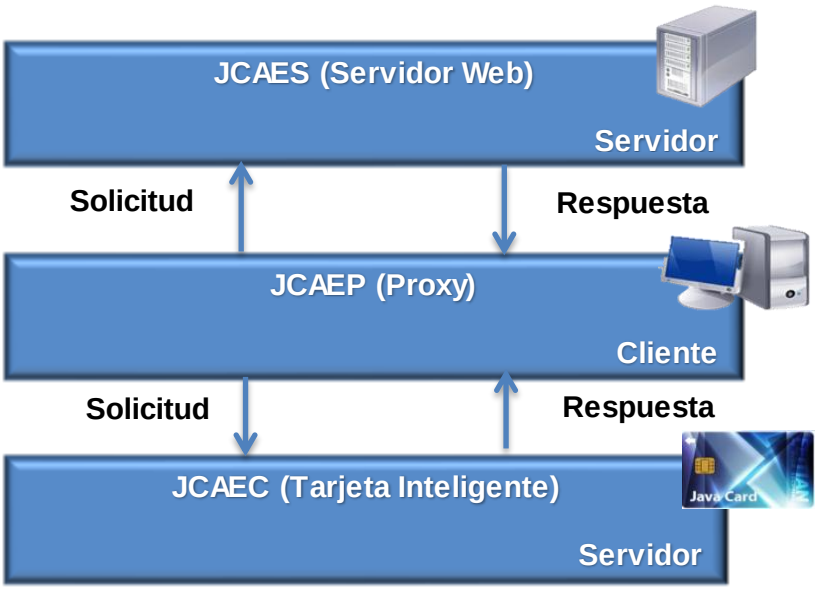

Figura 12 Arquitectura del Sistema

2.6.6 Patrones de Diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software, en otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares, identifican Clases, Roles, Colaboraciones y la distribución de responsabilidades.

Patrones de diseño GRASP: son patrones de principios generales para asignar responsabilidades, describen los principios fundamentales de diseño de objetos y la asignación de responsabilidades, expresados como patrones" [47], constituyen la base del cómo se diseñará el sistema y se aplican en los primeros momentos del diseño.

Para el diseño eficaz del sistema orientado a objeto se utilizan los patrones GRASP:

Experto en Información: permite que una clase contenga toda la información necesaria para implementar una responsabilidad. La clase JCEproxy.cs cumple con la responsabilidad de conocer y dar

toda la información relacionada con el envío y recepción de APDUs para establecer la comunicación entre el middleware y la tarjeta, así como entre la tarjeta y el servidor.

Creador: permite decidir cuáles serán las clases creadoras de otras clases. Este patrón se utiliza en las clases entidades, cuando estas se instancian y crean instancias de clases contenidas en las mismas, ejemplo la clase CustomApplet.class contiene objetos de la clase CustomAppExtManager.class, lo que es idónea para asumir la responsabilidad de crear las instancias de la misma.

Alta Cohesión: permite que una clase contenga responsabilidades moderadas en un área funcional, y colabora con las otras para llevar a cabo las tareas, incrementa la claridad y facilita la comprensión del diseño. Este diseño delega a CustomMiddleware.cs la responsabilidad de enviar el primer comando APDU al proxy inicializando la comunicación.

Bajo Acoplamiento: permite dar soporte a una dependencia escasa y un aumento de la reutilización. Acoplamiento bajo significa que una clase no depende de muchas clases. Se pone de manifiesto en JCEProxyExcepcion.cs que realiza la creación de las excepciones y no la clase JCEproxy.cs, reduciendo la dependencia de este último con el resto de las clases.

Patrones de diseño GoF (Gang of Four): este término, hace referencia a los cuatro autores del libro [48]. Son aquellos que describen la comunicación entre objetos y clases adaptada a la resolución de problemas generales de diseño en un contexto particular. Se centran en una solución a un problema concreto en el ámbito de la programación orientada a objetos, proponiendo una solución genérica de clases y relaciones para resolver dicho problema.

Para la solución implementable del sistema se utilizan los patrones GOF:

Instancia Única (Singlenton): se utiliza cuando solo es requerida una instancia única de un objeto durante el tiempo en que se está ejecutando la aplicación, ya sea porque solo una es eficiente o para evitar conflictos. El mismo fue manejado en varios escenarios, por ejemplo para asegurar que solo haya una instancia de la clase Reflaction en la clase ServerOnline.

Método Factoría (Factory Method): Se le encarga a un método específico la creación de un determinado tipo de objeto en dependencia de una condición dada, así como una fábrica construye distintos tipos de productos. El empleo del mismo se muestra en el método ReflactionClass de la clase Reflaction, que se encarga de construir dinámicamente objetos de otras clases mediante el identificador y el nombre de la clase.

Proxy: Hay situaciones en las que un cliente no referencia o no puede referenciar a un objeto directamente, pero necesita interactuar con él, por lo que se crea un objeto proxy que pueda actuar como intermediario entre el objeto cliente y el objeto destino. El objeto proxy mantiene una referencia al objeto destino y puede pasarle a él los mensajes recibidos (delegación). El mismo fue usado en el componente JCAEP para interactuar entre el middleware, la tarjeta y el servidor.

2.7 Conclusiones del Capítulo

- La realización del modelo de domino determinó las bases brindar una visión más clara de los componentes y los conceptos asociados a la solución propuesta, así como las relaciones entre estos, que son determinados luego del estudio de las herramientas, tecnologías y elementos tangibles asociados al dominio de la solución.
- Los principales artefactos logrados fueron las Historias de Usuario, obteniendo un enfoque más claro y objetivo de los requerimientos del cliente. También se generó el Plan de Iteraciones, y el Plan de Entrega, determinado en 28.1 semanas como tiempo estimado para la realización de la solución, divididas en 3 iteraciones, y estas iteraciones se expresaron en tareas de programación, para ofrecer el nivel de detalle requerido para saber qué deberían de implementar los desarrolladores.
- La definición de la arquitectura a utilizar posibilitó la organización de los componentes de la plataforma, sus relaciones entre ellos y el ambiente y los principios que dictan su diseño y evolución,
- Además la realización de una selección de los patrones de diseño tanto para implementar la solución como para diseñarla a nivel de principios generales, mostró a partir de los diagramas de clase que se brinda una solución ya probada y documentada para dar solución al problema planteado.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1 Introducción

En el siguiente capítulo se expondrá las instrucciones y estructuras definidas para realizar las operaciones de extensión. Se realizará un representación física de cómo se implementó la solución, a través de esquemas gráficos, enfocándose en el diagrama de componente y diagrama de despliegue de la solución desarrollada. Además de que se validará y verificará el cumplimiento de los requerimientos estipulados, mediante la aplicación de métodos de pruebas que garanticen la calidad del sistema.

3.2 Definición de las Instrucciones y Estructuras

Para dar solución al sistema que realiza las operaciones de extensión se definieron las siguientes estructuras:

- **Comando APDU**

La siguiente estructura del comando APDU se creó con el objetivo de conocer si la petición enviada necesita una operación de extensión.

CLA	INS	P1	Lc	Le	Data
00	AE	00	lc	le	Data

Tabla 13 Estructura del APDU respuesta

Donde:

CLA: 00 clase definida para una operación de extensión.

INS: AE instrucción definida para una operación de extensión.

P1: Parámetro 01 para pedir más datos a la tarjeta y parámetro 02 para enviar la respuesta de la operación de extensión.

Lc: Longitud de los datos enviados (Si es 00 es que no hay datos para enviar).

Le: Longitud de los datos que se esperan recibir (Si es 00 no se sabe la cantidad de datos a recibir)

Data: Se envía un arreglo de byte con los datos necesarios.

- **Estructura TLV**

La siguiente estructura TLV se crea con el objetivo de dar respuesta a la petición realizada con los datos de la respuesta final o la información necesaria para conformar la operación de extensión.

AE	Length	Data							
		AE	Length	IdO	AE	Length	IdCard	AE	Length	Datos

Tabla 14 Estructura del TLV

Donde:

AE: identificador para saber si es una operación de extensión.

Length: longitud de los datos.

Data: contiene los datos que se quieren enviar que son almacenados en una estructura TLV.

IdO: identificador de la operación de extensión.

IdCard: identificador de la tarjeta.

Datos: datos necesarios para realizar la operación de extensión.

- **Fichero XML**

Fichero de configuración que se crea con el objetivo de almacenar la dirección donde se encuentra el .jar con las operaciones de extensión y el nombre de las clases de las mismas.

-<documento>

-<Dirección>

C:-Users-Amed-Documents-NetBeansProjects-Operations_-dist-Operations_.jar

<Operaciones>WebService.almacenar</Operaciones>

<Operaciones>WebService.SumaResponse</Operaciones>

<Operaciones>WebService.Consultar</Operaciones> </Dirección>

</documento>

3.3 Diagrama de Componentes

En los diagramas de componentes se muestran los elementos de diseño de un sistema de software. Un diagrama de componentes admite visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces, así como describir un diseño que se implemente en cualquier lenguaje o estilo. Solo es necesario identificar los elementos del diseño que interactúan con otros elementos del diseño a través de un conjunto restringido de entradas y salidas [49]. Las figuras 13,14 y 15 muestran los diagramas de los componentes de la plataforma.

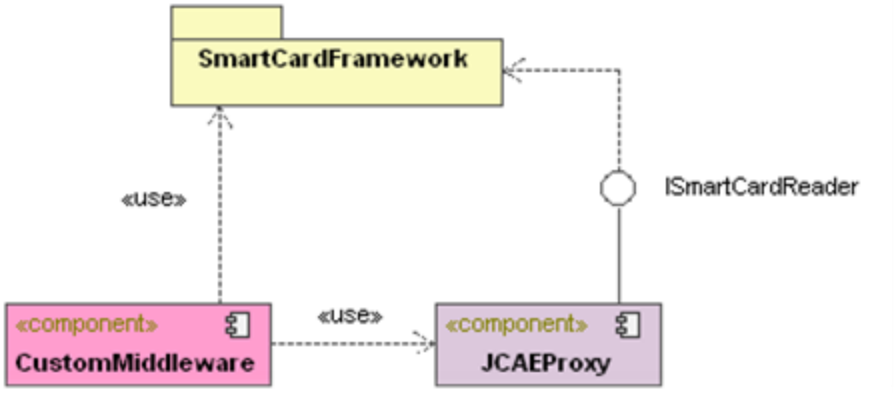

Figura 13 Diagrama de Componente del JCAEP

JCAEProxy: componente encargado de comunicarse con el applet mediante el lector y con el servidor de extensiones (JCAES), sus funciones son invocar las operaciones sobre una operación a realizar y dirigir las transmisiones de APDU.

CustomMiddleware: middleware que realiza el desarrollador para integrar y probar las funcionalidades que realiza el sistema, además de enviar el primer comando APDU para el inicio de la comunicación.

ISmartCardReader: es la interfaz que posee el SmartCardFramework para la comunicación con los lectores.

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

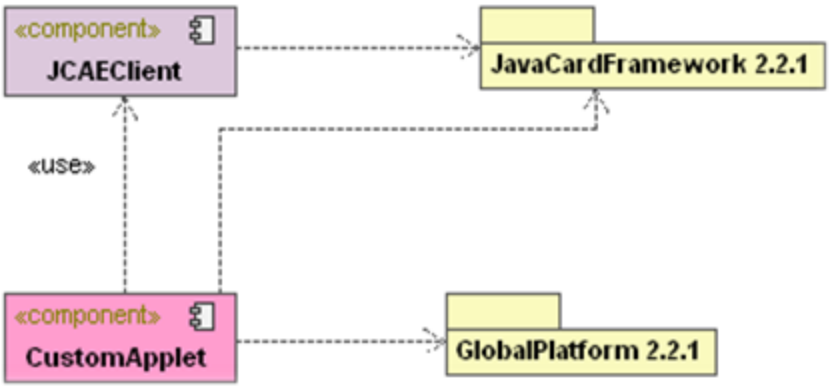

Figura 14 Diagrama de Componente del JACEC

JCAEClient: componente encargado de procesar los comandos APDUs enviados del componente JACEP, sus funciones son identificar si el comando APDU es de extensión, conforma la respuesta de la

petición realizada con los datos de la respuesta final o la información necesaria para conformar la operación de extensión y procesa el resultado de la operación de extensión.

CustomApplet: applet que crea el desarrollador para integrar y probar las funcionalidades del sistema, además en caso de ser una operación de extensión envía los parámetros necesarios al JCAEClient. Es el componente central de una tarjeta con arquitectura GlobalPlatform. Todos los servicios son ejecutados por él y ofrece interfaces para utilizar los servicios, internamente a través de las APIs de GlobalPlatform y externamente a través de los comandos APDUs.

JavaCardFramework: paquete que posee un conjunto de clases necesarias para el funcionamiento del JCRE y JavaCard API, así como clases utilitarias comerciales o extensiones propietarias del fabricante de la tarjeta. Finalmente, haciendo uso de este ambiente se encuentran la creación de los applets.

GlobalPlatform: paquete que posee una series de normas que permiten una infraestructura segura e interoperable para tarjetas inteligentes, dispositivos y sistemas que simplifican y aceleran el desarrollo, despliegue y gestión de aplicaciones en las industrias. Permite el acceso a las funciones del CardManager y autenticar el terminal con la tarjeta, utilizando un canal seguro.

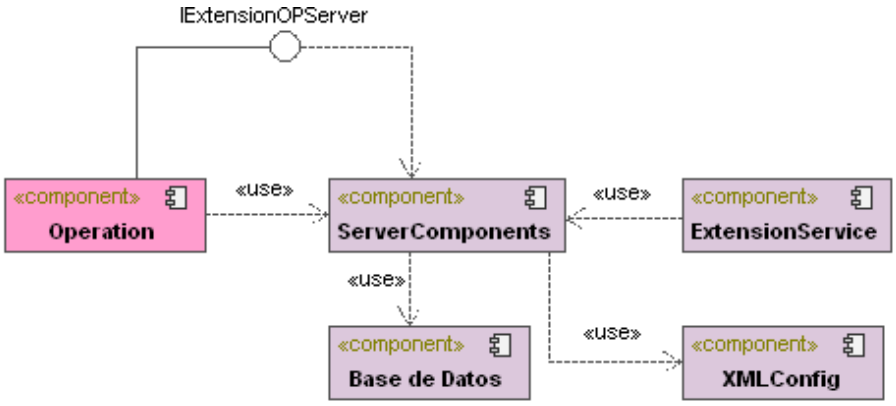

Figura 15 Diagrama de Componente del JACES

ServerComponents: componente que contiene la clase interfaz, MemoryManager y Reflaction. Se encarga de recibir los pedidos para ejecutar operaciones fuera de la tarjeta, utilizando la operación de extensión específica y devuelve el resultado al JCAEP.

Operation: componente que usa el componente ServerComponents y se encarga de ejecutar las operaciones de extensión y de retornar el resultado al JACES

IExtensionOPServer: interfaz que brinda un conjunto de métodos genéricos para ejecutar las operaciones de extensión.

Base de Datos: componente que almacena el identificador de las operaciones, el identificador del applet, los atributos y el valor de la información enviada.

XML Config: componente que almacena la dirección donde se encuentra el .jar con las operaciones de extensión y el nombre de las mismas.

ExtensionService: componente encargado de crear el servicio que se brinda al componente JCAEP.

3.4 Diagrama de Despliegue

Permiten modelar la disposición física o topología de un sistema, muestra el hardware usado y los componentes instalados en el hardware, así como las conexiones físicas entre el hardware y las relaciones entre componentes[50]. Para el modelo de las relaciones que se establecen entre los componentes de software y hardware, se realiza una representación mediante nodos estrechamente conectados. La siguiente figura muestra la distribución física de los componentes de la plataforma a través del diagrama de despliegue

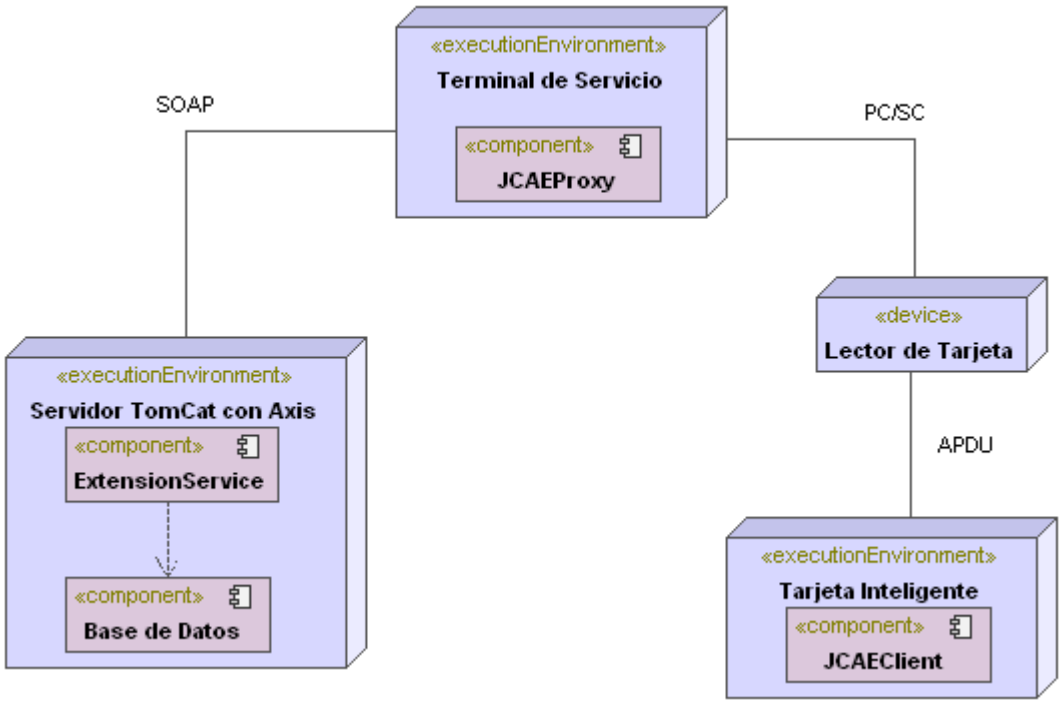

Figura 16 Diagrama de Despliegue

Los componentes creados para la plataforma de extensión de capacidades de procesamiento y memoria de applets Java Card en tarjetas inteligentes deben estar en cada una de los terminales de servicio de los desarrolladores que harán uso de la misma. Las peticiones a las operaciones que se quieran realizar se harán mediante el envío de comando APDUs con los lectores de tarjeta, los lectores se conectan a cada una de las terminales de servicio mediante el estándar PC/SC.

En la terminal de servicio se encuentran instalados los programas necesarios para la captura, envío y recepción de los datos enviados por la tarjeta. Estos datos serán almacenados en caso de ser necesarios

en la base de datos que posee el servidor TomCat con Axis mediante el protocolo de comunicación SOAP[18]

3.4 Pruebas

El principal objetivo de esta fase es evaluar la calidad del producto que se está desarrollando a través de las diferentes fases por las cuales este pasa, mediante la aplicación de pruebas concretas para validar que las suposiciones hechas en el diseño y los requerimientos se estén cumpliendo satisfactoriamente, esto quiere decir que se verifica que el producto funcione como se diseñó y que los requerimientos se cumplieron.

Como forma de verificar los objetivos trazados, se llevó a cabo un proceso de pruebas que validará y le dará un nivel de calidad a la solución implementada. Para ello se llevó a cabo las pruebas de unidad, de aceptación y del sistema.

3.4.1 Pruebas de Unidad

Nivel de prueba: Pruebas de unidad.

Objetivo: Detectar errores en los datos, lógica y algoritmos.

Participante: Programador.

Ambiente: Desarrollo.

Método de prueba - Caja Blanca: prueba con acceso al código fuente (lógica y datos). Se trabaja con entradas, salidas y el conocimiento interno.

Herramientas empleadas: JUnit para Java – VisualStudio.QualityTools.UnitTestFramework para C# - JCardManager para JavaCard

Resultados: Proporcionar una mejora en la documentación del código, una clara separación de la interfaz y la implementación, además los errores están más centralizados y son más fáciles de localizar.

Los programadores continuamente escriben pruebas unitarias antes de empezar a codificar lo que hará más sencillas y efectivas las pruebas finales que se corren reiteradamente a lo largo de todo el proyecto, asegurando siempre el funcionamiento correcto; de esta forma se evitan las ambigüedades y los requerimientos quedan afinados en la prueba, mientras que el cliente especifica las pruebas funcionales.

A continuación se muestra las pruebas unitarias más significativas de cada componente:

Componente JCAEP – Método Convertir TLV

18 Single Object Access Protocol: define como un cliente se comunica con un servidor usando HTTP y XML como mecanismo de intercambio de información.

En este método se prueba el correcto funcionamiento de la conversión de un APDU respuesta en cualquier arreglo de byte, para la interpretación del mismo en el componente JCAEP. De forma general los pasos a seguir fueron los siguientes: se almacenan los datos que vienen del APDU respuesta en arreglo de byte, se toma un subconjunto del mismo eliminando solamente la primera posición, luego se obtienen la cantidad de bytes necesarios para guardar la longitud de los datos y su cantidad. Se declara un arreglo con la cantidad de datos devueltos, almacenando el subconjunto obtenido en este. En la figura 17 se muestra una imagen de la prueba echa a traves de la herramienta que brinda el VisualStudio donde se recogen los resultados obtenidos.

```
[TestMethod()]
    public void Convert_TLVTest()
    {
        JCEProxy target = new JCEProxy(); // TODO: Initialize to an appropriate value
        APDUResponse _Response = new APDUResponse(new byte[] { 174, 9, 174, 1, 5, 174, 1, 50, 174, 1, 100 }); // TODO: Initialize to an appropriate value
        byte[] expected = new byte[2]; // TODO: Initialize to an appropriate value
        byte[] actual;
        actual = target.Convert_TLV(_Response);
        Assert.AreEqual(expected, actual);
        Assert.Inconclusive("Verify the correctness of this test method.");
    }
```

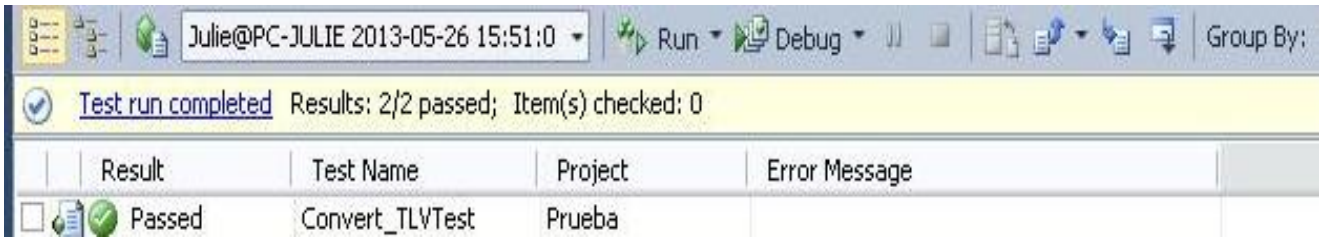

Figura 17 Pruebas de Unidad - Método Convertir TLV

Componente JCAES – Método Procesar resultado de la operación de extensión.

En este procedimiento se prueba el correcto funcionamiento del procesamiento del comando APDU enviado a la tarjeta entregándolo al intérprete de comandos para saber si es una operación de extensión, comprobando la clase e instrucciones. De manera general utilizando la herramienta JUnit que nos brinda el Netbeans se lograron los resultados esperados como se muestra en la figura 18:

```
@Test
 public void testGenericService()  throwsException{
```

```
System.out.println("GenericService");
 byte[] _Data=new byte[9];
 _Data[0]=-82;  _Data[1]=1; _Data[2]=6; _Data[3]=-82; _Data[4]=1; _Data[5]=50;
 _Data[6]=-82;_Data[7]=1;  _Data[8]=100;
  ServerOnline  instance = new ServerOnline();
  byte[] expResult= new byte[2];
  expResult[0]=6;
   expResult[1]=5;
   byte[] result = instance.GenericService(_Data);
   assertArrayEquals(expResult,  result);
    // TODO  review  the generated  test code and remove  the default  call to fail.
    // fail ("The  test case is a prototype.");
  }
```

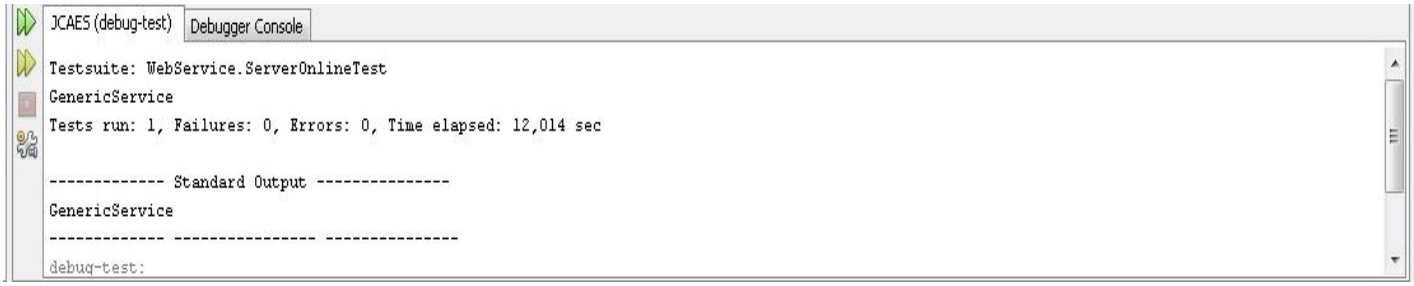

Figura 18 Prueba de Unidad - Método Procesar resultado de la operación de extensión.

Componente JCAEC

El componente JCAEC desarrollado con la herramienta Developer Suite se probó con el JCardManager permitiéndo depurar el applet con las funcionalidades de la operación de extensión, lográndose el resultado esperado como se muestra en la figura 19.

Con la instrucción 10 del comando APDU enviando se prueba la selección de applet retornando el resultado 90 00 ejecutándose correctamente.

Con la instrucción 01 del comando APDU enviando se prueba el envío del APDU a la tarjeta retornando el resultado 90 00 ejecutándose correctamente.

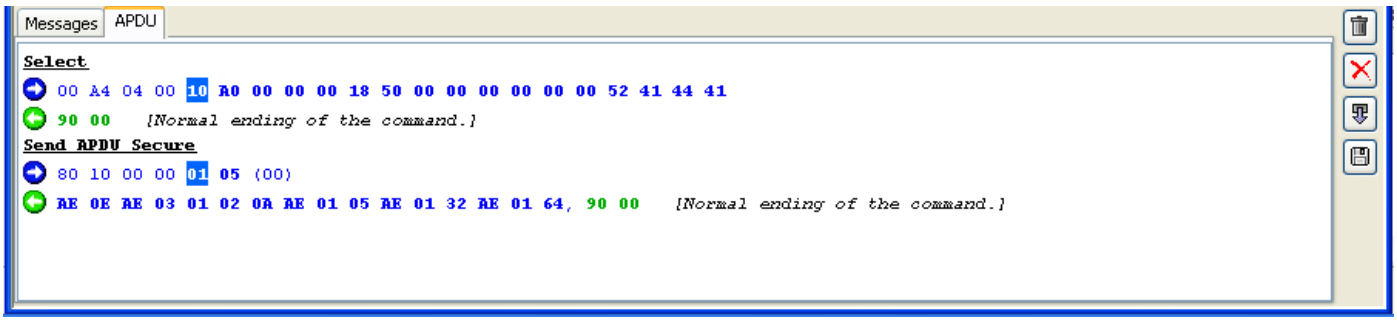

Figura 19 Prueba de Unidad al componte JCAEC

3.4.2 Pruebas de Aceptación

Nivel de prueba: Pruebas de aceptación

Objetivo: Detectar fallas en la implementación del sistema

Participante: Probador, Analista y Cliente

Ambiente: Productivo

Método de prueba - Funcional o Caja negra o de Comportamiento: el enfoque de este tipo de prueba se basa en el análisis de los datos de entrada y en los de salida

Resultados: Permitir al cliente saber cuándo el sistema funciona, y que los programadores conozcan que es lo que resta por hacer.

Mediante la planificación y en base a las especificaciones de las historias de usuarios, se crea las pruebas de aceptación, también denominadas pruebas de funcionalidad, las mismas son constantes y constituyen uno de los pilares básicos de la metodología XP, permitiendo reducir el número de errores e incrementar la calidad del producto. A continuación se muestran los casos de pruebas más significativos para cada componente de la aplicación y sus resultados, el resto las pueden encontrar en el epígrafe Anexo. (Ver Anexo 7: Casos de Pruebas).

Componente JCAEP

Prueba de Aceptación	
Código del caso de prueba: HU3_CP3	**Nombre de la historia de usuario:** Recibir comando APDU.
Nombre del responsable : Julié Arianne Pérez Vive	
Nombre del caso de prueba: Recibir comando APDU.	
Descripción: Se espera un comando APDU enviado por el middleware.	
Condiciones de Ejecución: El comando no puede estar vacío.	
Entradas: Comando APDU 80 10 00 00 00 00 05	
Resultado Esperado:	
Evaluación: Prueba satisfactoria	

Tabla 15 HU3_CP3 Recibir comando APDU.

Componente JCAEC

Prueba de Aceptación

Código del caso de prueba: HU11_CP11	**Nombre de la historia de usuario:** Conformar respuesta de la petición realizada.
Nombre del responsable: Amed AlfonsoRíos	
Nombre del caso de prueba: Conformar respuesta de la petición realizada.	
Descripción: Se conforma una estructura TLV con los datos definidos para la operación de extensión.	
Condiciones de Ejecución: Obtener una lista de arreglo de bytes	
Entradas: Arreglo de byte [] {05}, byte [] {50}, byte [] {100}.	
Resultado Esperado: Estructura TLV AE 09 AE 01 05 AE 01 50 AE 01 100	
Evaluación: Prueba satisfactoria	

Tabla 16 HU11_CP11 Conformar respuesta de la petición realizada.

Componente JCAES

Prueba de Aceptación	
Código del caso de prueba: HU16_CP16	**Nombre de la historia de usuario:** Ejecutar operación de extensión.
Nombre del responsable: Julié Arianne Pérez Vive	
Nombre del caso de prueba: Ejecutar operación de extensión.	
Descripción: Se ejecuta la operación de extensión solicitada por el componente Proxy.	
Condiciones de Ejecución: Los datos obtenidos no pueden estar vacío.	
Entradas: Recibe un identificador y una lista de arreglo de byte.	
Resultado Esperado: 10	
Evaluación: Prueba satisfactoria.	

Tabla 17 HU16_CP16 Ejecutar operación de extensión.

3.4.3 Pruebas del Sistema

Con el objetivo de verificar el cumplimiento de todos los requisitos funcionales, considerando el producto de software final al completo, en un entorno de sistema, se realizaron pruebas al sistema luego de haber sido terminada completamente las funcionalidades de cada componente de la plataforma. En los casos de falla se guardaron las no conformidades encontradas y tras el análisis y corrección de los errores se procedió a otra iteración de pruebas para estas funcionalidades, hasta que las mismas fuesen vencidas

completamente por el sistema. En la siguiente tabla se presentan los resultados de la ejecución de las operaciones definidas utilizando los componentes de la plataforma.

Cabe decir que en este caso los probadores fueron los mismos desarrolladores y los especialistas en el tema del departamento de Tarjetas Inteligentes del CISED.

Tipos de Operaciones	Tarjeta				Servidor				Cobertura
	Tiempo de respuesta (x iteración)			Cantidad de instrucciones	Tiempo de respuesta (x iteración)			Cantidad de instrucciones	
	1era	2da	3era	(x línea de código)	1era	2da	3ra	(x línea de código)	
Almacenamiento	5s	4s	1s	5	5s	1s	1s	12	100%
Consulta de Información	5s	3s	1s	6	5s	3s	1s	10	
Procesamiento (Método Factorial)	4s	10s	14s	8	10s	5s	3s	15	
Total	14s	17s	16s	17	20s	9s	5s	27	

Tabla 18 Análisis de las operaciones de extensión.

3.4.3 Análisis de los resultados

Tras la ejecución de las diferentes pruebas de software se alcanzaron los resultados que se recogen en la gráfica de la figura 20.

Para un total de 92 pruebas realizadas durante todo el ciclo del proyecto, dividido en 3 iteraciones y una especial para una revisión con un grupo de experto del centro CISED se detectaron un máximo de 25 no conformidades, que son simplemente oportunidades de mejora para la organización, ya que nos indican, dónde está el problema, para que lo trabajemos y podamos mejorarlo. Durante la primera iteración se realizaron 4 pruebas de unidad, 6 pruebas de aceptación y 1 al sistema, detectándose 14 no conformidades a las cuales se le dieron solución. En la segunda iteración se realizaron 6 pruebas de unidad, 12 de aceptación y 2 al sistema, mostrándose 8 no conformidades las mismas fueron resueltas en su totalidad. En la tercera iteración se realizaron 14 pruebas de unidad, 18 de aceptación y 2 al sistema,

detectándose 3 no conformidades. En las 3 iteraciones efectuadas las no conformidades detectadas, en su mayoría respondían a errores de bajo impacto en el correcto funcionamiento de la aplicación y todas tuvieron solución un tiempo máximo de 3 días. Posteriormente se realizó una iteración adicional para verificar que no existieran no conformidades, la misma arrojó un resultado de cero no conformidades, lo que indica que los componentes de la plataforma de extensión que permite extender las capacidades de procesamiento y memoria de los applets Java Card en tarjetas inteligentes presentan buena calidad.

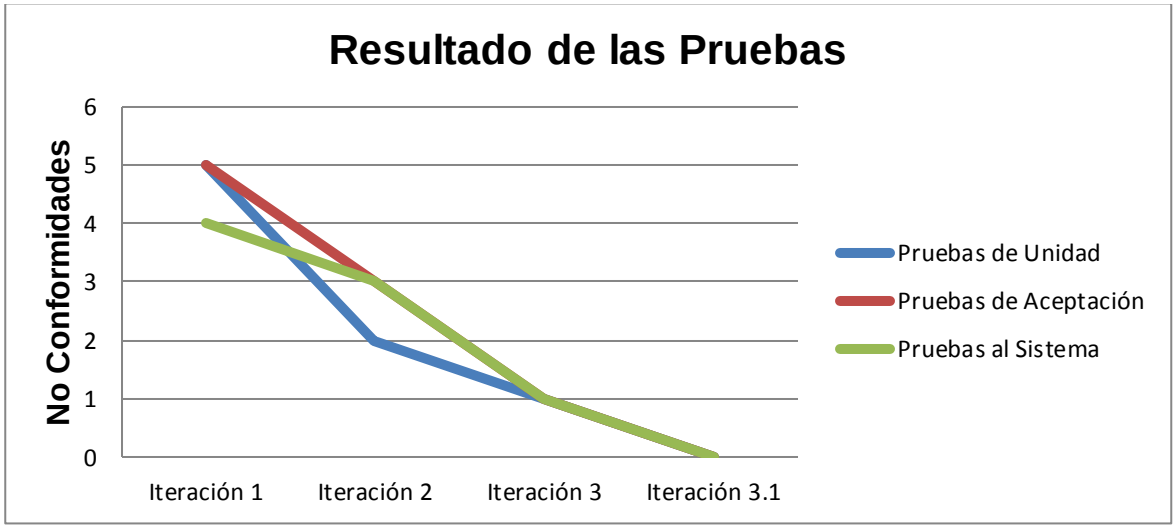

Figura 20 Resumen de No Conformidades

Se logró el éxito de todas las pruebas realizadas a los componentes, aunque en algunas el resultado fue satisfactorio luego de algunas iteraciones. Las mismas fueron de suma importancia al determinar el correcto o no funcionamiento de cada una de las funcionalidades que iban surgiendo, además permitieron la agilización del trabajo del equipo al ir marcando puntos de control hasta donde se sabía que la aplicación funcionaba bien, y permitía enfocar todos los esfuerzos en los fragmentos de código que hasta ese momento no habían sido comprobados. Se demostró que los componentes desarrollados para la plataforma cumplen con todas las funcionalidades, comprobándose que la ejecución de las distintas operaciones fuera de la tarjeta es mucho más rápida y factible.

3.5 Conclusiones del Capítulo

- Con el diseño del diagrama de despliegue se mostró las distribución física que tendría la solución, brindando una distribución completa del acople de los distintos componentes de hardware y software.
- Con la distribución de los paquetes y componentes a partir del diagrama de componentes se brindó una solución multiplataforma en la cual tanto en sistemas operativos propietarios como libres sea capaz de ejecutarse los componentes de la plataforma de extensión.

- Tras haber sido efectuadas todas las pruebas previstas y haberse alcanzado los 100% de las pruebas con resultados satisfactorios, se puede concluir que la aplicación es segura, robusta y cumple con los objetivos trazados. La misma es por tanto capaz de dar solución al problema de la investigación extendiendo las capacidades de procesamiento y memoria de los applets Java Card, basados en tarjetas inteligentes.

CONCLUSIONES GENERALES

A partir del cumplimiento de los objetivos trazados y con la realización de las tareas de la investigación que permitieron el desarrollo de los componentes de la plataforma de extensión, se arriban algunas de las conclusiones luego de terminada la primera versión de esta:

- Se realizó un estudio de los principales conceptos y tecnologías relacionadas con tarjetas inteligentes, además de las soluciones existentes que resolvían el problema de memoria y procesamiento de estas tarjetas, que permitió elaborar el marco teórico de la investigación.
- Se identificaron las herramientas, lenguajes, tecnologías, metodología y estándares, que permitieron flexibilidad y simplicidad en la obtención de los componentes de la plataforma, además de realizar el modelado y la implementación de la solución.
- La implementación de los componentes de la plataforma cumplieron con los requerimientos definidos para el desarrollo de los mismos, brindando la posibilidad de ser utilizada en los sistemas operativos Linux y Windows.
- La realización de un conjunto de pruebas a los componentes de la plataforma logró determinar el correcto funcionamiento de cada una de las funcionalidades implementadas, el cumplimiento de los requisitos especificados por el cliente, conjuntamente con los objetivos de este trabajo lograr ejecutar algoritmos costosos en menor tiempo y ampliar la capacidad de almacenamiento de la tarjeta Java Card.

RECOMENDACIONES

Tras haber finalizado el desarrollo de los componentes de la plataforma de extensión en su primera versión se recomienda:

- Implementar los mecanismos de seguridad entre los componentes de la plataforma que permita mantener el ambiente seguro, lo que es característico en estos tipos de dispositivos.
- Desarrollar una versión del componente JCAEP que se integre al OpenCardFramework

REFERENCIAS BIBLIOGRAFÍCAS

1. Carlos Nelson Henríquez Miranda , F.J.R.T., Diseño e implementación de tecnología basada en dispositivos inteligentes para apoyo a diferentes servicios educativos de la Universidad Autónoma del Caribe. Diciembre 2008. **Prospectiva Vol. 6**: p. 48.

2. Rodriguez, J., Tarjetas Inteligentes, in Tarjetas Inteligentes.pdf. 2001.

3. Msc Celeste Campos Vazques, A.G.A., Andres Marin Lopez, Ignacio Diaz Ansejo, Carlos Delgado Kloos, Carlos Garcia Rubio, Peter T. Breuer JCCM*: Java Card Certificate Management, in E-TICKET CYCYT No 2FD1997-1269-C02-01. 2000, Universidad Carlos III Madrid España. Avd. Universidad 30 28911 Leganés p. 9-10.

4. Microsystems, S., Java Card 3 Plataform Whitepaper, S. Microsystems, Editor. 2008, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA.

5. (ChaTo), C.C., JavaCard: CC51B. Informe final sobre el tema JavaCard, para el curso Arquitectura de Computadores;, 2010.

6. Cap, C.H., N. Maibaum, and L. Heyden, Extending the Date Storage Capabilities of a Java-based SmartCard. 2001, Chair for Information and Communication Services.

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

8. S.A, S.S. Tarjeta Inteligente, Lectores de Tarjetas Inteligentes. 2013; Available from: http://www.scssa.com.ar/tarjetas-inteligentes.htm.

9. GlobalPlatform., Card Specification Version 2.1.1. 2003.

10. Microsystem, S., Conozca mas sobre la tecnologia Java. 2012.

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

12. Crespo, J.B., Cifrado de la información con Algoritmos Simétricos Usando Claves de Identificacion Única de la Java Card para la Java ME, in Departamento de Ingieneria de Sistemas y Automática. Área de Telemática 2010, Universidad de Sevilla: España.

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

14. Z, C., “Java Card™ Technology for Smart Cards: Architecture and Programmer's Guide”. 2002: Addison-Wesley.

15. ORACLE. JAVA CARD TECHNOLOGY. About Java Card Platform

2012 [cited 2013; Available from: http://www.oracle.com/technetwork/java/javame/javacard/overview/about/.

16. Forum, J.J.C., THE ROAD TO THE NEXT GENERATION. 2007.

17. Sun Microsystems, I., Next Generation Java Card Plataform. 2007.

18. Beck, K., "Extreme Programming Explained. Embrace Change". 1999: Pearson Education.

19. Wells. Extreme Programming. 2011; Available from: http://www.extremeprogramming.org/.

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

21. Softpedia. Altova UModel Enterprise Edition 2010 2010; Available from: http://www.softpedia.com/get/Programming/Other-Programming-Files/Altova-UModel.shtml.

22. http://www.gemalto.com/., G. DeveloperSuite. 2010.

23. msdn.microsoft.com. Introducción al lenguaje C# y .NET Framework. 2013.

24. Tschernig. Tecnologia al Insante. 2013; Available from: http://www.tecnologiahechapalabra.com/tecnologia/glosario_tecnico/.

25. Msdn. Lenguaje Visual C#. Porque utilizar C# 2013; Available from: http://msdn.microsoft.com/es-es/library/.

26. free, G.s.t.b. Gemalto .NET Smart Card Framework. 2011; Available from: http://www.gemalto.com/products/dotnet_card/dotnet_framework.html.

27. TheFreeDictionary, Microsoft .Net, in Encyclopedia, Farlex, Editor. 2012.

28. Library, M. Visual Studio 2005. 2012; Available from: http://msdn.microsoft.com/es-ec/library/.

29. Library, M. Marco de pruebas unitarias. 2005; Available from: http://msdn.microsoft.com/es-es/library/microsoft.visualstudio.testtools.unittesting%28v=vs.80%29.aspx.

30. ORACLE. ¿Qué es la tecnología Java y por qué lo necesito? . 2012; Available from: http://www.java.com/es/download/faq/whatis_java.xml.

31. Ubuntu, G. NetBeans. 2012 [cited 16:40 23 feb 2012.; Available from: www.netbeans.org.

32. Juan Manuel Fernández Peña, Pruebas de unidad utilizando JUnit 2005.

33. Pérez, H.F., JUnit: Manual Básico. Artículos técnicos, noticias y desarrollo aplicaciones Java, PL/SQL Server Pages (PSP), …, 2011.

34. Axis.org. Herramientas Java. 2012; Available from: http://axis.apache.org/axis/.

35. Navarro, M., Axis y Axis2, contenedores de Web Services. 2010.

36. PostgreSQL, G.d.l.C.p.l.h.G.d. Herramientas GUI de PostgreSQL. 2012 [cited 18 May 2012, at 18:13.; Available from:

http://wiki.postgresql.org/wiki/Gu%C3%ADa_de_la_Comunidad_para_las_herramientas_GUI_de_PostgreSQL.

37. Argentina, G.d.U.d.P. pgAdmin 3. 2011; Available from: http://www.arpug.com.ar/trac/wiki/PgAdmin.

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

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

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

41. Amaro Calderón, S.D. and V.R.J. Carlos, Metodologías Ágiles, in Facultad de Ciencias Físicas y Matemáticas. Escuela de Informática. 2007, Universidad Nacional de Trujillo: Trujillo, Peru.

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

43. Caamal, R., Diagrama de clases. Material academico, 2012.

44. Naranjo, M., Fundamentos de Definición de Arquitectura en RUP, IFM y SunTone AM. The Software Architecture And Engineering Company, 2010.

45. Etcheverry, I.L., ARQUITECTURA DE UN SISTEMA DE INFORMACIÓN. 2010, Universidad de Uruguay.

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

47. Larman, C., UML y Patrones

Introducción al análisis y diseño orientado a objetos, ed. d. Edicion. 2002: Prentice-Hall.

48. Erich Gamma, R.H., Ralph Johnson y John Vlissides., Design Patterns: Elements of Reusable Object-Oriented Software. Versión española de 2003 ed, ed. Addison-Wesley. 1995.

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

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

BIBLIOGRAFÍA CONSULTADA

- Astudillo, M. V. y. H. Patrones de Diseño. Fundamentos de Ingeniería de Software. D. d. Informática. Universidad Técnica Federico Santa María, inf.utfsm.cl. Consultada el 3/14/2013.
- Bergeron, B. (2003). "Bioinformatics Computing". Uruguay, Pearson Education. Consultada el 3/1/2013.
- Booch, G. "Software Architecture". Consultada el 3/1/2013.
- Chile, A. d. I. (2005). Glosario Applet. AIChile Arquitecto de la información. Consultada 2/13/2013
- Elmasri & Navathe, A. W. (2007). "Fundamentals of Database Systems 5th Ed." Fundamentals of Database Systems. Uruguay. Consultada 3/1/2013.
- Frank Buschmann et al., W. (1996). Pattern-Oriented Software Architecture Vol 1. A System of Patterns. Consultada 3/1/2013.
- Fregozo, J. A. G. (2006). Plataforma Java Card. México, Grupo de Investigación en Tarjetas Inteligentes (GRINTAIN). Consultada 12/6/2012.
- Giorgio, R. D. "Understanding JavaCard 2.0." Consultada 12/6/2012.
- Macroseguridad (2012). "Smart Card / Java Card ". from http://www.macroseguridad.net/productos/smartcards/java_card/. Consultada el 2/26/2013.
- Markantonakis, K. E. M. a. K. (2008). Smart Cards, Tokens, Security and Applications, Springer Science+Business Media, LLC. Consultada 12/6/2012.
- Mestras, J. P. (2004). Patrones de diseño orientado a objetos. Programación Orientada a Objetos. D. I. d. S. e. I. Artificial. Universidad Complutense Madrid, Facultad de Informática. Consultada 3/14/2013.
- Molina, J. G. Patrones de Diseño. Análisis y Diseño de Software. D. d. I. y. Sistemas. Universidad de Murcia. Consultada 3/14/2012.
- Ortiz, C. E. (2003) An Introduction to Java Card Technology - Part 3, The Smart Card Host Application. Sitio Web Oracle Consultada 3/11/2013.

GLOSARIO DE TÉRMINOS

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

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

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

CAP: estándar para la conversión de applets y es un nuevo formato de archivo diseñado especialmente para Java Card.

Framework: Es una estructura de soporte definida en la cual un proyecto de software puede ser organizado y desarrollado.

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

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

Microprocesador: es un circuito integrado que contiene todos los elementos necesarios para conformar una unidad central de procesamiento (CPU por sus siglas en inglés).

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

TLV: Etiqueta, Longitud y Valor.

SOAP: es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Es uno de los protocolos utilizados en los servicios Web.

XML: *Lenguaje de Marcas Extensible*. Se trata de un metalenguaje (un lenguaje que se utiliza para decir algo sobre otro lenguaje) extensible de etiquetas que fue desarrollado por el Word Wide Web Consortium (W3C), un consorcio internacional que elabora recomendaciones para la World Wide Web.

ANEXOS

Anexo 1: Historia de Usuario

Componente JCAEP

Historia de Usuario	
Número: HU_1	**Usuario:** Desarrollador
Nombre historia: Realizar conexión.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Julié Arianne Pérez Vive	
Descripción: El componente JCAEP implementará un método conectar usando una interfaz del SmartCardFramework que permita la conexión con lectores.	
Observaciones: En caso de que no haya lector conectado el sistema le notificará la ausencia de los mismos al usuario.	

Tabla 19 HU_1 Realizar conexión.

Historia de Usuario	
Número: HU_2	**Usuario:** Desarrollador
Nombre historia: Obtener lectores disponibles.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Julié Arianne Pérez Vive	
Descripción: El componente JCAEP implementara un método usando una interfaz del SmartCardFramework, que permite obtener una lista de lectores disponibles.	
Observaciones: En caso de que no exista ningún lector conectado al ordenador, el sistema lanza una excepción.	

Tabla 20 HU_2 Obtener lectores disponibles.

Historia de Usuario	
Número: HU_3	**Usuario:** Desarrollador
Nombre historia: Recibir comando APDU	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 2

Programador responsable: Julié Arianne Pérez Vive
Descripción: Al iniciar la comunicación el middleware envía el comando APDU al componente JCAEP, comenzando así el intercambio de información.
Observaciones: Si la tarjeta no está en estado conectada en el lector, el sistema lanza una excepción. Si el comando enviado no es correcto el sistema lanza una excepción.

Tabla 21 HU_3 Recibir comando APDU

Historia de Usuario	
Número: HU_6	**Usuario:** Desarrollador
Nombre historia: Enviar petición de extensión al servidor	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Amed Alfonso Ríos	
Descripción: Envía los datos al componente JCAES, obtenidos del APDU respuesta del componente JCAEC para realizar la operación de extensión mediante un arreglo de byte, que contiene el identificador y los parámetros necesarios para la ejecución de la operación.	
Observaciones:	

Tabla 22 HU_6 Enviar petición de extensión al servidor

Historia de Usuario	
Número: HU_7	**Usuario:** Desarrollador
Nombre historia: Enviar APDU respuesta de operación de extensión a la tarjeta.	
Prioridad en negocio: Alta	**Riesgo en desarrollo**: Medio
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Amed Alfonso Ríos	
Descripción: El componente JCAEP conforma un APDU con los datos de la operación de extensión; la clase 00, la instrucción AE y los datos de la respuesta recibida del componente JCAES, y le envía este a un applet específico.	
Observaciones:	

Tabla 23 HU_7 Enviar APDU respuesta de operación de extensión a la tarjeta.

Historia de Usuario	
Número: HU_8	Usuario:

Nombre historia: Enviar APDU respuesta al middleware.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Amed Alfonso Ríos	
Descripción: El componente JCAEP interpreta el APDU recibido del applet y envía el APDU respuesta al middleware concluyendo la comunicación.	
Observaciones:	

Tabla 24 HU_8 Enviar APDU respuestas al middleware.

Historia de Usuario	
Número: HU_9	**Usuario:** Desarrollador
Nombre historia: Realizar desconexión.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 3	**Iteración asignada:** 1
Programador responsable: Julié Arianne Pérez Vive	
Descripción: El componente JCAEP implementará un método desconectar usando una interfaz del SmartCardFramework que permita la desconexión con lectores.	
Observaciones:	

Tabla 25 HU_9 Realizar desconexión.

Componente JCAEC

Historia de Usuario	
Número: HU_12	**Usuario:** Desarrollador
Nombre historia: Enviar APDU respuesta	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Julié Arianne Pérez Vive	
Descripción: Envía un APDU respuesta al componente JCAEP con la estructura TLV conformada con anterioridad para una operación de extensión o una respuesta a un comando APDU enviado.	
Observaciones:	

Tabla 26 HU_12 Enviar comando APDU.

Historia de Usuario

Número: HU_13	**Usuario:** Desarrollador
Nombre historia: Enviar cadena de comandos APDU	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Amed Alfonso Ríos	
Descripción: Envía comandos en cadena al componente JCAEP mediante un método que enviaría los próximos datos almacenados en el buffer de datos por enviar, mientras no se haya completados todos los datos necesarios.	
Observaciones: En caso de que durante el proceso de envío de cadena de comandos el lector es desconectado el sistema lanza una excepción.	

Tabla 27 HU_13 Enviar cadena de comandos APDU.

Historia de Usuario	
Número: HU_14	**Usuario:** Desarrollador
Nombre historia: Recibir un comando APDU de extensión.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Julié Arianne Pérez Vive	
Descripción: Recibe un comando APDU con características que permiten identificar si es un comando de extensión a partir de una comparación entre la clase e instrucción del APDU y la clase 00 e instrucción AE definida para la operación de extensión instrucción.	
Observaciones:	

Tabla 28 HU_14 Recibir un comando APDU de extensión.

Anexo 2: Estimación de Esfuerzo

Componente	No	Historia de Usuario	Tiempo Estimado		
			Semanas Estimadas	Días Estimados	Horas Estimadas
JCAEP	1	Realizar conexión.	1	5	40
	2	Obtener lectores disponibles.	1	5	40
	3	Recibir un comando APDU.	1.5	5.5	44
	4	Enviar un comando APDU a la tarjeta.	1.5	5.5	44
	5	Interpretar comando APDU.	1	5	40
	6	Enviar petición de extensión al servidor.	2.5	10.5	84
	7	Enviar APDU respuesta de operación de extensión a la tarjeta.	1.8	6	48
	8	Enviar APDU respuestas al middleware.	1.5	5.5	44
	9	Realizar desconexión.	1	5	40
JCAEC	10	Recibir un comando APDU.	1.5	5.5	44
	11	Conformar respuesta de la operación a realizar.	2.5	10.5	84
	12	Enviar APDU respuesta.	1.5	5.5	44
	13	Enviar cadena de APDU respuesta.	1.5	5.5	44
	14	Recibir un comando APDU de extensión.	1	5	40
	15	Procesar resultado de la operación de extensión.	1.8	6	48
JCAES	16	Ejecutar operación de extensión.	3	15	120
	17	Almacenar información de tarjeta	2.5	10.5	84
	18	Enviar respuesta de la petición de extensión.	1.5	5.5	44
Tiempo Estimado Total			**28.1**	**222**	**976**

Tabla 29 Estimación de Esfuerzo

Anexo 3: Plan de Entregas

Entregables	No HU	Historia de Usuario	Semanas Estimadas	Iteración Asignadas		
				1	2	3
JCAEP	1	Realizar conexión.	1	X		
	2	Obtener lectores disponibles.	1	X		
	3	Recibir un comando APDU.	1.5	X		
	4	Enviar un comando APDU a la tarjeta.	1.5	X		
	5	Interpretar comando APDU.	1	X		
	6	Enviar petición de extensión al servidor.	2.5	X		
	7	Enviar APDU respuesta de operación de extensión a la tarjeta.	1.8	X		
	8	Enviar APDU respuestas al middleware.	1.5	X		
	9	Realizar desconexión.	1	X		
JCAEC	10	Recibir un comando APDU.	1.5		X	
	11	Conformar respuesta de la operación a realizar.	2.5		X	
	12	Enviar APDU respuesta.	1.5		X	
	13	Enviar cadena de APDU respuesta.	1.5		X	
	14	Recibir un comando APDU de extensión.	1		X	
	15	Procesar resultado de la operación de extensión.	1.8		X	
JCAES	16	Ejecutar operación de extensión.	3			X
	17	Almacenar información de tarjeta	2.5			X
	18	Enviar respuesta de la petición de extensión.	1.5			X
Total 28.1						

Tabla 30 Plan de Entregas

Anexo 4: Plan de Iteraciones

Iteraciones	No HU	Historia de Usuario	Duración (semanas)	Comienzo	Fin
1	1	Realizar conexión.	13.3	Noviembre 2012	Enero 2013
	2	Obtener lectores disponibles.			
	3	Recibir un comando APDU.			
	4	Enviar un comando APDU a la tarjeta.			
	5	Interpretar comando APDU.			
	6	Enviar petición de extensión al servidor.			
	7	Enviar APDU respuesta de operación de extensión a la tarjeta.			
	8	Enviar APDU respuestas al middleware.			
	9	Realizar desconexión.			
2	10	Recibir un comando APDU.	9.8	Enero 2013	Marzo 2013
	11	Conformar respuesta de la operación a realizar.			
	12	Enviar APDU respuesta.			
	13	Enviar cadena de APDU respuesta.			
	14	Recibir un comando APDU de extensión.			
	15	Procesar resultado de la operación de extensión.			
3	16	Ejecutar operación de extensión.	7	Marzo 2013	Mayo 2013
	17	Almacenar información de tarjeta			
	18	Enviar respuesta de la petición de extensión.			

Tabla 31 Plan de Iteraciones

Anexo 5: Tareas de Ingeniería

No HU	Nombre de Tarea	Duración (semanas)	Comienzo	Fin
HU1	Realizar conexión			
T1	Diseño de la tarjeta CRC JACEProxy.cs			
T2	Graficación de los diagramas de clase, componente y despliegue.			
T3	Implantación método Connect() del componente JCAEP			
T4	Realización de prueba de unidad al método Connect().			
T5	Documentación del caso de prueba Conexión correcta al sistema			
HU2	Obtener lectores disponibles.			
T1	Diseño de la tarjeta CRC JACEProxy.cs			
T2	Graficación del diagrama de clase, componente y despliegue.			
T3	Implantación método ListReaders() del componente JCAEP			
T4	Realización de prueba de unidad al método ListReaders() .			
T5	Documentación del caso de prueba Listar lectores			
HU3	Recibir un comando APDU.			
T1	Diseño de la tarjeta CRC JACEProxy.cs			
T2	Graficación del diagrama de clase, componente y despliegue.			
T3	Implantación método Send_APDU() del componente JCAEP			
T4	Realización de prueba de unidad al método Send_APDU() .			
T5	Documentación del caso de prueba Recibir comando APDU			
HU4	Enviar un comando APDU a la tarjeta.			
T1	Diseño de la tarjeta CRC JACEProxy.cs			
T2	Graficación del diagrama de clase, componente y despliegue.			
T3	Implantación método Send_APDU() del componente JCAEP			
T4	Realización de prueba de unidad al método Send_APDU() .			
T5	Documentación del caso de prueba Enviar un comando APDU a la tarjeta.	13.3	Noviembre 2012	Enero 2012
HU5	Interpretar comando APDU.			

T1	Diseño de la tarjeta CRC JACEProxy.cs			
T2	Graficación del diagrama de clase, componente y despliegue.			
T3	Implantación método Send_APDU() del componente JCAEP			
T4	Realización de prueba de unidad al métodoSend_APDU() .			
T5	Documentación del caso de prueba Interpretar comando APDU.			
HU6	Enviar petición de extensión al servidor.			
T1	Diseño de la tarjeta CRC JACEProxy.cs			
T2	Graficación del diagrama de clase, componente y despliegue.			
T3	Implantación método SendRequestServer() del componente JCAEP			
T4	Realización de prueba de unidad al método SendRequestServer() .			
T5	Documentación del caso de prueba Enviar petición de extensión al servidor.			
HU7	Enviar APDU respuesta de operación de extensión a la tarjeta.			
T1	Diseño de la tarjeta CRC JACEProxy.cs			
T2	Graficación del diagrama de clase, componente y despliegue.			
T3	Implantación método Send_APDU() del componente JCAEP			
T4	Realización de prueba de unidad al método Send_APDU() .			
T5	Documentación del caso de prueba Enviar respuesta de operación de extensión a la tarjeta			
HU8	Enviar APDU respuesta al middleware.			
T1	Diseño de la tarjeta CRC JACEProxy.cs			
T2	Graficación del diagrama de clase, componente y despliegue.			
T3	Implantación método Send_APDU() del componente JCAEP			
T4	Realización de prueba de unidad al método Send_APDU() .			
T5	Documentación del caso de prueba Enviar APDU respuesta al middleware.			
HU9	Realizar desconexión.			
T1	Diseño de la tarjeta CRC JACEProxy.cs			
T2	Graficación del diagrama de clase, componente y despliegue.			

T3	Implantación método Disconnect() del componente JCAEP			
T4	Realización de prueba de unidad al método Disconnect() .			
T5	Documentación del caso de prueba Realizar desconexión.			
HU10	Recibir comando APDU.	9.8	Enero 2013	Marzo 2013
T1	Diseño de la tarjeta CRC CustomAppExtManager.class			
T2	Graficación del diagrama de clase, componente y despliegue.			
T3	Implantación método ProcessAPDU() del componente JCAEC			
T4	Realización de prueba de unidad al método ProcessAPDU() .			
T5	Documentación del caso de prueba Recibir comando APDU.			
HU11	Conformar respuesta de la petición realizada.			
T1	Diseño de la tarjeta CRC CustomAppExtManager.class.			
T2	Graficación del diagrama de clase, componente y despliegue.			
T3	Implantación método ExtensionOPInvoke() del componente JCAEC			
T4	Realización de prueba de unidad al método ExtensionOPInvoke() .			
T5	Documentación del caso de prueba Conformar respuesta de la petición realizada.			
HU12	Enviar APDU respuesta.			
T1	Diseño de la tarjeta CRC CustomAppExtManager.class.			
T2	Graficación del diagrama de clase, componente y despliegue.			
T3	Implantación método () del componente JCAEC.			
T4	Realización de prueba de unidad al método ().			
T5	Documentación del caso de prueba Enviar APDU respuesta.			
HU13	Enviar cadena de APDU respuesta.			
T1	Diseño de la tarjeta CRC CustomAppExtManager.class.			
T2	Graficación del diagrama de clase, componente y despliegue.			
T3	Implantación método NextDates() del componente JCAEC			
T4	Realización de prueba de unidad al método NextDates() .			
T5	Documentación del caso de prueba Enviar cadena de APDU respuesta.			

HU14	Recibir comando APDU de extensión.			
T1	Diseño de la tarjeta CRC CustomAppExtManager.class.			
T2	Graficación del diagrama de clase, componente y despliegue.			
T3	Implantación método () del componente JCAEC.			
T4	Realización de prueba de unidad al método() .			
T5	Documentación del caso de prueba Recibir comando APDU de extensión.			
HU15	Procesar resultado de la operación de extensión.			
T1	Diseño de la tarjeta CRC CustomAppExtManager.class.			
T2	Graficación del diagrama de clase, componente y despliegue.			
T3	Implantación método ProcessResult() del componente JCAEC.			
T4	Realización de prueba de unidad al método ProcessResult() .			
T5	Documentación del caso de prueba Procesar resultado de la operación de extensión.			
HU16	Ejecutar operación de extensión.	7	Marzo 2013	Mayo 2013
T1	Diseño de la tarjeta CRC ServiceOnline.java.			
T2	Graficación del diagrama de clase, componente y despliegue.			
T3	Implantación método GenericService() del componente JCAES.			
T4	Realización de prueba de unidad al método GenericService() .			
T5	Documentación del caso de prueba Ejecutar operación de extensión.			
HU17	Almacenar información de la tarjeta.			
T1	Diseño de la tarjeta CRC ServiceOnline.java.			
T2	Graficación del diagrama de clase, componente y despliegue.			
T3	Implantación método GenericService() del componente JCAES.			
T4	Realización de prueba de unidad al método GenericService() .			
T5	Documentación del caso de prueba Almacenar información de la tarjeta.			
HU18	Enviar respuesta de la petición de extensión.			
T1	Diseño de la tarjeta CRC ServiceOnline.java.			
T2	Graficación del diagrama de clase, componente y despliegue.			

T3	Implantación método GenericService() del componente JCAES.			
T4	Realización de prueba de unidad al método GenericService() .			
T5	Documentación del caso de prueba Enviar respuesta de la petición de extensión.			

Tabla 32 Tareas de Ingeniería

Anexo 6: Tarjetas CRC

Nombre de la clase: CustomAppExtManager.class	
Tipo de clase: Controladora	
Atributo:	**Tipo:**
_ControlQuantityByte	short
_DataGoup	byte[]
IExtensionOPResponse	List
_PositionByte	short
Para cada responsabilidad:	
Nombre:	CustomAppExtManager()
Descripción:	Crea una instancia del applet.
Nombre:	AddIExtensionOPResponse(IExtensionOPResponse OResponse)
Descripción:	Método público encargado de adicionar una operación de extensión para cuando se tenga una respuesta de la misma saber a quién pertenece.
Nombre:	ProcessAPDU(APDU apdu)
Descripción:	Método público encargado de procesar el comando APDU enviado a la tarjeta entregándolo al intérprete de comandos para saber si es una operación de extensión, comprobando la clase, instrucciones y P1.
Nombre:	ProcessResult(APDU apdu)
Descripción:	Método público que se encarga de dirigir la respuesta obtenida de la operación de extensión a quien la solicita comprobando el identificador de la operación.
Nombre:	ExtensionOPInvoke(List Data)
Descripción:	Método público que se encarga de conforma el TLV con los parámetros enviados por el customApplet, teniendo en cuenta el formato definido.
Nombre:	TLVdata(byte[]Data)
Descripción:	Método público que se encarga de crear los arreglos de byte que tiene forma de TLV que van dentro de un TLV más grande, para cada parámetro crea un TLV con el identificador, la longitud de los datos que estarán dentro y los datos.
Nombre:	TLVleng(short leng)
Descripción:	Método público que me devuelve la cantidad de byte que hacen falta para escribir la longitud de los datos que voy a enviar en el TLV.

Nombre:	NextDates()
Descripción:	Método público que se encarga de enviar los datos que no caben en el primer APDU, es decir envía una secuencia de datos en caso de la respuesta no llegue completa.

Tabla 33 Descripción de la clase CustomAppExtManager.class

Nombre de la clase: ServiceOnline.java	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	GenericService(byte[] _Data)
Descripción:	Método público que se encarga de controlar la ejecución de las operaciones a realizar por el servidor y conforma la respuesta, solicitada por el componente JACEP.
Nombre:	ConformToByte (byte[] _DataArray)
Descripción:	Método privado que se encarga de construir una lista de arreglos de byte, a partir de la estructura de dato enviada por el componente JCAEP.

Tabla 34 Descripción de la clase ServiceOnline.java

Anexo 7: Casos de Pruebas

Componente JCAEP

Prueba de Aceptación	
Código del caso de prueba: HU1_CP1	**Nombre de la historia de usuario:** Realizar conexión.
Nombre del responsable : Julié Arianne Pérez Vive	
Nombre del caso de prueba: Conexión correcta al sistema	
Descripción: Establece la conexión con la tarjeta.	
Condiciones de Ejecución: Deben existir lectores conectados y una tarjeta.	
Entradas: Recibe un Id de aplicación	
Resultado Esperado: Se realiza la conexión	
Evaluación: Prueba satisfactoria	

Tabla 35 HU1_CP1 Conexión correcta al sistema

Prueba de Aceptación	
Código del caso de prueba: HU2_CP2	**Nombre de la historia de usuario:** Obtener lectores disponibles.
Nombre del responsable : Amed Alfonso Ríos	
Nombre del caso de prueba: Listar lectores	
Descripción: Se obtiene los lista de lectores disponibles	
Condiciones de Ejecución: Debe haber lectores conectados	
Entradas:	
Resultado Esperado: Se muestra lista de lectores conectados	
Evaluación: Prueba satisfactoria	

Tabla 36 HU2_CP2 Listar lectores disponible

Prueba de Aceptación	
Código del caso de prueba: HU4_CP4	**Nombre de la historia de usuario:** Enviar comando APDU a la tarjeta.
Nombre del responsable : Amed Alfonso Ríos	
Nombre del caso de prueba: Enviar comando APDU a la tarjeta.	

Descripción: Envía comando APDU a la tarjeta
Condiciones de Ejecución: Que se haya autenticado la tarjeta
Entradas: Comando APDU 80 10 00 00 00 00 05
Resultado Esperado: APDU respuesta AE 09 AE 01 05 AE 01 50 AE 01 100
Evaluación: Prueba satisfactoria

Tabla 37 HU4_CP4 Enviar comando APDU a la tarjeta.

Prueba de Aceptación	
Código del caso de prueba: HU5_CP5	**Nombre de la historia de usuario:** Interpretar comando APDU.
Nombre del responsable : Julié Arianne Pérez Vive	
Nombre del caso de prueba: Interpretar comando APDU.	
Descripción: Se interpreta si el APDU respuesta es una operación de extensión o no.	
Condiciones de Ejecución: Que reciba una APDU respuesta de la tarjeta.	
Entradas: APDU respuesta AE 09 AE 01 05 AE 01 50 AE 01 100	
Resultado Esperado: Arreglo de byte con los siguientes valores AE 01 05 AE 01 50 AE 01 100	
Evaluación: Prueba satisfactoria	

Tabla 38 HU5_CP5 Interpretar comando APDU.

Prueba de Aceptación	
Código del caso de prueba: HU6_CP6	**Nombre de la historia de usuario:** Enviar petición de extensión al servidor.
Nombre del responsable : Amed Alfonso Ríos	
Nombre del caso de prueba: Enviar petición de extensión al servidor.	
Descripción: Se envía comando APDU con petición de la operación de extensión a realizar	
Condiciones de Ejecución: Haber obtenido datos de la operación de extensión.	
Entradas: Arreglo de byte con los siguientes valores AE 01 05 AE 01 50 AE 01	

100
Resultado Esperado: Arreglo de byte con los valores 05 10
Evaluación: Prueba satisfactoria

Tabla 39 HU6_CP6 Enviar petición de extensión al servidor.

Prueba de Aceptación	
Código del caso de prueba: HU7_CP7	**Nombre de la historia de usuario:** Enviar APDU respuesta de operación de extensión a la tarjeta.
Nombre del responsable : Julié Arianne Pérez Vive	
Nombre del caso de prueba: Enviar APDU respuesta de operación de extensión a la tarjeta.	
Descripción: Se envía APDU respuesta de operación de extensión a la tarjeta.	
Condiciones de Ejecución: Haber obtenido la respuesta del servidor 05 10	
Entradas: Comando APDU 00 AE 02 00 00 05 10	
Resultado Esperado: APDU respuesta 08 09 (90 00)	
Evaluación: Prueba satisfactoria	

Tabla 40 HU7_CP7 Enviar APDU respuesta de operación de extensión a la tarjeta.

Prueba de Aceptación	
Código del caso de prueba: HU8_CP8	**Nombre de la historia de usuario:** Enviar APDU respuesta al middleware.
Nombre del responsable : Amed Alfonso Rios	
Nombre del caso de prueba: Enviar APDU respuesta al middleware.	
Descripción: Se envía el APDU respuesta obtenido al middleware	
Condiciones de Ejecución: Se obtiene respuesta de la tarjeta	
Entradas: Comando APDU 08 09 (90 00).	
Resultado Esperado: Muestra la respuesta en la interfaz del middleware.	
Evaluación: Prueba satisfactoria	

Tabla 41 HU8_CP8 Enviar APDU respuesta al middleware.

Prueba de Aceptación	
Código del caso de prueba: HU9_CP9	**Nombre de la historia de usuario:** Realizar desconexión.
Nombre del responsable : Julié Arianne Pérez Vive	
Nombre del caso de prueba: Realizar desconexión.	
Descripción: Se cierra la conexión con la tarjeta	
Condiciones de Ejecución: Debe de existir alguna tarjeta conectada	
Entradas:	
Resultado Esperado: El lector Gemplus usb SmartCard Reader 0 se encuentra desconectado.	
Evaluación: Prueba satisfactoria	

Tabla 42 HU9_CP9 Realizar desconexión.

Componente JCAEC

Prueba de Aceptación	
Código del caso de prueba: HU10_CP10	**Nombre de la historia de usuario:** Recibir comando APDU.
Nombre del responsable : Amed Alfonso Ríos	
Nombre del caso de prueba: Recibir comando APDU.	
Descripción: Espera un comando APDU e interpreta si es una operación de extensión o no.	
Condiciones de Ejecución: Recibir un APDU que no este vacío.	
Entradas: Comando APDU 80 10 00 00 00 00 05	
Resultado Esperado:	
Evaluación: Prueba satisfactoria	

Tabla 43 HU10_CP10 Recibir comando APDU.

Prueba de Aceptación	
Código del caso de prueba: HU12_CP12	**Nombre de la historia de usuario:** Enviar APDU respuesta.
Nombre del responsable: Julié Arianne Pérez Vive	
Nombre del caso de prueba: Enviar APDU respuesta.	
Descripción: Envía APDU respuesta al componente JCAEP.	
Condiciones de Ejecución: Que el APDU respuesta no sea vacío.	
Entradas: APDU con los valores AE 09 AE 01 05 AE 01 50 AE 01 100	
Resultado Esperado:	
Evaluación: Prueba satisfactoria	

Tabla 44 HU12_CP12 Enviar APDU respuesta.

Prueba de Aceptación	
Código del caso de prueba: HU13_CP13	**Nombre de la historia de usuario:** Enviar cadena de APDU respuesta.
Nombre del responsable: Amed Alfonso Ríos	
Nombre del caso de prueba: Enviar cadena de APDU respuesta.	
Descripción: Se envía en caso de no estar completos los datos una cadena de APDU respuesta.	
Condiciones de Ejecución: Deben quedar datos sin enviar	
Entradas:	
Resultado Esperado:	
Evaluación: Prueba satisfactoria	

Tabla 45 HU13_CP13 Enviar cadena de APDU respuesta.

Prueba de Aceptación	
Código del caso de prueba: HU14_CP14	**Nombre de la historia de usuario:** Recibir comando APDU de extensión.
Nombre del responsable: Julié Arianne Pérez Vive	
Nombre del caso de prueba: Recibir comando APDU de extensión.	
Descripción: Recibe comando APDU con los datos de la operación de extensión realizada.	
Condiciones de Ejecución: Haber obtenido los datos de la operación de extensión.	
Entradas: APDU 00 AE 02 00 00 05 10	
Resultado Esperado:	
Evaluación: Prueba satisfactoria	

Tabla 46 HU13_CP13 Recibir comando APDU de extensión.

Prueba de Aceptación	
Código del caso de prueba: HU15_CP15	**Nombre de la historia de usuario:** Procesar resultado de la operación de extensión.
Nombre del responsable: Amed Alfonso Ríos	
Nombre del caso de prueba: Procesar resultado de la operación de extensión.	
Descripción: Procesa resultado de la operación de extensión.	
Condiciones de Ejecución: Que se haya recibido una operación de extensión.	
Entradas: 00 AE 02 00 00 05 10	
Resultado Esperado: 08 09 (90 00)	
Evaluación: Prueba satisfactoria	

Tabla 47 HU15_CP15 Procesar resultado de la operación de extensión.

Componente JCAES

Prueba de Aceptación	
Código del caso de prueba: HU17_CP17	**Nombre de la historia de usuario:** Almacenar información de la tarjeta.
Nombre del responsable: Amed Alfonso Ríos	
Nombre del caso de prueba: Almacenar información de la tarjeta.	
Descripción: Se almacena cualquier información relacionada con la tarjeta en la base de datos.	
Condiciones de Ejecución: El identificador no sea nulo	
Entradas: Se necesita el identificador de la tarjeta y una dupla entidad valor.	
Resultado Esperado: Se almacenan el identificador de la tarjeta y la dupla entidad valor.	
Evaluación: Prueba satisfactoria.	

Tabla 48 HU17_CP17 Almacenar información de la tarjeta.

Prueba de Aceptación	
Código del caso de prueba: HU18_CP18	**Nombre de la historia de usuario:** Enviar respuesta de la petición de extensión.
Nombre del responsable: Julié Arianne Pérez Vive	
Nombre del caso de prueba: Enviar respuesta de la petición de extensión.	
Descripción: Se envía al componente JCAEP la respuesta de la operación de extensión.	
Condiciones de Ejecución: Se obtuvo la respuesta de la operación de extensión.	
Entradas: 05 10	
Resultado Esperado:	
Evaluación: Prueba satisfactoria	

Tabla 49 HU18_CP18 Enviar respuesta de la petición de extensión.

Anexo 8: Manual de Usuario

Este manual está orientado al programador JavaCard para el trabajo con los componentes de la plataforma de extensión de las capacidades de procesamiento y memoria de los applets JavaCard en tarjetas Inteligentes, elaborado por los estudiantes de informática Julie Arianne Perez Vive y Amed Alfonso Rios.

Esta plataforma permite a los desarrolladores realizar operaciones fuera de la tarjeta, según las necesidades de almacenamiento y procesamiento que tengan las operaciones a realizar. El modelo tradicional cuenta con un middleware que se comunica con la tarjeta directamente, enviando comandos APDU a la misma y recibiendo una respuesta.

El nuevo modelo que se propone, está compuesto por tres componentes: El componente JCAEC debe estar instalado en la tarjeta, para guiar el proceso de conformar las operaciones de extensión e interpretar la petición recibida. El componente JCAES que debe estar instalado en la PC cliente para realizar todas las operaciones de extensión (dígase operaciones de almacenamiento o procesamiento) fuera de la tarjeta y luego debe ser desplegado en el servidor web para ser usado por el componente JCAEP y el componente JCAEP es una dll que se integra al middleware para servir de intermediario en las transmisiones de comandos APDUs entre el middleware y la tarjeta y entre la tarjeta y el servidor. Por lo que el desarrollador debe tener desplegado un sistema distribuido que permita obtener un servicio publicado en un servidor web que contenga una base de datos, al mismo tiempo tener en su PC el framework 1.1 SDK.

Pasos para la utilización de los componentes de la plataforma de extensión.

Paso 1: Tener incorporado dentro de su solución Java Card el API Java Card JCAEC.

Paso 2: Comprobar en el método Process del applet la siguiente sentencia

if(App_ExtManager.ProcessAPDU(apdu)){} para ver si el comando APDU recibido es una operación de extensión, en caso de serlo se ejecuta la operación que puede ser una respuesta de una operación de extensión o la petición de más datos a la tarjeta. Luego la sentencia else{} donde el programador desarrolla las funcionalidades que necesita.

Paso 3: Implementar en el applet la operación de extensión que se quiera realizar, que tiene que heredar de la interfaz IExtensionOPResponse que implementa el API Java Card.

Paso 4: Cuando se envía una operación de extensión se tiene que crear una instancia de la operación requerida y adicionarla a la lista de respuestas de operaciones.

Paso 5: Enviar los datos necesarios para la operación de extensión a la clase CustomAppExtension Manager invocando el método ExtensionOPInvoke que se le pasa por parámetro una lista de arreglos de byte donde en la primera posición está el identificador de la operación de extensión, que tienen que ser el

mismo de la operación definida en el servidor, luego tantos datos como sean necesarios y devuelve una estructura TLV como está definida en el capítulo 3.

Paso 6: Tener incorporado a su solución las librerías asociadas al componente JCAES que permite implementar a las operaciones que se deseen realizar fuera de la tarjeta la interfaz IExtensionOPServer. Esta interfaz contiene El métodos id de tipo integer que devuelve el id de la operación y el método Response que se le pasa una lista de arregloS de byte y devuelve un arreglo de byte.

Paso 7: Editar la estructura XML cambiando la dirección donde estará el módulo desarrollado y poniéndole el nombre del paquete seguido de punto y el nombre de clase.

Paso 8: Tener incorporado dentro de su middleware el componente JCAEP el cual implementa la interfaz ISmartCardReader que posee los métodos para la conexión, desconexión y el envió de comandos APDU a la tarjeta.

Paso 9: Llamar al método SendAPDU del componente JCAEP que se encarga de enviar el comando APDU a la tarjeta y obtiene una respuesta. Si esa respuesta es una operación de extensión se la envía al servidor sino se la envía directamente al middleware.